EVOLUTIONARY ALGORITHMS

Edited by **Eisuke Kita**

INTECHWEB.ORG

Evolutionary Algorithms
Edited by Eisuke Kita

Published by InTech
Janeza Trdine 9, 51000 Rijeka, Croatia

Publishing Process Manager Katarina Lovrecic
Technical Editor Teodora Smiljanic
Cover Designer Martina Sirotic
Image Copyright Designus, 2010. Used under license from Shutterstock.com

First published March, 2011
Printed in India

A free online edition of this book is available at www.intechopen.com
Additional hard copies can be obtained from orders@intechweb.org

Evolutionary Algorithms, Edited by Eisuke Kita
p. cm.
ISBN 978-953-307-171-8

Contents

Preface IX

Part 1 New Algorithms 1

Chapter 1 **Hybridization of Evolutionary Algorithms 3**
Iztok Fister, Marjan Mernik and Janez Brest

Chapter 2 **Linear Evolutionary Algorithm 27**
Kezong Tang, Xiaojing Yuan, Puchen Liu and Jingyu Yang

Chapter 3 **Genetic Algorithm Based on Schemata Theory 41**
Eisuke Kita and Takashi Maruyama

Chapter 4 **In Vitro Fertilization Genetic Algorithm 57**
Celso G. Camilo-Junior and Keiji Yamanaka

Chapter 5 **Bioluminescent Swarm Optimization Algorithm 69**
Daniel Rossato de Oliveira, Rafael S. Parpinelli and Heitor S. Lopes

Chapter 6 **A Memetic Particle Swarm Optimization Algorithm for Network Vulnerability Analysis 85**
Mahdi Abadi and Saeed Jalili

Chapter 7 **Quantum-Inspired Differential Evolutionary Algorithm for Permutative Scheduling Problems 109**
Tianmin Zheng and Mitsuo Yamashiro

Chapter 8 **Quantum-Inspired Particle Swarm Optimization for Feature Selection and Parameter Optimization in Evolving Spiking Neural Networks for Classification Tasks 133**
Haza Nuzly Abdull Hamed, Nikola K. Kasabov and Siti Mariyam Shamsuddin

Chapter 9 **Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures 149**
Ivan Zelinka, Donald Davendra, Roman Senkerik, Roman Jasek and Zuzana Oplatkova

Chapter 10 **PPCea: A Domain-Specific Language for Programmable Parameter Control in Evolutionary Algorithms 177**
Shih-Hsi Liu, Marjan Mernik, Mohammed Zubair, Matej Črepinšek and Barrett R. Bryant

Chapter 11 **Evolution Algorithms in Fuzzy Data Problems 201**
Witold Kosiński, Katarzyna Węgrzyn-Wolska and Piotr Borzymek

Chapter 12 **Variants of Hybrid Genetic Algorithms for Optimizing Likelihood ARMA Model Function and Many of Problems 219**
Basad Ali Hussain Al-Sarray and Rawa'a Dawoud Al-Dabbagh

Chapter 13 **Tracing Engineering Evolution with Evolutionary Algorithms 247**
Tino Stanković, Kalman Žiha and Dorian Marjanović

Part 2 Applications 269

Chapter 14 **Evaluating the α-Dominance Operator in Multiobjective Optimization for the Probabilistic Traveling Salesman Problem with Profits 271**
Bingchun Zhu, Junichi Suzuki and Pruet Boonma

Chapter 15 **Scheduling of Construction Projects with a Hybrid Evolutionary Algorithm's Application 295**
Wojciech Bożejko, Zdzisław Hejducki, Magdalena Rogalska and Mieczysław Wodecki

Chapter 16 **A Memetic Algorithm for the Car Renter Salesman Problem 309**
Marco Goldbarg, Paulo Asconavieta and Elizabeth Goldbarg

Chapter 17 **Multi-Objective Scheduling on a Single Machine with Evolutionary Algorithm 327**
A. S. Xanthopoulos, D. E. Koulouriotis and V. D. Tourassis

Chapter 18 **Evolutionary Algorithms in Decomposition-Based Logic Synthesis 343**
Mariusz Rawski

Chapter 19 **A Memory-Storable Quantum-Inspired Evolutionary Algorithm for Network Coding Resource Minimization 363**
Yuefeng Ji and Huanlai Xing

Chapter 20 **Using Evolutionary Algorithms for Optimization of Analogue Electronic Filters 381**
Lukáš Dolívka and Jiří Hospodka

Chapter 21 **Evolutionary Optimization of Microwave Filters 407**
Maria J. P. Dantas, Adson S. Rocha,
Ciro Macedo, Leonardo da C. Brito,
Paulo C. M. Machado and Paulo H. P. de Carvalho

Chapter 22 **Feature Extraction from High-Resolution Remotely Sensed Imagery using Evolutionary Computation 423**
Henrique Momm and Greg Easson

Chapter 23 **Evolutionary Feature Subset Selection for Pattern Recognition Applications 443**
G.A. Papakostas, D.E. Koulouriotis,
A.S. Polydoros and V.D. Tourassis

Chapter 24 **A Spot Modeling Evolutionary Algorithm for Segmenting Microarray Images 459**
Eleni Zacharia and Dimitris Maroulis

Chapter 25 **Discretization of a Random Field – a Multiobjective Algorithm Approach 481**
Guang-Yih Sheu

Chapter 26 **Evolutionary Algorithms in Modelling of Biosystems 495**
Rosario Guzman-Cruz, Rodrigo Castañeda-Miranda, Juan García-Escalante, Luis Solis-Sanchez, Daniel Alaniz-Lumbreras, Joshua Mendoza-Jasso, Alfredo Lara-Herrera, Gerardo Ornelas-Vargas, Efrén Gonzalez-Ramirez and Ricardo Montoya-Zamora

Chapter 27 **Stages of Gene Regulatory Network Inference: the Evolutionary Algorithm Role 521**
Alina Sîrbu, Heather J. Ruskin and Martin Crane

Chapter 28 **Evolutionary Algorithms in Crystal Structure Analysis 547**
Attilio Immirzi, Consiglia Tedesco and Loredana Erra

Chapter 29 **Evolutionary Enhanced Level Set Method for Structural Topology Optimization 565**
Haipeng Jia, Chundong Jiang, Lihui Du, Bo Liu and Chunbo Jiang

Preface

Evolutionary algorithms (EAs) are the population-based metaheuristic optimization algorithms. Candidate solutions to the optimization problem are defined as individuals in a population, and evolution of the population leads to finding better solutions. The fitness of individuals to the environment is estimated and some mechanisms inspired by biological evolution are applied to evolution of the population.

Genetic algorithm (GA), Evolution strategy (ES), Genetic programming (GP), and Evolutionary programming (EP) are very popular Evolutionary algorithms. Genetic Algorithm, which was presented by Holland in 1970s, mainly uses selection, crossover and mutation operators for evolution of the population. Evolutionary Strategy, which was presented by Rechenberg and Schwefel in 1960s, uses natural problem-dependent representations and primarily mutation and selection as operators. Genetic programming and Evolutionary programming are GA- and ES-based methodologies to find computer program or mathematical function that perform user-defined task, respectively.

As related techniques, Ant colony optimization (ACO) and Particle swarm optimization (PSO) are well known. Ant colony optimization (ACO) was presented by Dorigo in 1992 and Particle swarm optimization (PSO) was by Kennedy, Eberhart and Shi in 1995. While Genetic Algorithm and Evolutionary Strategy are inspired from the genetical evolution, Ant colony optimization and Particle swarm optimization are from the behavior of social insects (ants) and bird swarm, respectively. Therefore, Ant colony optimization and Particle swarm optimization are usually classified into the swarm intelligence algorithms.

Evolutionary algorithms are successively applied to wide optimization problems in the engineering, marketing, operations research, and social science, such as include scheduling, genetics, material selection, structural design and so on. Apart from mathematical optimization problems, evolutionary algorithms have also been used as an experimental framework within biological evolution and natural selection in the field of artificial life.

The book consists of 29 chapters. Chapters 1 to 9 describe the algorithms for enhancing the search performance of evolutionary algorithms such as Genetic Algorithm, Swarm Optimization Algorithm and Quantum-inspired Algorithm. Chapter 10 introduces the programming language for evolutionary algorithm. Chapter 11 explains evolutionary algorithms for the fuzzy data problems. Chapters 12 to 13 discuss theoretical analysis of evolutionary algorithms. The remaining chapters describe the applications of the

evolutionary algorithms. In chapters 12 to 17, the evolutionary algorithms are applied to several scheduling problems such as Traveling salesman problem, Job Scheduling problem and so on. Chapters 18 and 24 describe how to use evolutionary algorithm to logic synthesis, network coding, filters, pattern recognition and so on. Chapters 25 to 29 also discuss the other applications of evolutionary algorithms such as random field discretization, biosystem simulation, gene regulatory, crystal structure analysis and structural design.

Eisuke Kita
Graduate School of Information Science
Nagoya University
Japan

Part 1

New Algorithms

1

Hybridization of Evolutionary Algorithms

Iztok Fister, Marjan Mernik and Janez Brest
University of Maribor
Slovenia

1. Introduction

Evolutionary algorithms are a type of general problem solvers that can be applied to many difficult optimization problems. Because of their generality, these algorithms act similarly like Swiss Army knife (Michalewicz & Fogel, 2004) that is a handy set of tools that can be used to address a variety of tasks. In general, a definite task can be performed better with an associated special tool. However, in the absence of this tool, the Swiss Army knife may be more suitable as a substitute. For example, to cut a piece of bread the kitchen knife is more suitable, but when traveling the Swiss Army knife is fine.
Similarly, when a problem to be solved from a domain where the problem-specific knowledge is absent evolutionary algorithms can be successfully applied. Evolutionary algorithms are easy to implement and often provide adequate solutions. An origin of these algorithms is found in the Darwian principles of natural selection (Darwin, 1859). In accordance with these principles, only the fittest individuals can survive in the struggle for existence and reproduce their good characteristics into next generation.
As illustrated in Fig. 1, evolutionary algorithms operate with the population of solutions. At first, the solution needs to be defined within an evolutionary algorithm. Usually, this definition cannot be described in the original problem context directly. In contrast, the solution is defined by data structures that describe the original problem context indirectly and thus, determine the search space within an evolutionary search (optimization process). There exists the analogy in the nature, where the genotype encodes the phenotype, as well. Consequently, a genotype-phenotype mapping determines how the genotypic representation is mapped to the phenotypic property. In other words, the phenotypic property determines the solution in original problem context. Before an evolutionary process actually starts, the initial population needs to be generated. The initial population is generated most often randomly. A basis of an evolutionary algorithm represents an evolutionary search in which the selected solutions undergo an operation of reproduction, i.e., a crossover and a mutation. As a result, new candidate solutions (offsprings) are produced that compete, according to their fitness, with old ones for a place in the next generation. The fitness is evaluated by an evaluation function (also called fitness function) that defines requirements of the optimization (minimization or maximization of the fitness function). In this study, the minimization of the fitness function is considered. As the population evolves solutions becomes fitter and fitter. Finally, the evolutionary search can be iterated until a solution with sufficient quality (fitness) is found or the predefined number of generations is reached (Eiben & Smith, 2003). Note that some steps in Fig. 1 can be omitted (e.g., mutation, survivor selection).

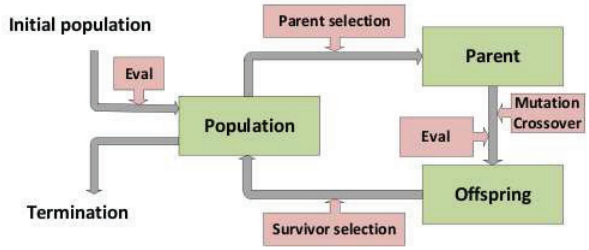


Fig. 1. Scheme of Evolutionary Algorithms

An evolutionary search is categorized by two terms: exploration and exploitation. The former term is connected with a discovering of the new solutions, while the later with a search in the vicinity of knowing good solutions (Eiben & Smith, 2003; Liu et al., 2009). Both terms, however, interweave each other in the evolutionary search. The evolutionary search acts correctly when a sufficient diversity of population is present. The population diversity can be measured differently: the number of different fitness values, the number of different genotypes, the number of different phenotypes, entropy, etc. The higher the population diversity, the better exploration can be expected. Losing of population diversity can lead to the premature convergence.

Exploration and exploitation of evolutionary algorithms are controlled by the control parameters, for instance the population size, the probability of mutation p_m, the probability of crossover p_c, and the tournament size. To avoid a wrong setting of these, the control parameters can be embedded into the genotype of individuals together with problem variables and undergo through evolutionary operations. This idea is exploited by a self-adaptation. The performance of a self-adaptive evolutionary algorithm depends on the characteristics of population distribution that directs the evolutionary search towards appropriate regions of the search space (Meyer-Nieberg & Beyer, 2007). Igel & Toussaint (2003), however, widened the notion of self-adaptation with a generalized concept of self-adaptation. This concept relies on the neutral theory of molecular evolution (Kimura, 1968). Regarding this theory, the most mutations on molecular level are selection neutral and therefore, cannot have any impact on fitness of individual. Consequently, the major part of evolutionary changes are not result of natural selection but result of random genetic drift that acts on neutral allele. An neutral allele is one or more forms of a particular gene that has no impact on fitness of individual (Hamilton, 2009). In contrast to natural selection, the random genetic drift is a whole stochastic process that is caused by sampling error and affects the frequency of mutated allele. On basis of this theory Igel and Toussaint ascertain that the neutral genotype-phenotype mapping is not injective. That is, more genotypes can be mapped into the same phenotype. By self-adaptation, a neutral part of genotype (problem variables) that determines the phenotype enables discovering the search space independent of the phenotypic variations. On the other hand, the rest part of genotype (control parameters) determines the strategy of discovering the search space and therefore, influences the exploration distribution.

Although evolutionary algorithms can be applied to many real-world optimization problems their performance is still subject of the No Free Lunch (NFL) theorem (Wolpert & Macready, 1997). According to this theorem any two algorithms are equivalent, when their performance is compared across all possible problems. Fortunately, the NFL theorem can be circumvented

for a given problem by a hybridization that incorporates the problem specific knowledge into evolutionary algorithms.

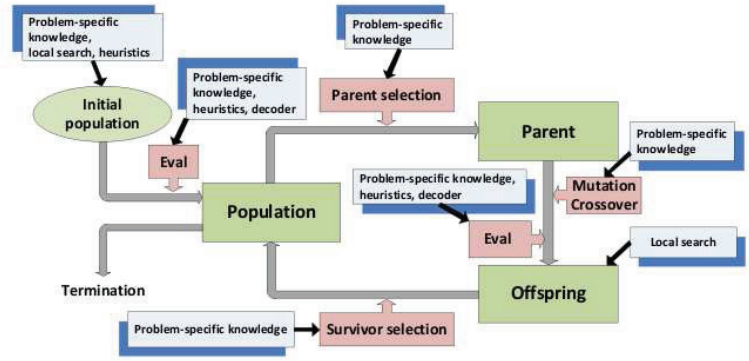


Fig. 2. Hybridization of Evolutionary Algorithms

In Fig. 2 some possibilities to hybridize evolutionary algorithms are illustrated. At first, the initial population can be generated by incorporating solutions of existing algorithms or by using heuristics, local search, etc. In addition, the local search can be applied to the population of offsprings. Actually, the evolutionary algorithm hybridized with local search is called a memetic algorithm as well (Moscato, 1999; Wilfried, 2010). Evolutionary operators (mutation, crossover, parent and survivor selection) can incorporate problem-specific knowledge or apply the operators from other algorithms. Finally, a fitness function offers the most possibilities for a hybridization because it can be used as decoder that decodes the indirect represented genotype into feasible solution. By this mapping, however, the problem specific knowledge or known heuristics can be incorporated to the problem solver.
In this chapter the hybrid self-adaptive evolutionary algorithm (HSA-EA) is presented that is hybridized with:

- construction heuristic,
- local search,
- neutral survivor selection, and
- heuristic initialization procedure.

This algorithm acts as meta-heuristic, where the down-level evolutionary algorithm is used as generator of new solutions, while for the upper-level construction of the solutions a traditional heuristic is applied. This construction heuristic represents the hybridization of evaluation function. Each generated solution is improved by the local search heuristics. This evolutionary algorithm supports an existence of neutral solutions, i.e., solutions with equal values of a fitness function but different genotype representation. Such solutions can be arisen often in matured generations of evolutionary process and are subject of neutral survivor selection. This selection operator models oneself upon a neutral theory of molecular evolution (Kimura, 1968) and tries to direct the evolutionary search to new, undiscovered regions of search space. In fact, the neutral survivor selection represents hybridization of evolutionary operators, in this case, the survivor selection operator. The hybrid self-adaptive evolutionary algorithm can be used especially for solving of the hardest combinatorial optimization problems (Fister et al., 2010).

The chapter is further organized as follows. In the Sect. 2 the self-adaptation in evolutionary algorithms is discussed. There, the connection between neutrality and self-adaptation is explained. Sect. 3 describes hybridization elements of the self-adaptive evolutionary algorithm. Sect. 4 introduces the implementations of hybrid self-adaptive evolutionary algorithm for graph 3-coloring in details. Performances of this algorithm are substantiated with extensive collection of results. The chapter is concluded with summarization of the performed work and announcement of the possibilities for the further work.

2. The self-adaptive evolutionary algorithms

Optimization is a dynamical process, therefore, the values of parameters that are set at initialization become worse during the run. The necessity to adapt control parameters during the runs of evolutionary algorithms born an idea of self-adaptation (Holland, 1992), where some control parameters are embedded into genotype. This genotype undergoes effects of variation operators. Mostly, with the notion of self-adaptation Evolutionary Strategies (Beyer, 1998; Rechenberg, 1973; Schwefel, 1977) are connected that are used for solving continuous optimization problems. Typically, the problem variables in Evolutionary Strategies are represented as real-coded vector $y = (y_1, \ldots, y_n)$ that are embedded into genotype together with control parameters (mostly mutation parameters). These parameters determine mutation strengths σ that must be greater than zero. Usually, the mutation strengths are assigned to each problem variable. In that case, the uncorrelated mutation with n step sizes is obtained (Eiben & Smith, 2003). Here, the candidate solution is represented as $(y_1, \ldots, y_n, \sigma_1, \ldots, \sigma_n)$. The mutation is now specified as follows:

$$\sigma_i^{'} = \sigma_i \cdot \exp(\tau^{'} \cdot \mathrm{N}(0,1) + \tau \cdot \mathrm{N}_i(0,1)), \tag{1}$$

$$y_i^{'} = y_i + \sigma_i^{'} \cdot \mathrm{N}_i(0,1), \tag{2}$$

where $\tau^{'} \propto 1/\sqrt{2 \cdot n}$ and $\tau \propto 1/\sqrt{2 \cdot \sqrt{n}}$ denote the learning rates. To keep the mutation strengths σ_i greater than zero, the following rule is used

$$\sigma_i < \varepsilon_0 \Rightarrow \sigma_i = \varepsilon_0. \tag{3}$$

Frequently, a crossover operator is used in the self-adaptive Evolutionary Strategies. This operator from two parents forms one offsprings. Typically, a discrete and arithmetic crossover is used. The former, from among the values of two parents x_i and y_i that are located on i-th position, selects the value of offspring z_i randomly. The later calculates the value of offspring z_i from the values of two parents x_i and y_i that are located on i-th position according to the following equation:

$$z_i = \alpha \cdot x_i + (1 - \alpha) \cdot y_i, \tag{4}$$

where parameter α captures the values from interval $\alpha \in [0 \ldots 1]$. In the case of $\alpha = 1/2$, the uniform arithmetic crossover is obtained.

The potential benefits of neutrality was subject of many researches in the biological science (Conrad, 1990; Hynen, 1996; Kimura, 1968). At the same time, the growing interest for the usage of this knowledge in evolutionary computation was raised (Barnett, 1998; Ebner et al., 2001). Toussaint & Igel (2002) dealt with the non-injectivity of genotype-phenotype mapping that is the main characteristic of this mapping. That is, more genotypes can be mapped to the same phenotype. Igel & Toussaint (2003) pointed out that in the absence of an external control and with a constant genotype-phenotype mapping only neutral genetic variations can allow

an adaptation of exploration distribution without changing the phenotypes in the population. However, the neutral genetic variations act on the genotype of parent but does not influence on the phenotype of offspring.
As a result, control parameters in evolutionary strategies represent a search strategy. The change of this strategy enables a discovery of new regions of the search space. The genotype, therefore, does not include only the information addressing its phenotype but the information about further discovering of the search space as well. In summary, the neutrality is not necessary redundant but it is prerequisite for self-adaptation. This concept is called the general concept of self-adaptation as well (Meyer-Nieberg & Beyer, 2007).

3. How to hybridize the self-adaptive evolutionary algorithms

Evolutionary algorithms are a generic tool that can be used for solving many hard optimization problems. However, the solving of that problems showed that evolutionary algorithms are too problem-independent. Therefore, there are hybridized with several techniques and heuristics that are capable to incorporate problem-specific knowledge. Grosan & Abraham (2007) identified mostly used hybrid architectures today as follows:

- hybridization between two evolutionary algorithms (Grefenstette, 1986),
- neural network assisted evolutionary algorithm (Wang, 2005),
- fuzzy logic assisted evolutionary algorithm (Herrera & Lozano, 1996; Lee & Takagi, 1993),
- particle swarm optimization assisted evolutionary algorithm (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995),
- ant colony optimization assisted evolutionary algorithm (Fleurent & Ferland, 1994; Tseng & Liang, 2005),
- bacterial foraging optimization assisted evolutionary algorithm (Kim & Cho, 2005; Neppalli & Chen, 1996),
- hybridization between an evolutionary algorithm and other heuristics, like local search (Moscato, 1999), tabu search (Galinier & Hao, 1999), simulated annealing (Ganesh & Punniyamoorthy, 2004), hill climbing (Koza et al., 2003), dynamic programming (Doerr et al., 2009), etc.

In general, successfully implementation of evolutionary algorithms for solving a given problem depends on incorporated problem-specific knowledge. As already mentioned before, all elements of evolutionary algorithms can be hybridized. Mostly, a hybridization addresses the following elements of evolutionary algorithms (Michalewicz, 1992):

- initial population,
- genotype-phenotype mapping,
- evaluation function, and
- variation and selection operators.

First, problem-specific knowledge incorporated into heuristic procedures can be used for creating an initial population. Second, genotype-phenotype mapping is used by evolutionary algorithms, where the solutions are represented in an indirect way. In that cases, a constructing algorithm that maps the genotype representation into a corresponding phenotypic solution needs to be applied. This constructor can incorporate various heuristic or other problem-specific knowledge. Third, to improve the current solutions by an evaluation

Algorithm 1 The construction heuristic. I: task, S: solution.

1: **while NOT** $final\_solution(y \in S)$ **do**
2: $\quad add\_element\_to\_solution\_heuristicaly(y_i \in I, S)$;
3: **end while**

function, local search heuristics can be used. Finally, problem-specific knowledge can be exploited by heuristic variation and selection operators.
The mentioned hybridizations can be used to hybridize the self-adaptive evolutionary algorithms as well. In the rest of chapter, we propose three kinds of hybridizations that was employed to the proposed hybrid self-adaptive evolutionary algorithms:

- the construction heuristics that can be used by the genotype-phenotype mapping,
- the local search heuristics that can be used by the evaluation function, and
- the neutral survivor selection that incorporates the problem-specific knowledge.

Because the initialization of initial population is problem dependent we omit it from our discussion.

3.1 The construction heuristics

Usually, evolutionary algorithms are used for problem solving, where a lot of experience and knowledge is accumulated in various heuristic algorithms. Typically, these algorithms work well on limited number of problems (Hoos & Stützle, 2005). On the other hand, evolutionary algorithms are a general method suitable to solve very different kinds of problems. In general, these algorithms are less efficient than heuristics specialized to solve the given problem. If we want to combine a benefit of both kind of algorithms then the evolutionary algorithm can be used for discovering new solutions that the heuristic exploits for building of new, probably better solutions. Construction heuristics build the solution of optimization problem incrementally, i.e., elements are added to a solution step by step (Algorithm 1).

3.2 The local search

A local search belongs to a class of improvement heuristics (Aarts & Lenstra, 1997). In our case, main characteristic of these is that the current solution is taken and improved as long as improvements are perceived.
The local search is an iterative process of discovering points in the vicinity of current solution. If a better solution is found the current solution is replaced by it. A neighborhood of the current solution y is defined as a set of solutions that can be reached using an unary operator $\mathcal{N} : S \rightarrow 2^S$ (Hoos & Stützle, 2005). In fact, each neighbor y' in neighborhood $\mathcal{N}$ can be reached from current solution y in k strokes. Therefore, this neighborhood is called $k-opt$ neighborhood of current solution y as well. For example, let the binary represented solution y and 1-*opt* operator on it are given. In that case, each of neighbors $\mathcal{N}(y)$ can be reached changing exactly one bit. The neighborhood of this operator is defined as

$$\mathcal{N}_{1\text{-}opt}(y) = \{y' \in S | d_H(y, y') = 1\}, \tag{5}$$

where d_H denotes a Hamming distance of two binary vectors as follows

$$d_H(y, y') = \sum_{i=1}^{n} (y_i \oplus y'_i), \tag{6}$$

Algorithm 2 The local search. I: task, S: solution.

1: $generate\_initial\_solution(y \in S)$;
2: **repeat**
3: $find\_next\_neighbor(y' \in \mathcal{N}(y))$;
4: **if** $(f(y') < f(y))$ **then**
5: $y = y'$;
6: **end if**
7: **until** $set\_of\_neighbor\_empty$;

where operator $\oplus$ means *exclusive or* operation. Essentially, the Hamming distance in Equation 6 is calculated by counting the number of different bits between vectors y and y'. The 1-opt operator defines the set of feasible 1-opt strokes while the number of feasible 1-opt strokes determines the size of neighborhood.
As illustrated by Algorithm 2, the local search can be described as follows (Michalewicz & Fogel, 2004):

- The initial solution is generated that becomes the current solution (procedure $generate\_initial\_solution$).
- The current solution is transformed with $k - opt$ strokes and the given solution y' is evaluated (procedure $find\_next\_neighbor$).
- If the new solution y' is better than the current y the current solution is replaced. On the other hand, the current solution is kept.
- Lines 2 to 7 are repeated until the set of neighbors is not empty (procedure $set\_of\_neighbor\_empty$).

In summary, the $k - opt$ operator represents a basic element of the local search from which depends how exhaustive the neighborhood will be discovered. Therefore, the problem-specific knowledge needs to be incorporated by building of the efficient operator.

3.3 The neutral survivor selection

A genotype diversity is one of main prerequisites for the efficient self-adaptation. The smaller genotypic diversity causes that the population is crowded in the search space. As a result, the search space is exploited. On the other hand, the larger genotypic diversity causes that the population is more distributed within the search space and therefore, the search space is explored (Bäck, 1996). Explicitly, the genotype diversity of population is maintained with a proposed neutral survivor selection that is inspired by the neutral theory of molecular evolution (Kimura, 1968), where the neutral mutation determines to the individual three possible destinies, as follows:

- the fittest individual can survive in the struggle for existence,
- the less fitter individual is eliminated by the natural selection,
- individual with the same fitness undergo an operation of genetic drift, where its survivor is dependent on a chance.

Each candidate solution represents a point in the search space. If the fitness value is assigned to each feasible solution then these form a fitness landscape that consists of peeks, valleys and plateaus (Wright, 1932). In fact, the peaks in the fitness landscape represents points with higher fitness, the valleys points with the lower fitness while plateaus denotes regions,

where the solutions are neutral (Stadler, 1995). The concept of the fitness landscape plays an important role in evolutionary computation as well. Moreover, with its help behavior of evolutionary algorithms by solving the optimization problem can be understood. If on the search space we look from a standpoint of fitness landscape then the heuristical algorithm tries to navigate through this landscape with aim to discover the highest peeks in the landscape (Merz & Freisleben, 1999).
However, to determine how distant one solution is from the other, some measure is needed. Which measure to use depends on a given problem. In the case of genetic algorithms, where we deal with the binary solutions, the Hamming distance (Equation 6) can be used. When the solutions are represented as real-coded vectors an Euclidian distance is more appropriate. The Euclidian distance between two vectors x and y is expressed as follows:

$$d_E(x,y) = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - y_i)^2}, \tag{7}$$

and measures the root of quadrat differences between elements of vectors x and y. The main characteristics of fitness landscapes that have a great impact on the evolutionary search are the following (Merz & Freisleben, 1999):

- the fitness differences between neighboring points in the fitness landscape: to determine a ruggedness of the landscape, i.e., more rugged as the landscape, more difficultly the optimal solution can be found;
- the number of peaks (local optima) in the landscape: the higher the number of peaks, the more difficulty the evolutionary algorithms can direct the search to the optimal solution;
- how the local optima are distributed in the search space: to determine the distribution of the peeks in the fitness landscape;
- how the topology of the basins of attraction influences on the exit from the local optima: to determine how difficult the evolutionary search that gets stuck into local optima can find the exit from it and continue with the discovering of the search space;
- existence of the neutral networks: the solutions with the equal value of fitness represent a plateaus in the fitness landscape.

When the stochastic fitness function is used for evaluation of individuals the fitness landscape is changed over time. In this way, the dynamic landscape is obtained, where the concept of fitness landscape can be applied, first of all, to analyze the neutral networks that arise, typically, in the matured generations. To determine, how the solutions are dissipated over the search space some reference point is needed. For this reason, the current best solution y^* in the population is used. This is added to the population of μ solutions.
An operation of the neutral survivor selection is divided into two phases. In the first phase, the evolutionary algorithm from the population of λ offsprings finds a set of neutral solutions $N_S = \{y_1, \ldots, y_k\}$ that represents the best solutions in the population of offsprings. If the neutral solutions are better than or equal to the reference, i.e. $f(y_i) \leq f(y^*)$ for $i = 1, \ldots, k$, then reference solution y^* is replaced with the neutral solution $y_i \in N_S$ that is the most faraway from reference solution according to the Equation 7. Thereby, it is expected that the evolutionary search is directed to the new, undiscovered region of the search space. In the second phase, the updated reference solution y^* is used to determine the next population of

survivors. Therefore, all offsprings are ordered with regard to the ordering relation $\prec$ (read: is better than) as follows:

$$f(y_1) \prec \ldots \prec f(y_i) \prec f(y_{i+1}) \prec \ldots \prec f(y_\lambda), \tag{8}$$

where the ordering relation $\prec$ is defined as

$$f(y_i) \prec f(y_{i+1}) \Rightarrow \begin{cases} f(y_i) < f(y_{i+1}), \\ f(y_i) = f(y_{i+1}) \wedge (d(y_i, y^*) > d(y_{i+1}, y^*)). \end{cases} \tag{9}$$

Finally, for the next generation the evolutionary algorithm selects the best μ offsprings according to the Equation 8. These individuals capture the random positions in the next generation. Likewise the neutral theory of molecular evolution, the neutral survivor selection offers to the offsprings three possible outcomes, as follows. The best offsprings survive. Additionally, the offspring from the set of neutral solutions that is far away of reference solution can become the new reference solution. The less fitter offsprings are usually eliminated from the population. All other solutions, that can be neutral as well, can survive if they are ordered on the first μ positions regarding to Equation 8.

4. The hybrid self-adaptive evolutionary algorithms in practice

In this section an implementation of the hybrid self-adaptive evolutionary algorithms (HSA-EA) for solving combinatorial optimization problems is represented. The implementation of this algorithm in practice consists of the following phases:

- finding the best heuristic that solves the problem on a traditional way and adapting it to use by the self-adaptive evolutionary algorithm,
- defining the other elements of the self-adaptive evolutionary algorithm,
- defining the suitable local search heuristics, and
- including the neutral survivor selection.

The main idea behind use of the construction heuristics in the HSA-EA is to exploit the knowledge accumulated in existing heuristics. Moreover, this knowledge is embedded into the evolutionary algorithm that is capable to discover the new solutions. To work simultaneously both algorithms need to operate with the same representation of solutions. If this is not a case a decoder can be used. The solutions are encoded by the evolutionary algorithm as the real-coded vectors and decoded before the construction of solutions. The whole task is performed in genotype-phenotype mapping that is illustrated in Fig. 3.
The genotype-phenotype mapping consists of two phases as follows:

- decoding,
- constructing.

Evolutionary algorithms operate in genotypic search space, where each genotype consists of real-coded problem variables and control parameters. For encoded solution only the problem variables are taken. This solution is further decoded by decoder into a decoded solution that is appropriate for handling of a construction heuristic. Finally, the construction heuristic constructs the solution within the original problem context, i.e., problem solution space. This solution is evaluated by the suitable evaluation function.
The other elements of self-adaptive evolutionary algorithm consists of:

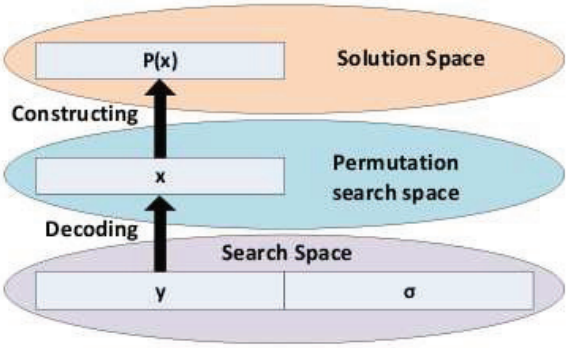


Fig. 3. The genotype-phenotype mapping by hybrid self-adaptive evolutionary algorithm

Algorithm 3 Hybrid Self-Adaptive Evolutionary Algorithm.

1: $t = 0$;
2: $Q^{(0)} = initialization\_procedure()$;
3: $P^{(0)} = evaluate\_and\_improve(Q^{(0)})$;
4: **while not** termination_condition **do**
5: $P' = select\_parent(P^{(t)})$;
6: $P'' = mutate\_and\_crossover(P')$;
7: $P''' = evaluate\_and\_improve(P'')$;
8: $P^{(t+1)} = select\_survivor(P''')$;
9: $t = t + 1$;
10: **end while**

- evaluation function,
- population model,
- parent selection mechanism,
- variation operators (mutation and crossover), and
- initialization procedure and termination condition.

The evaluation function depends on a given problem. The self-adaptive evolutionary algorithm uses the population model (μ, λ), where the λ offsprings is generated from the μ parents. However, the parents that are selected with tournament selection (Eiben & Smith, 2003) are replaced by the μ the best offsprings according to the appropriate population model. The ratio $\lambda/\mu \approx 7$ is used for the efficient self-adaptation (Eiben & Smith, 2003). Typically, the normal uncorrelated mutation with n step sizes, discrete and arithmetic crossover are used by the HSA-EA. Normally, the probabilities of mutation and crossover are set according to the given problem. Selection of the suitable local search heuristics that improve the current solution is a crucial for the performance of the HSA-EA. On the other hand, the implementation of neutral survivor selection is straightforward. Finally, the scheme of the HSA-EA is represented in the Algorithm 3.

In the rest of the chapter we present the implementation of the HSA-EA for the graph 3-coloring. This algorithm is hybridized with the DSatur (Brelaz, 1979) construction heuristic that is well-known traditional heuristic for the graph 3-coloring.

4.1 Graph 3-coloring

Graph 3-coloring can be informally defined as follows. Let assume, an undirected graph $G = (V, E)$ is given, where V denotes a finite set of vertices and E a finite set of unordered pairs of vertices named edges (Murty & Bondy, 2008). The vertices of graph G have to be colored with three colors such that no one of vertices connected with an edge is not colored with the same color.

Graph 3-coloring can be formalized as constraint satisfaction problem (CSP) that is denoted as a pair $\langle S, \phi \rangle$, where S denotes a free search space and ϕ a Boolean function on S. The free search space denotes the domain of candidate solutions $x \in S$ and does not contain any constraints, i.e., each candidate solution is feasible. The function ϕ divides the search space S into feasible and unfeasible regions. The solution of constraint satisfaction problem is found when all constraints are satisfied, i.e., when $\phi(x) = true$.

However, for the 3-coloring of graph $G = (V, E)$ the free search space S consists of all permutations of vertices $v_i \in V$ for $i = 1 \dots n$. On the other hand, the function ϕ (also feasibility condition) is composed of constraints on vertices. That is, for each vertex $v_i \in V$ the corresponding constraint C^{v_i} is defined as the set of constraints involving vertex v_i, i.e., edges $(v_i, v_j) \in E$ for $j = 1 \dots m$ connecting to vertex v_i. The feasibility condition is expressed as conjunction of all constraints $\phi(x) = \wedge_{v_i \in V} C^{v_i}(x)$.

Direct constraint handling in evolutionary algorithms is not straightforward. To overcome this problem, the constraint satisfaction problems are, typically, transformed into unconstrained (also free optimization problem) by the sense of a penalty function. The more the infeasible solution is far away from feasible region, the higher is the penalty. Moreover, this penalty function can act as an evaluation function by the evolutionary algorithm. For graph 3-coloring it can be expressed as

$$f(x) = \sum_{i=0}^{n} \psi(x, C^{v_i}), \tag{10}$$

where the function $\psi(x, C^{v_i})$ is defined as

$$\psi(x, C^{v_i}) = \begin{cases} 1 \text{ if } x \text{ violates at least one } c_j \in C^{v_i}, \\ 0 \qquad\qquad \text{otherwise.} \end{cases} \tag{11}$$

Note that all constraints in solution $x \in S$ are satisfied, i.e., $\phi(x) = true$ if and only if $f(x) = 0$. In this way, the Equation 10 represents the feasibility condition and can be used to estimate the quality of solution $x \in S$ in the permutation search space. The permutation x determines the order in which the vertices need to be colored. The size of the search space is huge, i.e., $n!$. As can be seen from Equation 10, the evaluation function depends on the number of constraint violations, i.e., the number of uncolored vertices. This fact causes that more solutions can have the same value of the evaluation function. Consequently, the large neutral networks can arise (Stadler, 1995). However, the neutral solutions are avoided if the slightly modified evaluation function is applied, as follows:

$$f(x) = \sum_{i=0}^{n} w_i \times \psi(x, C^{v_i}), \quad w_i \neq 0, \tag{12}$$

where w_i represents the weight. Higher than the value of weights harder the appropriate vertex is to color.

4.1.1 The hybrid self-adaptive evolutionary algorithm for graph 3-coloring

The hybrid self-adaptive evolutionary algorithm is hybridized with the DSatur (Brelaz, 1979) construction heuristic and the local search heuristics. In addition, the problem specific knowledge is incorporated by the initialization procedure and the neutral survivor selection. In this section we concentrate, especially, on a description of those elements in evolutionary algorithm that incorporate the problem specific knowledge. That are:

- the initialization procedure,
- the genotype-phenotype mapping,
- local search heuristics and
- the neutral survivor selection.

The other elements of this evolutionary algorithm, as well as neutral survivor selection, are common and therefore, discussed earlier in the chapter.

The Initialization Procedure

Initially, original DSatur algorithm orders the vertices $v_i \in V$ for $i = 1 \dots n$ of a given graph G descendingly according to the vertex degrees denoted by $d_G(v_i)$ that counts the number of edges that are incident with the vertex v_i (Murty & Bondy, 2008). To simulate behavior of the original DSatur algorithm (Brelaz, 1979), the first solution in the population is initialized as follows:

$$y_i^{(0)} = \frac{d_G(v_i)}{\max_{i=1\dots n} d_G(v_i)}, \quad \text{for } i = 1 \dots n. \tag{13}$$

Because the genotype representation is mapped into a permutation of weights by decoder the same ordering as by original DSatur is obtained, where the solution can be found in the first step. However, the other $\mu - 1$ solutions in the population are initialized randomly.

The Genotype-phenotype mapping

As illustrated in Fig. 3, the solution is represented in genotype search space as tuple $\langle y_1, \dots, y_n, \sigma_1, \dots, \sigma_n \rangle$, where problem variables y_i for $i = 1 \dots n$ denote how hard the given vertex is to color and control parameters σ_i for $i = 1 \dots n$ mutation steps of uncorrelated mutation. A decoder decodes the problem variables into permutation of vertices and corresponding weights. However, all feasible permutation of vertices form the permutation search space. The solution in this search space is represented as tuple $\langle v_1, \dots, v_n, w_1, \dots, w_n \rangle$, where variables v_i for $i = 1 \dots n$ denote the permutation of vertices and variables w_i corresponding weights. The vertices are ordered into permutation so that vertex v_i is predecessor of vertex v_{i+1} if and only if $w_i \geq w_{i+1}$. Values of weights w_i are obtained by assigning the corresponding values of problem variables, i.e. $w_i = y_i$ for $i = 1 \dots n$. Finally, DSatur construction heuristic maps the permutation of vertices and corresponding weights into phenotypic solution space that consists of all possible 3-colorings c_i. Note that the size of this space is 3^n. DSatur construction heuristic acts like original DSatur algorithm (Brelaz, 1979), i.e. it takes the permutation of vertices and color these as follows:

- the heuristic selects a vertex with the highest saturation, and colors it with the lowest of the three colors;
- in the case of a tie, the heuristic selects a vertex with the maximal weight;
- in the case of a tie, the heuristic selects a vertex randomly.

Algorithm 4 Evaluate and improve. y: solution.

```
1: est = evaluate(y);
2: repeat
3:    climbing = FALSE;
4:    y' = k_move(y);
5:    ls_est = evaluate(y');
6:    if ls_est < est then
7:       y = y';
8:       est = ls_est;
9:       climbing = TRUE;
10:   end if
11: until climbing = TRUE
```

The main difference between this heuristic and the original DSatur algorithm is in the second step where the heuristic selects the vertices according to the weights instead of degrees.

Local Search Heuristics
The current solution is improved by a sense of local search heuristics. At each evaluation of solution the best neighbor is obtained by acting of the following original local search heuristics:

- inverse,
- ordering by saturation,
- ordering by weights, and
- swap.

The evaluation of solution is presented in Algorithm 4 from which it can be seen that the local search procedure ($k\_move(y)$) is iterated until improvements are perceived. However, this procedure implements all four mentioned local search heuristics. The best neighbor is generated from the current solution by local search heuristics with k-exchanging of vertices. In the case, the best neighbor is better than the current solution the later is replaced by the former.
In the rest of the subsection, an operation of the local search heuristics is illustrated in Fig. 4-7 by samples, where a graph with nine vertices is presented. The graph is composed of a permutation of vertices v, corresponding coloring c, weights w and saturation degrees d.

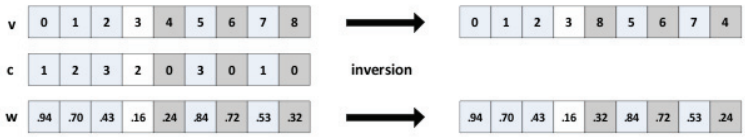


Fig. 4. Inverse local search heuristic

The inverse local search heuristic finds all uncolored vertices in a solution and inverts their order. As can be shown in Fig. 4, the uncolored vertices 4, 6 and 8 are shadowed. The best neighbor is obtained by inverting of their order as is presented on right-hand side of this figure. The number of vertex exchanged is dependent of the number of uncolored vertices ($k - opt$ neighborhood).

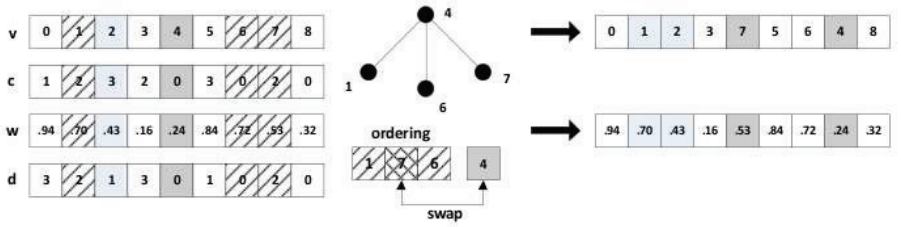


Fig. 5. Ordering by saturation local search heuristic

The ordering by saturation local search heuristic acts as follows. The first uncolored vertex is taken at the first. To this vertex a set of adjacent vertices are selected. Then, these vertices are ordered descending with regard to the values of saturation degree. Finally, the adjacent vertex with the highest value of saturation degree in the set of adjacent vertices is swapped with the uncolored vertex. Here, the simple 1-*opt* neighborhood of current solution is defined by this local search heuristic. In the example on Fig. 5 the first uncolored vertex 4 is shadowed, while its adjacent vertices 1, 6 and 7 are hatched. However, the vertices 1 and 7 have the same saturation degree, therefore, the vertex 7 is selected randomly. Finally, the vertices 4 in 7 are swapped (right-hand side of Fig. 5).

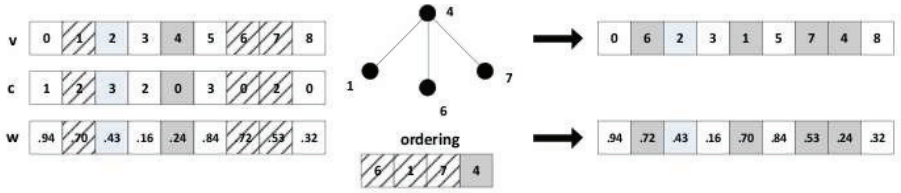


Fig. 6. Ordering by weights local search heuristic

When ordering of weights, the local search heuristic takes the first uncolored vertex and determines a set of adjacent vertices including it. This set of vertices is then ordered descending with regard to the values of weights. This local search heuristic determines the $k - opt$ neighborhood of current solution, where k is dependent of a degree of the first uncolored vertex. As illustrated by Fig. 6, the uncolored vertex 4 is shadowed, while its adjacent vertices 1, 6 and 7 are hatched. The appropriate ordering of the selected set of vertices is shown in the right-hand of Fig. 6 after the operation of the local search heuristic.

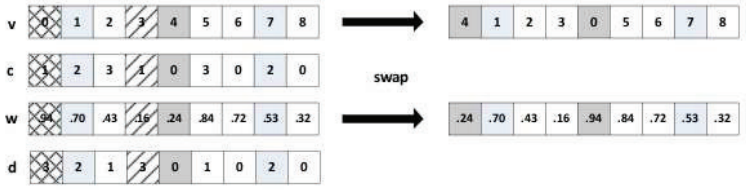


Fig. 7. Swap local search heuristic

The swap local search heuristic finds the first uncolored vertex and descendingly orders the set of all predecessors in the solution according to the saturation degree. Then, the uncolored

vertex is swapped with the vertex from the set of predecessors with the highest saturation degree. When more vertices with the same highest saturation degree are arisen, the subset of these vertices is determined. The vertex from this subset is then selected randomly. Therefore, the best neighbor of the current solution is determined by an exchange of two vertices (1-opt neighborhood). As illustrated in Fig. 7, the first uncolored vertex 4 is shadowed, while the vertices 0 and 4 that represent the subset of vertices with the highest saturation are hatched. In fact, the vertex 0 is selected randomly and the vertices 0 and 4 are swapped as is presented in right-hand of Fig. 7.

4.1.2 Analysis of the hybrid self-adaptive evolutionary algorithm for graph 3-coloring

The goal of this subsection is twofold. At the first, an influence of the local search heuristics on results of the HSA-EA is analyzed in details. Further, a comparison of the HSA-EA hybridized with the neutral survivor selection and the HSA-EA with the deterministic selection is made. In this context, the impact of the heuristic initialization procedure are taken into consideration as well.

Characteristics of the HSA-EA used in experiments were as follows. The normal distributed mutation was employed and applied with mutation probability of 1.0. The crossover was not used. The tournament selection with size 3 selects the parents for mutation. The population model $(15, 100)$ was suitable for the self-adaptation because the ratio between parents and generated offspring amounted to $100/15 \approx 7$ as recommended by Bäck (1996). As termination condition, the maximum number of evaluations to solution was used. Fortunately, the average number of evaluations to solution (AES) that counts the number of evaluation function calls was employed as the performance measure of efficiency. In addition, the average number of uncolored nodes (AUN) was employed as the performance measure of solution quality. This measure was applied when the HSA-EA does not find the solution and counts the number of uncolored vertices. Nevertheless, the success rate (SR) was defined as the primary performance measure and expressed as the ratio between the runs in which the solution was found and all performed runs.

The Culberson (2008) random graph generator was employed for generation of random graphs that constituted the test suite. It is capable to generate the graphs of various types, number of vertices, edge densities and seeds of random generator. In this study we concentrated on the equi-partite type of graphs. This type of graphs is not the most difficult to color but difficult enough for many existing algorithms (Culberson & Luo, 2006). The random graph generator divides the vertices of graph into three color sets before generating randomly. In sense of equi-partite random graph, these color sets are as close in size as possible.

All generated graphs consisted of $n = 1,000$ vertices. An edge density is controlled by parameter p of the random graph generator that determines probability that two vertices v_i and v_j in the graph G are connected with an edge (v_i, v_j) (Chiarandini & Stützle, 2010). However, if p is small the graph is not connected because the edges are sparse. When p is increased the number of edges raised and the graph becomes interconnected. As a result, the number of constraints that needs to be satisfied by the coloring algorithm increases until suddenly the graph becomes uncolorable. This occurrence depends on a ratio between the number of edges and the number of vertices. The ratio is referred to as the threshold (Hayes, 2003). That is, in the vicinity of the threshold the vertices of the random generated graph becomes hard to color or even the graph becomes uncolorable. Fortunately, the graph instances with this ratio much higher that the threshold are easy to color because these graphs are densely interconnected. Therefore, many global optima exist in the search space that can

be discovered easy by many graph 3-coloring algorithms. Interestingly, for random generated graphs the threshold arises near to the value 2.35 (Hayes, 2003). For example, the equi-partite graph generated with number of vertices $1,000$ and the edge density determined by $p = 0.007$ consists of 2,366 edges. Because the ratio $2,366/1,000 = 2.37$ is near to the threshold, we can suppose that this instance of graph is hard to color. The seed s of random graph generator determines which of the two vertices v_i and v_j are randomly drawn from different 3-color sets to form an edge (v_i, v_j) but it does not affect the performance of the graph 3-coloring algorithm (Eiben et al., 1998). In this study, the instances of random graphs with seed $s = 5$ were employed.

To capture a phenomenon of the threshold, the parameter p by generation of the equi-partite graphs was varied from $p = 0.005$ to $p = 0.012$ in a step of 0.0005. In this way, the test suite consisted of 15 instances of graphs, in which the hardest graph with $p = 0.007$ was presented as well. In fact, the evolutionary algorithm was applied to each instance 25 times and the average results of these runs were considered.

The impact of the local search heuristics

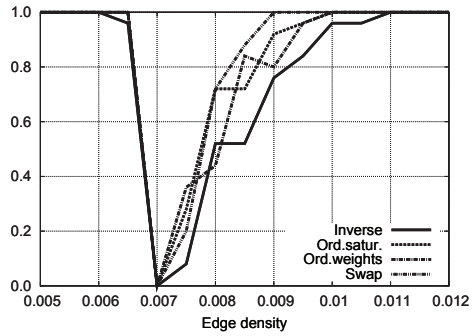
In this experiments, the impact of four implemented local search heuristics on results of the HSA-EA was taken into consideration. Results of the experiments are illustrated in the Fig. 8 that is divided into six graphs and arranged according to the particular measures SR, AES and AUN. The graphs on the left side of the figure, i.e. 8.a, 8.c and 8.e, represent a behavior of the HSA-EA hybridized with four different local search heuristics. This kind of the HSA-EA is referred to as original HSA-EA in the rest of chapter.

A seen by the Fig. 8.a, no one of the HSA-EA versions was succeed to solve the hardest instance of graph with $p = 0.007$. The best results in the vicinity of the threshold is observed by the HSA-EA hybridizing with the ordering by saturation local search heuristic ($SR = 0.36$ by $p = 0.0075$). The overall best performance is shown by the HSA-EA using the swap local search heuristic. Although the results of this algorithm is not the best by instances the nearest to the threshold ($SR = 0.2$ by $p = 0.0075$), this local search heuristic outperforms the other by solving the remaining instances in the collection.

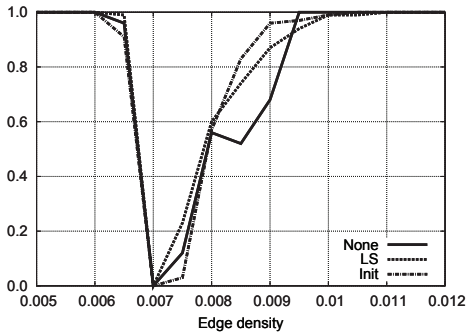
In average, results according to the AES (Fig. 8.c) show that the HSA-EA hybridized with the swap local search heuristic finds the solutions with the smallest number of the fitness evaluations. However, troubles are arisen in the vicinity of the threshold, where the HSA-EA with other local search heuristics are faced with the difficulties as well. Moreover, at the threshold the HSA-EA hybridizing with all the used local search heuristics reaches the limit of 300,000 allowed function evaluations.

The HSA-EA hybridizing with the ordering by saturation local search heuristic demonstrates the worst results according to the AUN, as presented in the Fig. 8.e. The graph instance by $p = 0.0095$ was exposed as the most critical by this algorithm ($AUN = 50$) although this is not the closest to the threshold. In average, when the HSA-EA was hybridized with the other local search heuristics than the ordering by saturation, all instances in the collection were solved with less than 20 uncolored vertices.

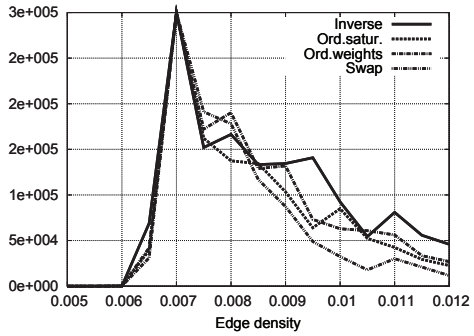
In the right side of the Fig. 8, results of different versions of the HSA-EA are collected. The first version that is designated as *None* operates with the same parameters as the original HSA-EA but without the local search heuristics. The label LS in this figure indicates the original version of the HSA-EA. Finally, the label *Init* denotes the original version of the HSA-EA with the exception of initialization procedure. While all considered versions of the HSA-EA uses the heuristic initialization procedure this version of the algorithm employs the



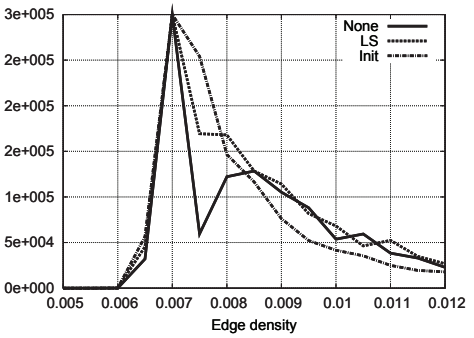
(a) Success rate (SR)



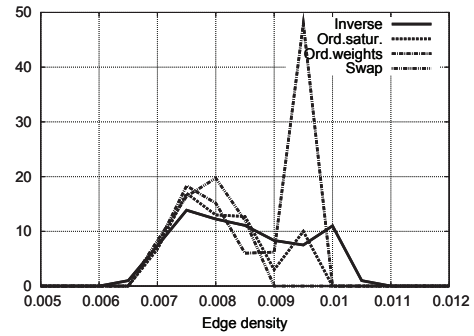
(b) Success rate (SR)



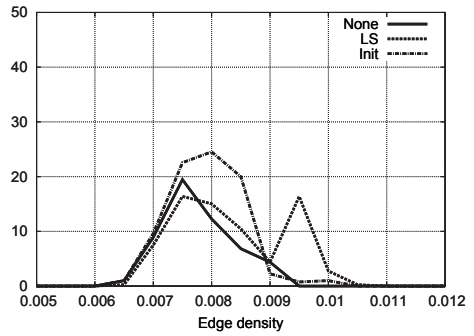
(c) Average evaluations to solution (AES)



(d) Average evaluations to solution (AES)



(e) Average number of uncolored nodes (AUN)



(f) Average number of uncolored nodes (AUN)

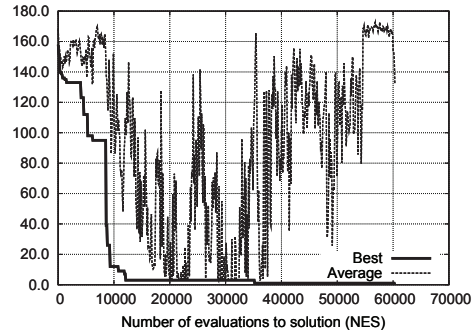
Fig. 8. Influence of local search heuristics on results of HSA-EA solving equi-partite graphs

pure random initialization. Note, in the figure, results for this version of the HSA-EA were obtained after 25 runs, while for the versions of the HSA-EA with the local search heuristics the average results were obtained after performing of all four local search heuristics, i.e. after 100 runs.
In Fig. 8.b results of different versions of the HSA-EA according to the SR are presented. The best results by the instances the nearest to the threshold ($p \in [0.0075 \ldots 0.008]$) are observed by the original HSA-EA. Conversely, the HSA-EA with the random initialization procedure ($Init$) gained the worst results by the instances the nearest to the threshold, while these were better while the edge density was raised regarding the original HSA-EA. The turning point represents the instance of graph with $p = 0.008$. After this point is reached the best results were overtaken by the HSA-EA with the random initialization procedure ($Init$).
In contrary, the best results by the instances the nearest to the threshold according to the AES was observed by the HSA-EA without local search heuristics ($None$). Here, the turning point regarding the performance of the HSA-EA ($p = 0.008$) was observed as well. After this point results of the HSA-EA without local search heuristics becomes worse. Conversely, the HSA-EA with random initialization procedure that was the worst by the instances before the turning point becomes the best after this.
As illustrated by Fig. 8.f, all versions of the HSA-EA leaved in average less than 30 uncolored vertices by the 3-coloring. The bad result by the original HSA-EA coloring the graph with $p = 0.0095$ was caused because of the ordering by saturation local search heuristic that got stuck in the local optima. Nevertheless, note that most important measure is SR.

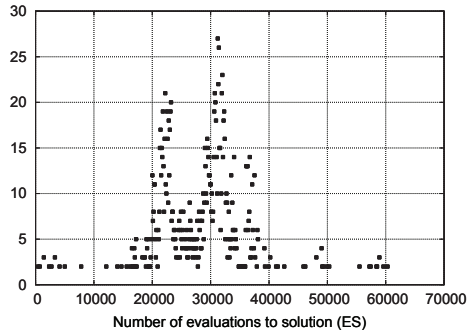
The impact of the neutral survivor selection
In this experiments the impact of the neutral survivor selection on results of the HSA-EA was analyzed. In this context, the HSA-EA with deterministic survivor selection was developed with the following characteristic:

- The Equation 12 that prevents the generation of neutral solutions was used instead of the Equation 10.
- The deterministic survivor selection was employed instead of the neutral survivor solution. This selection orders the solutions according to the increasing values of the fitness function. In the next generation the first μ solutions is selected to survive.

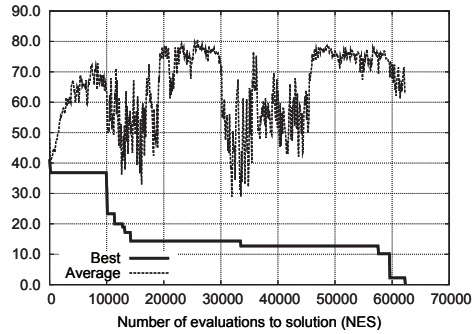
Before starting with the analysis, we need to prove the existence of neutral solution and to establish they characteristics. Therefore, a typical run of the HSA-EA with neutral survivor selection is compared with the typical run of the HSA-EA with the deterministic survivor selection. As example, the 3-coloring of the equi-partite graph with $p = 0.010$ was taken into consideration. This graph is easy to solve by both versions of the HSA-EA. Characteristics of the HSA-EA by solving it are presented in Fig. 9.
In the Fig. 9.a the best and the average number of uncolored nodes that were achieved by the HSA-EA with neutral and the HSA-EA with deterministic survivor selection are presented. The figure shows that the HSA-EA with the neutral survivor selection converge to the optimal solution very fast. To improve the number of uncolored nodes from 140 to 10 only 10,000 solutions to evaluation were needed. After that, the improvement stagnates (slow progress is detected only) until the optimal solution is found. The closer look at the average number of uncolored nodes indicates that this value changed over every generation. Typically, the average fitness value is increased when the new best solution is found because the other solutions in the population try to adapt itself to the best solution. This self-adaptation consists of adjusting the step sizes that from larger starting values becomes smaller and smaller over



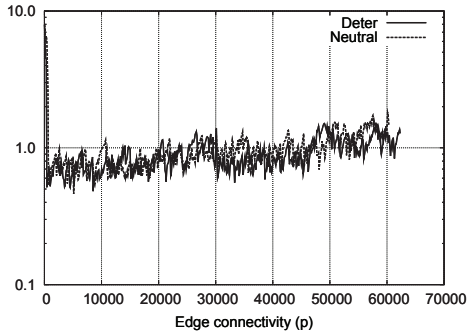
(a) Number of uncolored nodes by neutral selection



(b) Percent of neutral solutions



(c) Number of uncolored nodes by det. selection



(d) Diversity of population

Fig. 9. Characteristics of the HSA-EA runs on equi-partite graph with $p = 0.010$

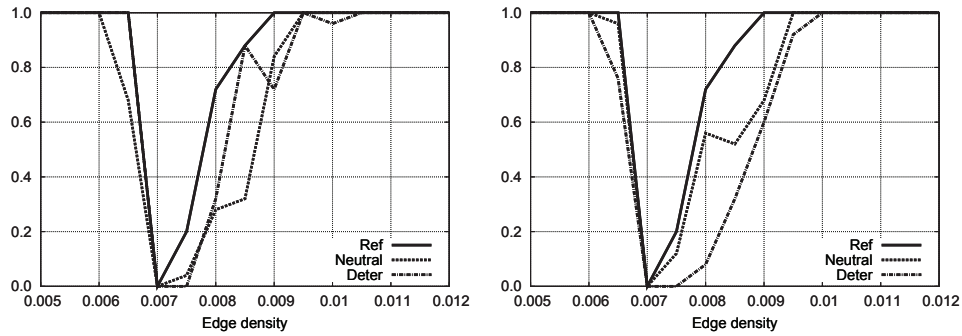
the generations until the new best solution is found. The exploring of the search space is occurred by this adjusting of the step sizes. Conversely, the average fitness values are changed by the HSA-EA in the situations where the best values are not found as well. The reason for that behavior is the stochastic evaluation function that can evaluate the same permutation of vertices always differently.

More interestingly, the neutral solution occurs when the average fitness values comes near to the best (Fig. 9.b). As illustrated by this figure, the most neutral solutions arise in the later generations when the population becomes matured. In example from Fig. 9.b, the most neutral solutions occurred after 20,000 and 30,000 evaluations of fitness function, where almost 30% of neutral solution occupied the current population.

In contrary, the HSA-EA with deterministic survivor selection starts with the lower number of uncolored vertices (Fig. 9.c) than the HSA-EA with neutral selection. However, the convergence of this algorithm is slower than by its counterpart with the neutral selection. A closer look at the average fitness value uncovers that the average fitness value never come close to the best fitness value. A falling and the rising of the average fitness values are caused by the stochastic evaluation function.

In the Fig. 9.d a diversity of population as produced by the HSA-EA with different survivor selections is presented. The diversity of population is calculated as a standard deviation of the vector consisting of the mean weight values in the population. Both HSA-EA from this figure

lose diversity of the initial population (close to value 8.0) very fast. The diversity falls under the value 1.0. Over the generations this diversity is raised until it becomes stable around the value 1.0. Here, the notable differences between curves of both HSA-EA are not observed.
To determine what impact the neutral survivor selection has on results of the HSA-EA, a comparison between results of the HSA-EA with neutral survivor selection (*Neutral*) and the HSA-EA with deterministic survivor selection (*Deter*) was done. However, both versions of the HSA-EA run without local search heuristics. Results of these are represented in the Fig. 10. As reference point, the results of the original HSA-EA hybridized with the swap local search heuristic (*Ref*) that obtains the overall best results are added to the figure. The figure is divided in two graphs where the first graph (Fig. 10.a) presents results of the HSA-EA with heuristic initialization procedure and the second graph (Fig. 10.b) results of the HSA-EA with random initialization procedure according to the *SR*.



(a) Heuristic initialization procedure (b) Random initialization procedure

Fig. 10. Comparison of the HSA-EA with different survivor selections according to the SR

As shown by the Fig. 10.a the HSA-EA with neutral survivor selection (*Neutral*) exposes better results by the instances near to the threshold ($p \in [0.0075 \ldots 0.008]$) while the HSA-EA with deterministic survivor selection (*Deter*) was slightly better by the instance of graph with $p = 0.0085$. Interestingly, while the curve of the former regularly increases the curve of the later is sawing because it raises and falls from the instance to the instance. In contrary, from the Fig. 10.b it can be seen that the HSA-EA with neutral survivor selection outperforms its counterpart with deterministic survivor selection by all instances of random graphs if the random initialization procedure is applied.
In summary, the original HSA-EA with swap local search heuristic used as reference outperforms all observed versions of the HSA-EA.

4.1.3 Summary

In this subsection the characteristics of the HSA-EA were studied on the collection of equi-partite graphs, where we focused on the behavior of the algorithm in the vicinity of the threshold. Therefore, an impact of the hybridizing elements, like the initialization procedure, the local search heuristics, and neutral survivor selection, on results of the HSA-EA are compared. The results of these comparisons in vicinity of the threshold ($p \in [0.0065 \ldots 0.010]$) are presented in Table 1, where these are arranged according to the applied selection (column *Sel.*), the local search heuristics (column *LS*) and initialization procedure (column *Init*).

In column SR average results of the corresponding version of the HSA-EA are presented. Additionally, the column SR_{avg1} denotes the averages of the HSA-EA using both kind of initialization procedure. Finally, the column SR_{avg2} represents the average results according to SR that are dependent on the different kind of survivor selection only.

Sel.	LS	Init	SR	SR_{avg1}	SR_{avg2}
Neut.	No	Rand_1	0.52	0.56	0.62
		Heur_1	0.61		
	Yes	Rand_2	0.66	0.66	
		Heur_2	**0.67**		
Det.	No	Rand_3	0.46	0.53	0.57
		Heur_3	0.61		
	Yes	Rand_4	0.60	0.60	
		Heur_4	0.61		

Table 1. Average results of various versions of the HSA-EA according to the SR

As shown by the table 1, results of the HSA-EA with deterministic survivor selection without local search heuristics and without random initialization procedure ($SR = 0.46$, denoted as Rand_3) were worse than results or its counterpart with neutral survivor selection ($SR = 0.52$, denoted as Rand_1) in average for more than 10.0%. Moreover, the local search heuristics improved results of the HSA-EA with neutral survivor selection and random initialization procedure from $SR = 0.52$ (denoted as Rand_1) to $SR = 0.66$ (denoted as Rand_2) that amounts to almost 10.0%. Finally, the heuristic initialization improved results of the HSA-EA with neutral selection and with local search heuristics from $SR = 0.66$ (denoted as Rand_2) to $SR = 0.67$ (denoted as Heur_2), i.e. for 1.5%. Note that the $SR = 0.67$ represents the best result that was found during the experimentation.
In summary, the construction heuristics has the most impact on results of the HSA-EA. That is, the basis of the graph 3-coloring represents the self-adaptive evolutionary algorithm with corresponding construction heuristic. However, to improve results of this base algorithm additional hybrid elements were developed. As evident, the local search heuristics improves the base algorithm for 10.0%, the neutral survivor selection for another 10.0% and finally the heuristic initialization procedure additionally 1.5%.

5. Conclusion

Evolutionary algorithms are a good general problem solver but suffer from a lack of domain specific knowledge. However, the problem specific knowledge can be added to evolutionary algorithms by hybridizing different parts of evolutionary algorithms. In this chapter, the hybridization of search and selection operators are discussed. The existing heuristic function that constructs the solution of the problem in a traditional way can be used and embedded into the evolutionary algorithm that serves as a generator of new solutions. Moreover, the generation of new solutions can be improved by local search heuristics, which are problem specific. To hybridized selection operator a new neutral selection operator has been developed that is capable to deal with neutral solutions, i.e., solutions that have the different genotype but expose the equal values of objective function. The aim of this operator is to directs the evolutionary search into new undiscovered regions of the search space, while on the other hand exploits problem specific knowledge. To avoid wrong setting of parameters that control the behavior of the evolutionary algorithm, the self-adaptation is used as well. Such

hybrid self-adaptive evolutionary algorithms have been applied to the the graph 3-coloring that is well-known NP-complete problem. This algorithm was applied to the collection of random graphs, where the phenomenon of a threshold was captured. A threshold determines the instanced of random generated graphs that are hard to color. Extensive experiments shown that this hybridization greatly improves the results of the evolutionary algorithms. Furthermore, the impact of the particular hybridization is analyzed in details as well.
In continuation of work the graph *k*-coloring will be investigated. On the other hand, the neutral selection operator needs to be improved with tabu search that will prevent that the reference solution will be selected repeatedly.

6. References

Aarts, E. & Lenstra, J. (1997). *Local Search in Combinatorial Optimization*, Princeton University Press, Princeton.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, New York.

Barnett, L. (1998). Ruggedness and neutrality - the nkp family of fitness landscapes, *in* C. Adami, R. Belew, H. Kitano & C. Taylor (eds), *Alife VI: Sixth International Conference on Articial Life*, MIT Press, pp. 18–27.

Beyer, H. (1998). *The Theory of Evolution Strategies*, Springer-Verlag, Berlin.

Brelaz, D. (1979). New methods to color vertices of a graph, *Communications of the Association for Computing Machinery* 22: 251–256.

Chiarandini, M. & Stützle, T. (2010). An analysis of heuristics for vertex colouring, *in* P. Festa (ed.), *Experimental Algorithms, Proceedings of the 9th International Symposium, (SEA 2010)*, Vol. 6049 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 326–337.

Conrad, M. (1990). The geometry of evolution, *Biosystems* 24: 61–81.

Culberson, J. (2008). Graph coloring page, http://web.cs.ualberta.ca/ joe/Coloring/.

Culberson, J. & Luo, F. (2006). Exploring the k-colorable landscape with iterated greedy, *in* D. Johnson & M. Trick (eds), *Cliques, coloring and satisfiability: Second DIMACS Implementation Challenge*, American Mathematical Society, Rhode Island, pp. 245–284.

Darwin, C. (1859). *On the Origin of Species*, Harward University Press, Cambridge.

Doerr, B., Eremeev, A., Horoba, C., Neumann, F. & Theile, M. (2009). Evolutionary algorithms and dynamic programming, *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 771–778.

Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory, *Proceedings of 6th International Symposium on Micro Machine and Human Science*, IEEE Service Center, Piscataway, NJ, Nagoya, pp. 39–43.

Ebner, M., Langguth, P., Albert, J., Shackleton, M. & Shipman, R. (2001). On neutral networks and evolvability, *Proceedings of the 2001 Congress on Evolutionary Computation*, IEEE Press, pp. 1–8.

Eiben, A., Hauw, K. & Hemert, J. (1998). Graph coloring with adaptive evolutionary algorithms, *Journal of Heuristics* 4: 25–46.

Eiben, A. & Smith, J. (2003). *Introduction to evolutionary computing*, Springer-Verlag, Berlin.

Fister, I., Mernik, M. & Filipič, B. (2010). A hybrid self-adaptive evolutionary algorithm for marker optimization in the clothing industry, *Applied Soft Computing* 10: 409–422.

Fleurent, C. & Ferland, J. (1994). Genetic hybrids for the quadratic assignment problems, *in* P. Pardalos & H. Wolkowicz (eds), *Quadratic Assignment and Related Problems*,

DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS: Providence, Rhode Island, pp. 190–206.

Galinier, P. & Hao, J. (1999). Hybrid evolutionary algorithms for graph coloring, *Journal of Combinatorial Optimization* 3: 379–397.

Ganesh, K. & Punniyamoorthy, M. (2004). Optimization of continuous-time production planning using hybrid genetic algorithms-simulated annealing, *International Journal of Advanced Manufacturing Technology* 26: 148–154.

Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms, *IEEE Transactions on Systems, Man, and Cybernetics* 16: 122–128.

Grosan, C. & Abraham, A. (2007). Hybrid evolutionary algorithms: Methodologies, architectures, and reviews, *in* C. Grosan, A. Abraham & H. Ishibuchi (eds), *Hybrid Evolutionary Algorithms*, Springer-Verlag, Berlin, pp. 1–17.

Hamilton, M. (2009). *Population Genetics*, Wiley-Blackwell, Hong Kong.

Hayes, B. (2003). On the threshold, *American Scientist* 91: 12–17.

Herrera, F. & Lozano, M. (1996). Adaptation of genetic algorithm parameters based on fuzzy logic controllers, *in* F. Herrera & J. Verdegay (eds), *Genetic Algorithms and Soft Computing*, Physica-Verlag HD, pp. 95–125.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge.

Hoos, H. & Stützle, T. (2005). *Stochastic Local Search. Foundations and Applications*, Elsevier, Oxford.

Hynen, M. (1996). Exploring phenotype space through neutral evolution, *Journal of Molecular Evolution* 43: 165–169.

Igel, C. & Toussaint, M. (2003). Neutrality and self-adaptation, *Natural Computing: An International Journal* 2: 117–132.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization, *Proceedings of IEEE International Conference on Neural Networks*, Perth, pp. 1942–1948.

Kim, D. & Cho, J. (2005). Robust tuning of pid controller using bacterial-foraging based optimization, *Journal of Advanced Computational Intelligence and Intelligent Informatics* 9: 669–676.

Kimura, M. (1968). Evolutionary rate at the molecular level, *Nature* 217: 624–626.

Koza, J., Keane, M., Streeter, M., Mydlowec, W., Yu, J. & Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers, Massachusetts.

Lee, M. & Takagi, H. (1993). Dynamic control of genetic algorithms using fuzzy logic techniques, *in* S. Forrest (ed.), *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmmann, San Mateo, pp. 76–83.

Liu, S.-H., Mernik, M. & Bryant, B. (2009). To explore or to exploit: An entropy-driven approach for evolutionary algorithms, *International Journal of Knowledge-based and Intelligent Engineering Systems* 13: 185–206.

Merz, P. & Freisleben, B. (1999). Fitness landscapes and memetic algorithm design, *in* D. Corne, M. Dorigo & F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, London, pp. 245–260.

Meyer-Nieberg, S. & Beyer, H.-G. (2007). Self-adaptation in evolutionary algorithms, *in* F. Lobo, C. Lima & Z. Michalewicz (eds), *Parameter Setting in Evolutionary Algorithms*, Springer-Verlag, Berlin, pp. 47–76.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs*, Springer-Verlag, Berlin.

Michalewicz, Z. & Fogel, D. (2004). *How to Solve It: Modern Heuristics*, Springer-Verlag, Berlin.

Moscato, P. (1999). Memetic algorithms: A short introduction, *in* D. Corne, M. Dorigo & F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill Inc., New York, pp. 219–234.

Murty, U. & Bondy, J. (2008). *Graph Theory: An Advanced Course (Graduate Texts in Mathematics)*, Springer-Verlag, Berlin.

Neppalli, V. & Chen, C. (1996). Genetic algorithms for the two stage bicriteria flowshop problem, *European Journal of Operational Research* 95: 356–373.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog Verlag, Stuttgart.

Schwefel, H.-P. (1977). Numerische optimierung yon computer-modellen mittels der evolutionsstrategie, *Interdisciplinary Systems Research*, Vol. 26, Birkhtiuser Verlag, Basel.

Stadler, P. (1995). Towards a theory of landscapes, *in* R. Lopez-Pena (ed.), *Complex Systems and Binary Networks*, Vol. 461 of *Lecture Notes in Physics*, Springer-Verlag, Berlin, pp. 77–163.

Toussaint, M. & Igel, C. (2002). Neutrality: A necessity for self-adaptation, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1354–1359.

Tseng, L. & Liang, S. (2005). A hybrid metaheuristic for the quadratic assignment problem, *Computational Optimization and Applications* 34: 85–113.

Wang, L. (2005). A hybrid genetic algorithm-neural network strategy for simulation optimization, *Applied Mathematics and Computation* 170: 1329–1343.

Wilfried, J. (2010). A general cost-benefit-based adaptation framework for multimeme algorithms, *Memetic Computing* 2: 201–218.

Wolpert, D. & Macready, W. (1997). No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* 1: 67–82.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution, *Proceedings of the 6th International Congress of Genetics 1*, pp. 356–366.

2

Linear Evolutionary Algorithm

Kezong Tang[1,2], Xiaojing Yuan[2], Puchen Liu[3] and Jingyu Yang[1]
[1]*Computer Science and Technology Department Nanjing University of Science and Technology,*
[2]*Engineering Technology Department University of Houston,*
[3]*Department of Mathematics University of Houston,*
[1]*China*
[2,3]*United States*

1. Introduction

During the past three decades, global optimization problems (including single-objective optimization problems (SOP) and multi-objective optimization problems (MOP)) have been intensively studied not only in Computer Science, but also in Engineering. There are many solutions in literature, such as gradient projection method [1-3], Lagrangian and augmented Lagrangian penalty methods [4-6], and aggregate constraint method [7-9]. Among these methods, penalty function method is an important approach to solve global optimization problems.. To obtain the optimal solution of the original problem, the first step is to convert the optimization problem into an unconstrained optimization problem with a certain penalty function (such as Lagrangian multiplier). As the penalty multiplier approaches zero or infinite, the iteration point might approach optimal too. However, at the same time, the objective function of the unconstrained optimization problem might gradually become worse. This leads to increased computational complexity and long computational time in implementing the penalty function method to solve the complex optimization problems. In most of the research, both the original constraints and objective function are required to be smooth (or differentiable). However, in real-world problem, it is seldom to be able to guarantee a derivative for of the specific complex optimization problem. Hence, the development of efficient algorithms for handling complex optimization problems is of great importance. In this chapter, we present a new framework and algorithm that can solve problems belong to the family of stochastic search algorithms, often referred to as evolutionary algorithms.

Evolutionary algorithms (EAs) are stochastic optimization techniques based on natural evolution and survival of the fittest strategy found in biological organisms. Evolutionary algorithms have been successfully applied to solve complex optimization problems in business [10,11], engineering [12,13], and science [14,15]. Some commonly used EAs are Genetic algorithms (GAs)[16], Evolutionary Programming (EP)[17], Evolutionary Strategy (ES)[18] and Differential Evolution (DE)[19]. Each of these methods has its own characteristics, strengths and weaknesses. In general, a EA algorithm generate a set of initial solutions randomly based on the given seed and population size. Afterwards, it will go through evolution operations such as cross-over and mutation before evaluated by the

objective function. The winning entity in the population will be selected as the parents (or seed) of the next generation (i.e., iteration). The optimization iteration continues until the termination criteria are satisfied. Typically, either the evolution process reached user defined maximum number of iteration or the improvement in objective function between the two generations converges.

The major advantages of the improved EAs compared with traditional optimization techniques include [20-23]:

1. EAs do not require objective function to be continuous and can be used in algebraic form.
2. EAs tend to escape more easily from local optimum due to the randomness introduced at the beginning and perturbation introduced by the mutation operation. The amount of perturbation is a parameter defends on the step size specified by the user.
3. EAs do not require specific domain information or prior knowledge although they can exploit it if such information is available. It does not involve calculation of the gradients of the objective function.
4. EAs are conceptually simple and relatively easy to implement.

The major disadvantages of EAs are their poor performance in handling constraints, long computational time, and high computational complexity, especially when the solution space is hard to explore. To overcome these difficulties, some 'more intelligent' rules and /or hybrid techniques such as evolutionary-gradient search (EGS) have been developed to extend EAs to overcome the slow convergence phenomena of the EAs near the optimum solution [24-27]. In addition, improving fitness function, crossover and mutation operators, selection mechanisms, and adaptive controlling of parameter settings all enhance EA's efficiency and performance. An excellent comparison study of evolutionary algorithms has been published for global optimization problems by Michalewicz and Schoenauer [28].

Among the evolutionary algorithms the methods based on penalty functions have proven to be the most popular. These methods augment the cost function, so that it includes the squared or absolute values of the constraint violations multiplied by penalty coefficients. However, there are also serious drawbacks with penalty function methods. For example, small values of the penalty coefficients drive the search outside the feasible region and often produce infeasible solutions [29], if imposing very severe penalties makes it difficult to drive the population to the optimum [29-31]. To overcome these drawbacks, Kim and Myung [26] proposed the concept of two phase evolutionary algorithm, where the penalty method is implemented in the first phase, while during the second phase an augmented Lagrangian function is applied on the best solution of the first phase. Tahk and Sun [32] presented the co-evolutionary augmented Lagrangian method which uses an evolution of two populations with opposite objectives to solve constrained optimization problems. Tang proposed a special hybrid genetic algorithm (HGA) [33]with penalty function and gradient direction search, which uses mutation along the weighted gradient direction as the main operation and only in the later generation it utilizes an arithmetic combinatorial crossover. The approach presented in [34] is an extended hybrid genetic algorithm (EHGA), which is a fuzzy-based methodology that embeds the information of the infeasible points into the evaluation function.

Based on the above analysis, our major concern in this chapter was how to design a linear fitness function based on the general penalty function so as to fast evaluate candidate solutions, regardless of the design variables' dimensions of solving the complex optimization problems. The major advantage of linear function is their simplicity and

computational attractiveness. The chapter starts with the review that we have a briefly review for various evolutionary approaches in the last years. In the sequel we will focus on this compression process, in which search space of design variables will be compressed into 2-dimensional performance space and it is possible to fast discriminate 'good' solutions from candidate solutions, regardless of the complexity of original space. In addition, our method combines two improved operators in reproduction phase, i.e., crossover and mutation. Simulation results over a comprehensive set of benchmark functions show that our method is feasible and effective. Meanwhile, it can provide good performance in terms of uniformity and diversity of solutions.

2. Description of EA

Evolutionary algorithm is a random search based optimization technique. Various applications have shown that when problems are formulated properly, EA can give good results with reasonable time complexity. EA mimics the process of natural selection and starts with artificial individuals (represented by a population of "chromosomes"). EA tries to evolve those individuals that are fitter and, by applying genetic operators (crossover and mutation), it attempts to produce descendants that are better than their parents in terms of a certain quantitative measure. In spite of their diversity, most of them are based on the same iterative procedure.

As a heuristic population-based method, EA is really like a "black box", completely independent from the characteristic of the problem. Fig.1 presents the classical EA flow chart. An initial population of individuals is generated randomly. Each of these individuals is evaluated in terms of a certain "fitness function" that can "guide" EA to the desired region of the search space. EA's three genetic operators (Selection, Crossover and Mutation) are the main components to improve the EA's behavior. Selection is the process that mimics the "survival of the fittest" principle in the biological theory of evolution. Firstly, the selection operator assures that individuals are copied to the next generation with a probability associated to their fitness values. Although selection is implemented in a EA as a policy for determining the best candidate individuals that will be presented in the next generation with a higher probability, it does not search the space further, because it just copies the previous candidate individuals. The search results from the creation of new individuals from old ones. Secondly, the crossover operator is implemented in EA by exchanging chromosome segments between two randomly selected chromosomes.

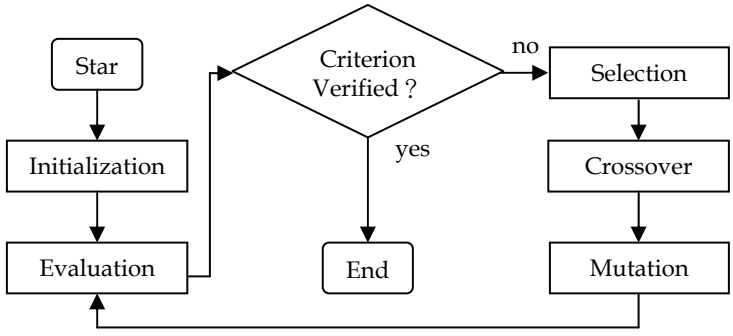


Fig. 1. Evolutioary algorithm flow chart

Crossover process provides a mechanism to allow new chromosomes to inherit the properties from old ones. Thirdly, mutation is a random perturbation to one or more genes in the chromosomes during evolutionary process. The purpose of the mutation operator is to provide a mechanism to avoid local optima by exploring the new regions of the search space, which selection and crossover could not fully guarantee. The searching process terminates when the predefined criterion is satisfied.

3. Related concepts of MOP

Almost all real world engineering designing problems are characterized by the presence of several conflicting and/or cooperating objectives, as opposed to having a single objective, and result in a set of non-dominated solutions. This set, generally called Pareto front, helps the decision-maker to identify the best compromise solutions by eliminating inferior ones and articulating his preference pertaining to the different objectives once he has an additional knowledge of the Pareto frontier. The term MOP is used to broadly classify problems with more than one objective. Without loss of generality, a general multi-objective optimization problem can be expressed in the following equations:

$$\min F(x) = (f_1(x), f_2(x), ..., f_k(x))^T \tag{1}$$

$$s.t. G(x) = (g_1(x), g_2(x), ..., g_l(x))^T \le 0 \tag{2}$$

$$H(x) = (h_{l+1}(x), h_{l+2}(x), ..., h_m(x)) = 0 \tag{3}$$

where k is the number of objective functions, l and m-l are the number of unequal and equal constraints respectively, and the vector G(x) represents constraints that probably are easily handled explicitly, such as lower and upper bounds on the variables, $x=(x_1,x_2,\ldots,xn) \in S \subseteq \Omega$. n is the number of designing variables. Ω and S are the searching space of objective function and the feasible searching space, respectively.

The Pareto optimal concept is initially introduced by Vilfredo Pareto in the 19th century, and the concept has already been widely used in MOP to aid designers in their decision-making processes. In this paper, we assume that all objectives are to be minimized for clarity purpose since maximization of any maximization of any $-f(\bullet)$. The Pareto optimal concept is stated as follows[35,36]:

Definition 1 order relation between design vectors. Let x and x' be two designing variables. The dominance relations in a minimization problem are:

x dominates x' ($x \prec x'$), *iff* $f_t(x) < f_t(x')$ and $f_{t'}(x) \not> ft'(x')$, $\forall\ t' \neq t \in [1, k]$.

x are incomparable with x' ($x \sim x'$),*iff* $f_t(x) < f_t(x')$ and $f_{t'}(x) > ft'(x')$, $t' \neq t \in [1, k]$.

Definition 2 Pareto-optimal solution. A solution x is called Pareto-optimal if there is no other $x' \in F$, such that $f(x') < f(x)$. All the Pareto-optimal solutions define the Pareto-optimal set.

Definition 3 Non-dominated solution. A solution $x \in S$ is non-dominated with respect to a set $x' \in S$ if and only if $\nexists\ x' \in S$, verifying that $x' \prec x$.

Definition 4 Non-dominated set. Given a set of solutions S', such that $S' \in S$ and $Y'=f(S')$, the function $h(S')$ returns the set of non-dominated solutions from S':

$$h(S')=\{\ \forall\ x \in S'\ \mid\ x \text{ is non-dominated by any other } z',\ z' \in S'\ \}$$

Fig.2 graphically describes the process of mapping from designing space to objective space with objectives (f_1 and f_2), and the Pareto-optimal set are shown as the Pareto optimal front. All of the Pareto solutions in designing space are equally important and all are the global optimal solutions. The decision-maker articulates his preference pertaining to the different objectives once he has knowledge of the Pareto front.

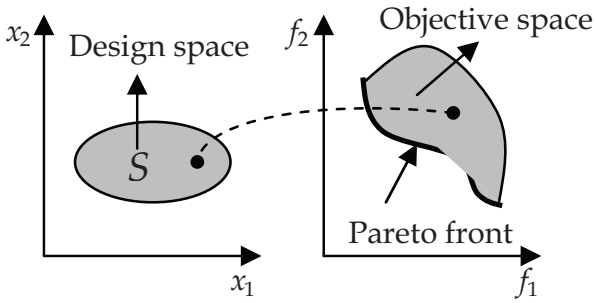


Fig. 2. Mapping from design space to objective space.

4. The linear fitness function (LFF)

A very popular approach for handling complex constraints by using EAs is to adopt penalty function such as [37-39]. When handling individuals violating any one of the constraints, if the added penalties do not depend on the current iteration number and remain constant during the entire evolutionary process, then the penalty function is called static penalty function, and its penalties are weighted sum of all constraint violations. If, alternatively, the current iteration number is considered while determining the penalties, then the penalty function is called dynamic penalty function. In this paper, we adopt static penalty function as the following manner:

$$p_j(x) = \begin{cases} \max(0, g_j(x)), & 1 \le j \le q \\ |h_j(x)|, & l+1 \le j \le m \end{cases} \tag{4}$$

where $p_j(x)$ denotes the degree of individual violating constraints. The generally fitness function for evaluating individuals can be defined as below:

$$fitness(x) = f(x) + r \times \sum_{j=1}^{m} p_j(x) \tag{5}$$

$$f(x) = \sum_{i=1}^{k} w_i f_i(x) = w^T F(x),$$

where $f(x)$ is a convex combination of the different objectives in that the multi-objective problem is converted into a scalar optimization one. The weighted value w_i is chosen such that $w_i \ge 0$, i=1,…,n, and $\sum_{i=1}^{k} w_i = 1$.

One of existing difficulties in penalty function is mainly that parameter r is not easily to be selected and controlled [37]. For many MOP, we note that the searching space of the design

vector is always situated in n-dimension space (n≥2). In terms of human imagination of space, if we can attempt to give a good method to transform n-dimension space into low dimensional space for MOP since the dimensions below three are geometrically prone to human understanding. Therefore, we give the following transforming procedure.

$y_1 = f(x), y_2 = \sum_{i=1}^{m} p_i(x)$. If we can make an appropriate mapping between vector $y = [y_1, y_2]$ and $\bar{x}$, then *fitness*(x) can be represented as a linear fitness function:

$$fitness(\bar{x}) = a^T \bar{x} = \sum_{i=1}^{2} a_i \bar{x}_i, \tag{6}$$

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ r \end{bmatrix}, \bar{x} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = y$$

without loss of generality, the variable $\bar{x}$ is still denoted by x. x is a point in 2-dimension space. $a^T y = 0$ ascertains a hyperplane via origin, then the entire search space is divided into two subspaces with 2-dimension respectively. a is a normal vector in hyperplane (see Fig.3). The division of initial searching space with n-dimension is equivalent to the above results, and the searching process is focused on Φ_1. A mapping process is described between feasible region S and corresponding abnormity region Φ_3 while any point x in 2-dimension space is required to satisfy one of the following expressions:

$$fitness(x) \begin{cases} > 0, x \in \Phi_1, \\ < 0, x \in \Phi_2, \\ = 0, x \in H, \end{cases} \tag{7}$$

Further analysis shows that linear fitness function may be regarded as an algebraic measurement from point x to the hyperplane.

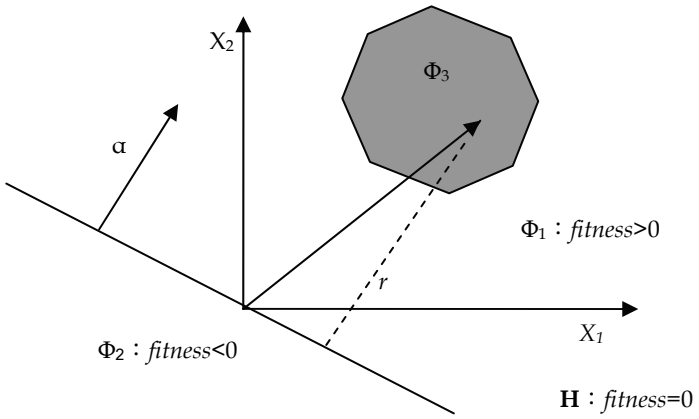


Fig. 3. Transformation of searching space.

By means of above anaysis, we give a new linear fitness function to evaluate individuals:

$$fitness(x) = \frac{a^T x}{||a||} \tag{8}$$

While evaluating new individuals we may directly use formula (8) without considering the feasibility of individual. It is very convenient to identify the good or bad individuals from population according to corresponding fitness values.

5. Evolutionary operators

In conventional EAs, the basic operators are selection and mutation. The selection assigns greater probabilities to the fittest individuals according to its fitness. Crossover is the exchange of information between different individuals, and it is the principal process in generating new individuals. Mutation is a security factor to avoid entrapment at any other point when the population is completely converged. The parents undergo crossover and mutation to generate two new children, and each individual in the population evolve to get higher generation by generation.

5.1 Crossover operator

The crossover operator based on a modified self-adaptive density adopts two parents to generate children [40]. The procedure is formulated as follows:

First, two individuals are selected using a random selection procedure to generate new individuals. The two parents can be expressed as $X=(x_1, x_2, \ldots, x_d)$ and $Y=(y_1,y_2,\ldots,y_d)$, where $x_i < y_i$.

$$[x_i + \frac{x_i - y_i}{2}, y_i + \frac{y_i - x_i}{2}] = [\bar{x}_i, \bar{y}_i] \tag{9}$$

Here, we give a density function in (10). It depends on two parameters a and β, and a, $\beta \in [0,1]$.We give a cumulated distribution function in (11) and $0 \le G_{x_i,y_i} \le 1$. It ascertains parameter η_i in children by (12). $G^{-1}_{x_i,y_i}$ is the inverse function of G_{x_i,y_i} in (12).

$$g_{x_i,y_i}(x,\alpha,\beta) : [\bar{x}_i, \bar{y}_i] \times \alpha \times \beta \to r \tag{10}$$

$$G_{x_i,y_i}(x,\alpha,\beta) : [\bar{x}_i, \bar{y}_i] \times \alpha \times \beta \to [0,1] \tag{11}$$

$$\eta_i = G^{-1}{}_{x_i,y_i}(r,\alpha,\beta) \tag{12}$$

The experimental results show that density crossover operator is good at finding optimal solutions and enlarging the search region.

5.2 Mutation operator

The neighborhood mutation operator is introduced in Ref.[41]. Individual mutation is in its neighborhood space by using $x'=x+a\times r$, where x' and x are child and parent respectively, a is a uniformly random number in [-1,1], and r is the radius of neighborhood space dynamically compressed using $r=r\times cr$, where $cr=(1+0.1^b\times c)^{-t}$ and cr is the compression ratio, b and c are random integer between 0 and 9, t is the current generation.

Although the neighborhood mutation operator evenly scans the whole neighborhood space of individual at the beginning, the exploration becomes localized with the generation increasing. In order to overcome this deficiency, the following improvements are made on the neighborhood mutation operator. Give a binary number δ, add the following part.

$$x'_{new} = \begin{cases} x' + \lambda(t, b_i - x'), if & \delta = 0 \\ x' - \lambda(t, x_i - a_i), if & \delta = 1 \end{cases} \tag{13}$$

$$\lambda(t,\tau) = \tau(1 - r^{\gamma}), \gamma = (1 - t / n_{\max})^{\beta}$$

where r is a random number in [0,1], $n_{\max}$ and β denote total iterative number and random number respectively. The added part uniformly explores the whole searching space during the execution of the algorithm. Hence, it may overcome the deficiency of the neighborhood mutation operator. By integrating the above two parts, not only the abilities of the global search and local exploration are balanced, but also the diversity of the population increases. As a result, the premature convergence is avoided in a way. The experimental results show that the improved mutation operator is very feasible for the known searching space and insure x'_{new} is a random number in [a_i, b_i].

6. Numerical experiments

6.1 Linear evolutionary algorithm

The LEA has a different design from other variants of EAs. It does not use any information from the dominated individuals. In each generation, we only preserve nondominated individuals. The number of generated children is a fixed constant η. However, we limit the number of the nondominated individuals using a nearest neighborhood distance function (NNDF) in Ref.[41], which can help disperse the non-dominated individuals. The LEA is described as follows:

Step 1. (***Initialization***). Generate an initial population containing N_{pop} individuals where N_{pop} is the number of individuals in each population.

Step 2. (***Evaluation***). Calculate the fitness values of the generated individuals using the LFF. Update a tentative set of non-dominated solutions. The number of non-dominated solutions is $u_{nd.}$

Step 3. (***Selection***). If $u_{nd} > u_{max}$, select u_{max} individuals using NNDF, then $u_{nd} = u_{max}$.

Step 4. (***Crossover and mutation***). Generate the η offspring using crossover based on density and modified neighbourhood space mutation.

Step 5. (***Termination test***). If a prespecified stopping condition is not satisfied, return to step 2.

In step 3, we control the number of non-dominated individuals so as not to exceed a maximum number, u_{max}. Hence the generated offspring are under control. To filter better individuals from the tentative set, we evenly distribute the individuals on the Pareto front using NNDF.

Consider the entire complexity of one iteration of the proposed algorithm, the overall complexities of the algorithm are focused on evaluation and selection. Assuming the solved optimization problems totally have m decision variables, and population size is n. The basic operations and their worst-case complexities are as follows:

1. Evaluation in step 2 $o(2n(n+m))$.
2. Selection in step 3 $o(n^2)$.

The overall complexity of the algorithm is $o(2n(n+m))$, which is governed by evaluation in step2 (including linear mapping for decision variables from n-dimension to 2-dimension and updating a tentative set of non-dominated solutions).

6.2 Simulation results

Simulations are performed in MATLAB with a 2 GHZ Pentium PC. In our study, five numerical constrained optimization problems were divided into two groups ($G1$ and $G2$) were applied to test the LEA. These benchmark problems are taken from Ref.[37] and [41]. $G1$ only contains inequality constraints in test problems which involving two designing variables. $G2$ contains inequality and equality constraints of single objective optimization problems which have no limitation on the number of designing variables. For all conducted experiments, four parameters of the LEA, namely, population size (Pop_t), iterative number (NG), Crossover probability (CP) and mutation probability (MP) are set 200, 300, 0.9, 0.05. All problems are repeated for 200 in the same environment.

$G1$: *Test Problem 1 BNH*

$$\min F_1(x) = (f_1(x), f_2(x)); f_1(x) = 4x_1^2 + 4x_2^2; f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2$$

$$s.t.\ C_1(x) = (x_1 - 5)^2 + x_2^2 \le 25; C_2(x) = (x_1 - 8)^2 + (x^2 + 3)^2 \ge 7$$

$$0 \le x_1 \le 5, 0 \le x_2 \le 3\ .$$

Test problem 2 TNK

$$\min F_2(x) = (f_1(x), f_2(x)); f_1(x) = x_1; f_2(x) = x_2;$$

$$s.t.\ C_1(x) = x_1^2 + x_2^2 - 0.1\cos(16\arctan(x_1 / x_2)) - 1 \ge 0;$$

$$C_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \le 0.5\ ;$$

$$0 \le x_1 \le \pi, 0 \le x_2 \le \pi\ .$$

Test problem 3 Constr-Er

$$\min F_3(x) = (f_1(x), f_2(x)); f_1(x) = x_1; f_2(x) = (1 + x_2) / x_1;$$

$$s.t.\ C_1(x) = 9x_1 + x_2 - 6 \ge 0; C_2(x) = 9x_1 - x_2 - 1 \ge 0;$$

$$0.1 \le x_1 \le 1, 0 \le x_2 \le 5\ .$$

Here, the circular symbols represent the results obtained using NSGA-II or LEA in six figures. For case 1, *BNH* is a two-objective function problem with a convex Pareto front and constrained conditions are two inequalities. LEA gives a very good approximation of the Pareto front by obtaining evenly distributed solutions, as shown in Fig.5. However, Fig.4

shows that although a number of optimal solutions are obtained using NSGA-II, in terms of diversity of solutions, these solutions are not evenly spread out over the entire front. There exists some disconnected spaces on Pareto front in Fig.4. The Pareto front consists of $x_1^* = x_2^* \in [0,3]$, $x_1^* \in [3,5]$ and x_2^*=3. For case 2, constraint conditions are also two inequalities' state. The Pareto front is well predicted, and a number of optimal solutions obtained are spread out over the entire front using the LEA in Fig.7. Decision makers make a final choice of optimal solution according to real conditions from Pareto optimal set. But result of Fig.6 shows that we probably are not able to well predict the three disconnected curves. For case 3, it is the two-objective function problem of Constr-Ex with two-dimensional curve. Fig.9 shows the predicted Pareto front in the two-dimensional objective space obtained by the LEA. The method can capture the distinct solution along this front. Fig.8 shows that comparative method can performs well elsewhere along the Pareto front.

We adopt some performance measures described in [42] so as to obtain more quantitative measures of algorithm performances. These performance metrics are generational distance and the diversity metric. The performance of algorithm is measured both in terms of the proximity to the true Pareto front that is achieved as well as in terms of the diversity of the optimal solutions. The means and variance of these measures are evaluated by conducting 20 distinct runs of each simulation. The results are tabulated in table 1. From table 1, we can see that the LEA has better convergence performance than NSGA-II because it is obvious that the generational distance metric is larger than the later, and diversity metric is also larger using NSGA-II. The LEA is an effective and robust method.

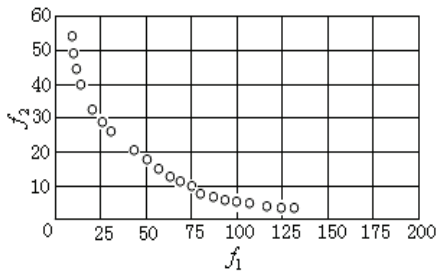


Fig. 4. Pareto front on BNH using NSGA-II

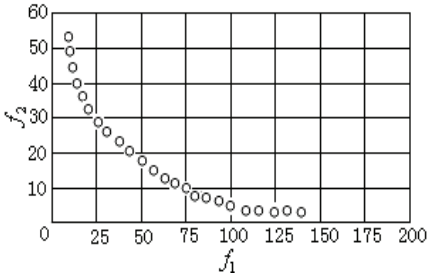


Fig. 5. Pareto front on BNH using LEA

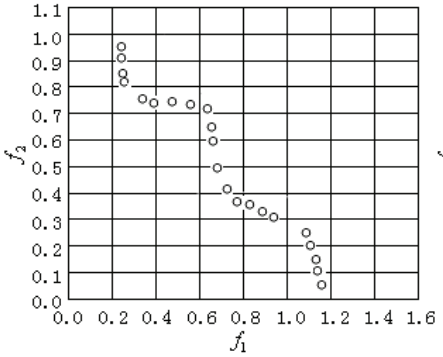


Fig. 6. Pareto front on TNK using NSGA-II

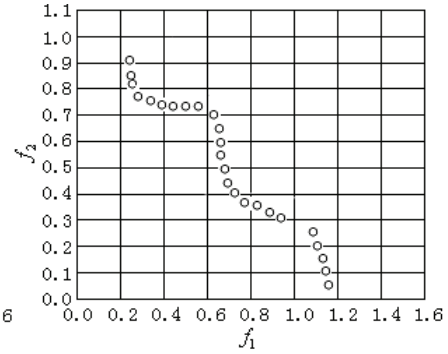


Fig. 7. Pareto front on TNK using LEA

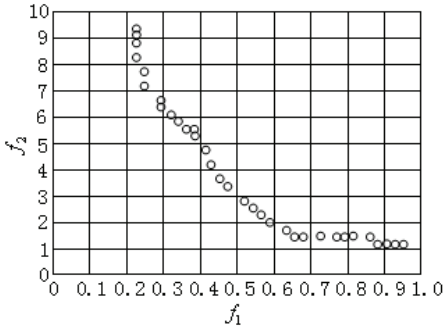


Fig. 8. Pareto front on CE using NSGA-II

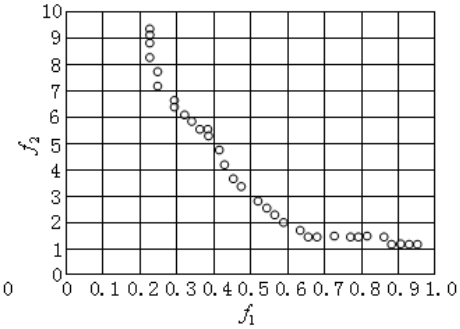


Fig. 9. Pareto front on CE using LEA

Test problems	Generational distance		Diversity	
	Mean	Variance	Mean	Variance
BNH	0.0032356	0.0000009	0.4516783	0.00156753
	0.0026733	0.0000005	0.4356745	0.00107867
TNK	0.0051289	0.0000012	0.2389783	0.00825676
	0.0050125	0.0000011	0.2337892	0.77867826
CE	0.0043216	0.0000011	0.3567882	0.00578933
	0.0040102	0.0000010	0.3389563	0.00623124

Table 1. The results of mean and variance on test problems using NSGA-II and LEA respectively.

G2: Test Problem 2 TP1

$$\min g(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

$$s.t.\ \ x_4 - x_3 + 0.55 \geq 0, -x_4 + x_3 + 0.55 \geq 0 \ ;$$

$$1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0;$$
$$1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0;$$
$$1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

$$0 \leq x_i \leq 1200(i = 1,2) \ ; \ 055 \leq x_i \leq 0.55(i = 3,4) .$$

Test Problem 2 TP2

$$\min g(x) = e^{x_1 x_2 x_3 x_4 x_5}$$

$$s.t.\ \ x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 ;$$

$$x_2 x_3 - 5x_4 x_5 = 0, x_1^3 + x_2^3 + 1 = 0 ;$$

$$-2.3 \leq x_i \leq 2.3(i = 1,2) \ ; \ -3.2 \leq x_i \leq 3.2(i = 3,4,5) .$$

Two test problems in second group contain equality and inequality's constraints that are tested for single objective optimization problems. In many real-world optimization problems, optimal solutions of them are obtained by transforming MOP into single objective optimization. Thus, the research of single objective optimization is also very improtant in engineering application areas. For *TP*1, we find total optimal solutions for 25, (668.94675327911, 1013.10377656821, 0.10773654866, 0.39654576851). Distance of optimal solution is 2.3323246783310e-13, which is corresponding to optimal value 5198.5467. For test problems 2, we find total optimal solutions for 21, x = (-1.77365745561,1.45675698761,-1.5678457772,0.66755656893,-0.75778765788), which is corresponding to the optimal value 0.055894567. The mean and worst values of solutions are formulated for two test problems in Table 2.

For *TP*1 and *TP*2, results of the first row are obtained using LEA, similarly, the second row takes a Pareto strength evolutionary algorithm [37] (denoted by ZW). The third row means results of a random sorting [43] (denoted by RY). From Table 2, we can see that the LEA outperforms other two algorithms in terms of experimental data involving the best value, mean and the worst value, as demonstrate the LEA is a robust algorithm with generality and effectivity.

Problems		Best	Mean	Worst
*TP*1	LEA	5126.4266	5126.5461	5126.9586
	ZW	5126.49811	5126.52654	5127.15641
	RY	5126.497	5128.881	5142.472
*TP*2	LEA	0.053945563	0.053999775	0.054993677
	ZW	0.053949831	0.053950257	0.053972292
	RY	0.053957	0.057006	0.216915

Table 2. Comparison among LEA(new algorithm), ZW(in Ref.[27]) and RY(in Ref.[33]) (40 independent run)

7. Conclusion

In this paper, we propose an approach of using the LEA to optimize MOP, which finds the optimal solutions using the method of transforming the search space with high dimensions into low dimensional space. Numerical experiments show that the LEA performed well in the problems of two groups in terms of the quality of the solutions found. Moreover, the LEA is also a fast and robust method. A future work aims to validate this new optimization method on real-life engineering optimiztion problems (e.g. issued from mechanical engineering and power systems).

8. References

[1] Yang, J.B. (1999). Gradient projection and local region search for multiobjective optimisation. *European Journal of operational Research*. 112(2), 432-459.

[2] Harada, K. ; Sakuma, J.; Ono, I.; Kobayashi, S. (2007). Constraint-handling method for multi-objective function optimization: Pareto descent repair operator. *In 4th International Conference of Evolutionary Multi-Criterion Optimization*. pp.156-170.

[3] Miettinen, K.; Makela, M. M.(1993). Interactive method for nonsmooth multiobjective optimization with an application to optimal control. *Optimization Methods and Software*. 2(1):31-44.

[4] Bazaraa,M.S.; Shetty,L.M.(1993). *Non-linear programming: theory and algorithms*. New York:Wiley.
[5] McCormick, G.P. (1983). *Nonlinear programming: theory,algorithms and applications*.New York: Wiley.
[6] Zhang, J.Z.; Zhang, L.W.(2010). An augmented Lagrangian method for a class of inverse quadratic programming problems. *Applied Mathematics and Optimization*. 61(1):57-83.
[7] Yu, B.; Feng, G.C.; Zhang, S.L.(2001).The aggregate constraint homotopy method for nonconvex nonlinear programming. *Nonlinear Analysis*. 45(7):839-847.
[8] Thakur, M.; Wang, L.Z.; Hurhurgh, C.R. (2010). A multi-objective optimization approach to balancing cost and traceability in bulk grain handling. *Journal of Food Engineering*. 101(2):193-200.
[9] Li, Y.Y.; Tan, T.; Li, X.S.(2010). A gradient-based approach for discrete optimum design. *Structural and Multidisciplinary Optimization*. 41(6):881-892.
[10] Su, C.T.; Chiang, C.L.(2004). An incorporated algorithm for combined heat and power economic dispatch. *Electric Power Systems Research*. 69(2-3):187-195.
[11] Wu, J. Y. (2010). Computational Intelligence-Based Intelligent Business Intelligence System: Concept and Framework.*In Proceedings of the 2010 Second International Conference on Computer and Network Technology*. pp. 57-89.
[12] Carlos,A.; Coello,C.(2000). Use of a self-adaptive penalty approach for engineering optimization problems. *computers in Industry*.41(2):113-127.
[13] Falconer, M.; Greenwood, G.; Morgan, K.;et al. (2010). Using evolutionary algorithms for signal integrity assessment of high-speed data buses. *Journal of Electronic Testing: Theory and Applications*. 26(3): 297-305.
[14] Rahnamayan, S.; Tizhoosh, H.R.; Salama,M.M.A. (2007). A novel population initialization method for accelerating evolutionary algorithms.*Computers and Mathematics with Application*. 53(10):1605-1614.
[15] Nobakhti, A.(2010). On natural based optimization. *Cognitive Computation*. 2(2):97-119.
[16] Ge, J.K.; Qiu, Y.H.; Wu, C.M.; et al. (2008). Summary of genetic algorithms research. *Application Research of Computers*. 25(10):2911-2916. (In Chinese).
[17] Dong, H.B.; He, J.; Huang, H.K.; et al. (2007). Evolutionary programming using a mixed mutation strategy. *Information Sciences*. 177(1):312-327.
[18] Jürgen, P.(2004). Finding optimal decision scores by evolutionary strategies. *Artificial Intelligence in Medicine*. 32(2):85-95.
[19] Babu, B.V.; Chakole, P.G.; Syed Mubeen, J.H.(2005). Multiobjective differential evolution (MODE) for optimization of adiabatic styrene reactor. *Chemical Engineering Science*. 60(17): 4822-4837.
[20] John w. Fowler,; Esma S. Gel.; Murat M.K□ksalan. ;et al.(2010). Interactive evolutionary multi-objective optimization for quasi-concave preference functions. *European Journal of Operational Research*. 206(2):417-425.
[21] Sun, D.Y.; NI, S.C.; Huang, T.C.(2010). Apply a novel evolutionary algorithm to the solution of parameter selection problems. *Applied Mathematics and Ccomputation*. 216(11):3343-3354.
[22] Mitchell, M. *An introduction to genetic algorithms*. Cambridge, MA:MIT Press,1996.
[23] Sarimveis, H.; Nikolakopoulos, A. (2005). A line up evolutionary algorithm for solving nonlinear constrained optimization problems. *Computers & Operations Resarch*. 32(6): 1499-1514.
[24] Glannakoglou,K.C.; Papadimitriou, D.I.; Kampolis, I.C.(2006).Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels. *Compute Methods in Applied Mechanics and Engineering*. 195(44-47):6312-6329.

[25] Rada. R.(1982). Evolutionary search: Gradients and information.*Biosystems*. 15(2):169-177.
[26] Kim J. H, Myung H. ; Evolutionary programming techniques for constrained optimization problems. IEEE Trans Evol Comput 1997;1(2):129-140.
[27] Salomon R. Evolutionary algorithms and gradient search: similarities and differences. IEEE Transa Evol comput 1997;2(2):45-55;
[28] Michalewicz, Z.; Schoenauer, M.(1996). Evolutionary algorithms for constraint parameter optimization problems. *Evolutionary Computation*. 4(1):1-32;
[29] Michalewicz, Z. *Genetic algorithms + data structures = evolution programming*. New York: Springer,1996.
[30] Anderson,E. J.; Ferris, M. C.(1994). Genetic algorithms for combinatorial optimization: the assembly line balancing problem. *ORSA Journal on Computing*.6(2):161-73..
[31] Coit, D. W.; Smith, A.E.; Tate, D.M. (1996). Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*. 8(2):173-82.
[32] Tahk, M.J.; Sun, B.C. (2000). Co-evolutionary augmented Lagrangian methods for constrained optimization. IEEE Transactions on Evolutionary Computation . 4(2):114-24.
[33] Tang, J.; Wang, D.; Ip, A.; Fung, R.Y.K.(1998). A hybrid genetic algorithm for a type of nonlinear programming problem. *Computers and Mathematics with Application*. 36(5):11-21.
[34] Fung, R.Y.K.;Tang, J.F; Wang, D.W.(2002). Extension of a hybrid genetic algorithm for nonlinear programming problems with equality and inequality constraints. *Computers and operations Research*. 29(3):261-74.
[35] Kelner, V.; Capitanescu, F. (2008). A hybrid optimization technique coupling an evolutionary and a local search algorithm. *Computational and Applied Mathematics*. 215(2): 448-456.
[36] Tang, K.Z.; Yang, J. Y.; Gao, S.; Sun, T.K.(2010). A self adaptive linear evolutionary algorithm for solving constrained optimization problems. *Journal of Control Theory and Application*. Online First™,28 March.
[37] Zhou, Y.R.; Li, Y. X.(2003). A Pareto Strength evolutionary algorithm for constrained optimization. *Journal of Software*, 14(7) : 1243-1249. (In Chinese)
[38] Sarimveis, H.; Nikolakopoulos, A. (2005). A line up evolutionary algorithm for solving nonlinear constrained optimization problems. Computers and Operations Research, 32(6):1499-1514.
[39] Varadarajan, M.; Swarup, K.S. (2008). Differential evolutionary algorithm for optimal reactive power dispatch. Electrical Power and Energy Systems, 30(8): 435-441.
[40] Gegndex, M.E.; Palacios. P.; Lvarez. J.L.(2007). A new self-adaptive crossover operator for real-coded evolutionary algorithms. Lecture Notes in Computer Science, 4431(1):39-48.
[41] Zou, X.F.; Liu.M.Z.(2004). A roubst evolutionary algorithm for constrained multi-objective optimization problems. Computer Research and Development,41(6): 986-990.
[42] Madavan, N.K.(2001). Multiobjective optimization using a Pareto differential evolution approach. Proceedings of 2002 World Congress on Computational Intelligence. Piscataway: IEEE Press, 1145-1150.
[43] Runarason, T.; Yao, X.(2000). Stochastic ranking for constrained evolutionary optimization. IEEE transactions on Evolutionary Computation, 4(3): 284-294.

3

Genetic Algorithm Based on Schemata Theory

Eisuke Kita and Takashi Maruyama
Graduate School of Information Science, Nagoya University
Japan

1. Introduction

In actual complicated optimization problems, it is often difficult to find a global optimum solution in admissible computing time. Therefore, in industrial problems, we should find quasi-optimum solution in admissible computing time. In that case, evolutionary computations (ECs) are very attractive.
There are several algorithms in the EC family; genetic algorithm (GA), evolutionary strategy (ES), genetic programming (GP), and so on(4; 5; 10). Genetic algorithm (GA) has been firstly presented by J.Holland in 1975(5). The GA, which is the algorithm to mimic the natural evolution, is widely applied to optimization, adaptation and learning problems. The basic algorithm of the GA is often called as simple genetic algorithm (SGA)(4). Many improved algorithms are derived from the SGA. The search performance of the SGA can be discussed from the viewpoints of the early convergence and the evolutionary stagnation(2; 10). The early convergence means that all individuals are rapidly attracted to a local optimum solution and therefore, the global optimum solution cannot be found. The evolutionary stagnation means that the convergence speed becomes slower as the iterative process goes. Once a quasi-optimal solution is found, it is generally difficult for the SGA to find better ones. For overcoming these difficulties, Sato et.al. has presented Minimal Generation Gap (MGG)(8). The application of GA with MGG to several actual problems reveals that the GA with MGG is very effective for actual optimization problems(6).
Stochastic Schemata Exploiter (SSE) is also classified into the ECs(1). Although the basic concept of SSE comes from the GA, its algorithm is very different from GA. In GA, the individuals are generated randomly in order to construct a population. After estimating the fitness of individuals, parents are selected from the population according to the fitness value. Offspring are generated from the parents by using genetic operators such as the mutation, the crossover, and so on. SSE algorithm also starts from the population of randomly generated individuals. After estimating the fitness of individuals, sub-populations are generated from the whole population according to semi-order relationship of the sub-populations. Common schemata are extracted from the sub-populations and offspring are generated from the common schemata. The SSE has two attractive features. Firstly, the SSE convergence speed is faster than the SGA because SSE can spread better schemata over the whole population faster than the GA. Secondly, there are very small number of control parameters which has to be defined by users in advance. Since the selection and crossover operators are not necessary in SSE, the control parameters are only population size and mutation rate. However, SSE sometimes converges to not global optimum solution but local one.

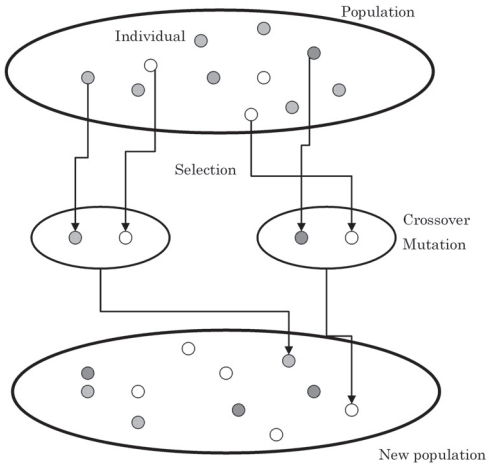


Fig. 1. Simple genetic algorithm (SGA)

The aim of this study is to improve the search performance of SSE without sacrificing the convergence speed. For this purpose, we introduce in this chapter Extended Schemata Exploiter (ESSE) and cross-generational elitist selection SSE (cSSE). In the ESSE, once the common schemata list is defined from the common schemata which are extracted from the individuals in the sub-populations, the list is modified by deleting individual schemata, updating similar schemata and so on. In the cSSE, the cross generational elitist selection(3) is introduced to the original SSE. In the numerical examples, SSE, ESSE and cSSE are compared with genetic algorithm (GA) with minimum generation gap (MGG) and Bayesian Optimization Algorithm (BOA).
The remaining of the chapter is organized as follows. Algorithms of GA, SSE, ESSE, cSSE and BOA are compared briefly in section 2. In sections 3 and 4, the SSE, ESSE and cSSE algorithms are described minutely. Numerical examples are shown in section 5. Some conclusions are summarized again in section 6.

2. Back ground

2.1 Genetic algorithm

Genetic algorithm (GA) was firstly presented by J.Holland in 1975(5). The most fundamental genetic algorithm is often called as simple genetic algorithm (SGA)(4). SGA is illustrated in Fig.1.

1. Construct an initial population by individuals with randomly defined chromosomes.
2. Estimate individuals fitness.
3. Select parents from the population according to the roulette selection of the fitness value.
4. Generate two offspring from the parents by one-point crossover operator.
5. Apply mutation operator to all offspring.
6. Go to step 2 unless convergence criterion is satisfied.

SGA has two disadvantages; the early convergence and evolutionary stagnation(2; 10). The early convergence means that all individuals in the population converge to same local

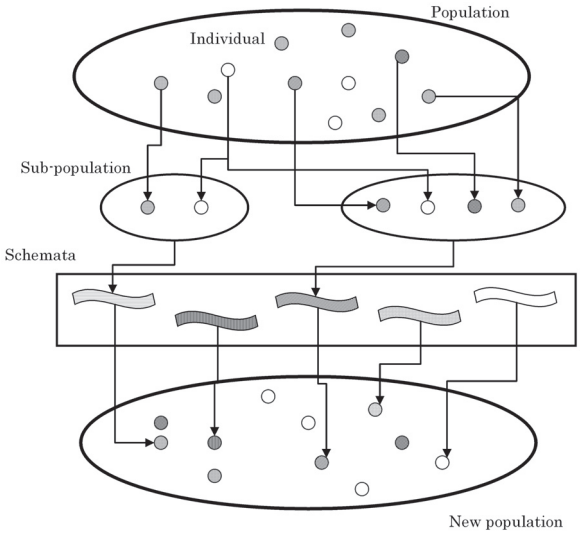


Fig. 2. Stochastic schemata exploiter (SSE)

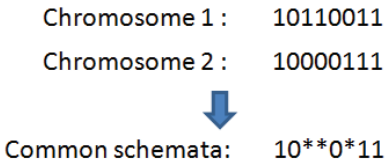


Fig. 3. Example of chromosome and common schema

optimum solutions at early generation and therefore, the global (real) optimum solution cannot be found. The evolutionary stagnation means that the convergence speed slows down at final generations. Minimal Generation Gap (MGG) was presented for overcoming these problems(8). Several numerical results show that the GA with MGG can find better global solutions although the convergence speed is sacrificed(6). The process of the GA with JMMG is summarized as follows.

1. Construct an initial population by individuals with randomly defined chromosomes.
2. Select parents from the population randomly.
3. Generate two offspring from the parents by one-point crossover operator.
4. Estimate fitness of two parents and two offspring.
5. Select the best individual among them.
6. Select the other individual than the best one among them randomly.
7. Replace two selected individuals with parents.
8. Go to step 2 unless convergence criterion is satisfied.

2.2 Stochastic schemata exploiter

The SGA search process can be explained according to the schemata theory. The SGA final goal is to find the set of 0′ and 1′s which represents the optimal solution of the function.

Common set of the individual chromosomes is named as common schemata (Fig3). In the common schema, the uncommon '0's and '1's for all chromosomes are replace with '*'s. As the SGA search process goes, common schemata of the better individuals develops to the binary representation of an optimal solution. If the developing speed of the common schemata can be accelerated, the SGA search performance must be improved well. This is the basic concept of Stochastic Schemata Exploiter (SSE).
The algorithm of SSE is illustrated in Fig.2.

1. Construct an initial population by individuals with randomly defined chromosomes.
2. Estimate individual fitness.
3. Rank individuals according to the descending order of their fitness.
4. Generate sub-populations according to individual rank.
5. Extract common schemata from the individuals in each sub-population.
6. Generate offspring from the extracted schemata.
7. Go to step 2 unless convergence criterion is satisfied.

While SGA generates offspring from parents (individuals), SSE generates from the common schemata extracted from better individuals. Therefore, SSE can accelerate the developing speed of the better common schemata.
The SSE algorithm is described minutely in section 3.

2.3 Extended stochastic schemata exploiter

SSE extracts the common schemata from better individuals. The extracted common schemata are very often identical or very similar schemata. If the identical or very similar common schemata are deleted from the common schemata list, a wider variety of individuals can be generated from the list. This is the basic idea of the extended stochastic schemata exploiter (ESSE).
When two schemata are selected from the list, their similarities are classified as follows:

- **Case 1:** two schemata are identical,
- **Case 2:** one schema is included into the other one,
- **Case 3:** they are partially identical, and
- **Case 4:** they are different.

ESSE operations 1,2 and 3 are defined for the case 1, 2 and 3, respectively.
The ESSE algorithm is composed of the original SSE algorithm and one or more of the above ESSE operations. The ESSE algorithm is illustrated in Fig.4. The different process against SSE is to update the common schemata list by applying the ESSE operations.
The ESSE algorithm is described minutely in section 4.1.

2.4 Cross generational elitist selection SSE

ESSE updates the common schemata list by deleting the identical or similar common schemata from the list in order to improve the search performance. The computational cost for updating the list, however, is relatively expensive.
The aim of cross generational elitist selection SSE (cSSE) is to reduce the computational cost without sacrificing the ESSE search performance. For the purpose, instead of update of the common schemata list in ESSE, the cSSE adopts cross-generational elitist selection(3) and exclusion of identical individuals from a population.
The cSSE algorithm is illustrated in Fig.5. The different process against ESSE is to select better individuals alone from the large population composed of all parents and all offspring in order to define a new population.

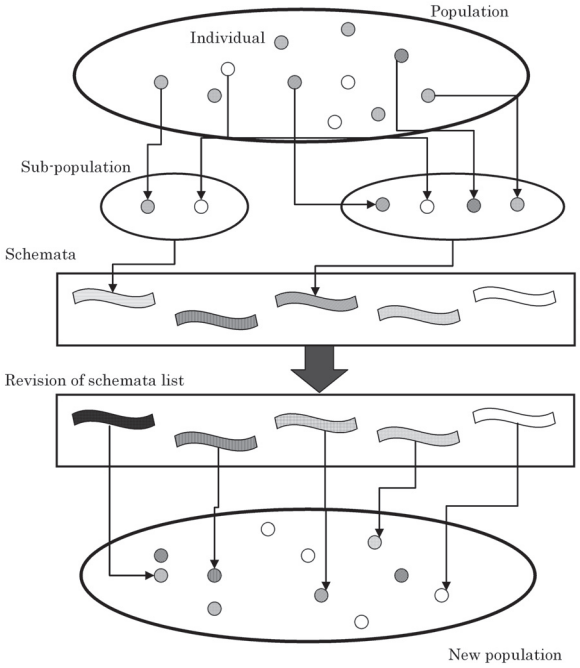


Fig. 4. Extended stochastic schemata exploiter (ESSE)

The cSSE algorithm is described minutely in section 4.2.

2.5 Bayesian optimization algorithm

Estimation of Distribution Algorithm (EDA) is also one of evolutionary computations. The EDA searches a solution according to stochastic model learned from the information of the better solutions in the population. Since offspring are generated from the stochastic model, the selection, the crossover, and the mutation operations are not necessary in EDA.
Bayesian Optimization Algorithm (BOA), which is one of the EDA, was presented by Pelikan et. al.(7). In BOA, the Bayesian network plays as the stochastic model of EDA. The convergence speed of BOA is much faster than that of SGA. In this study, BOA is adopted for confirming the convergence speed of SSE, ESSE and cSSE.

3. Stochastic schemata exploiter

3.1 SSE algorithm

We would like to explain again the process of the stochastic schemata exploiter (SSE).

1. Construct an initial population with randomly generating M individuals.
2. Estimate individual fitness
3. Rank individuals according to the descending order of their fitness.
4. Define M sub-populations according to individual rank.
5. Extract common schemata from individuals in M sub-populations.
6. Generate M offspring from the extracted schemata.

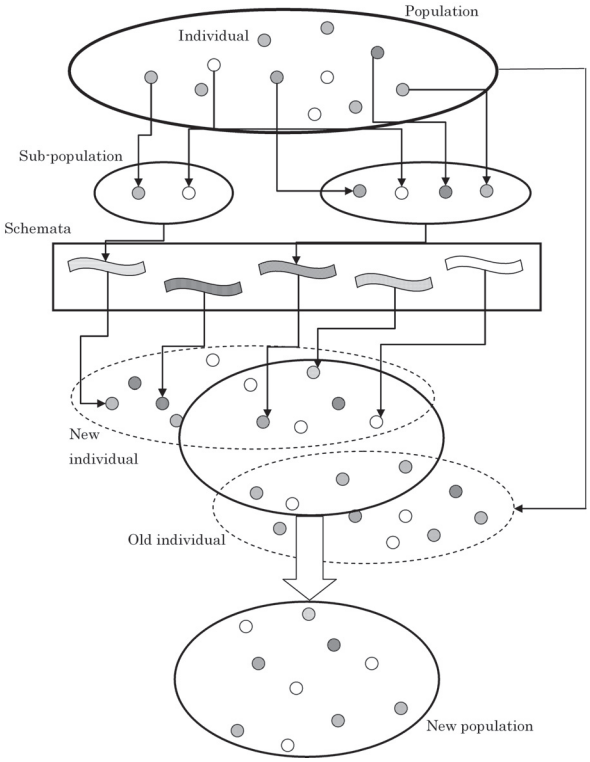


Fig. 5. Cross generational elitist selection stochastic schemata exploiter (cSSE)

7. Go to step 2 unless convergence criterion is satisfied.

The particular processes in the SSE are defining sub-populations, extracting common schemata, and generating new individuals. So, we would like to explain them in the followings.

3.2 Defining sub-populations

The sub-populations are generated according to the semi-order relation between the sub-populations. In the followings, we will explain the semi-order relation and then, how to define sub-populations.

3.2.1 Semi-order relation

The population P is composed of the individuals $c_1, c_2, \cdots$ and c_M, which are numbered according to the descending order of their fitness function. Therefore, the individual c_k denotes the $k-$th best individuals in the population P. The symbol S denotes the sub-population of the population P. When the individual c_k is excluded from S, a new population is represented as $S - c_k$. The operator $\cup$ denotes the union of sets.
When the worst individual in the sub-population S is numbered as $L(S)$, the following semi-order relation is held in the sub-populations of the population P.

1. Since the individual $c_{(L(S))}$ is the worst one in the sub-population S, the individual $c_{(L(S)+1)}$ is worse by one order than the individual $c_{(L(S))}$. When the individual $c_{(L(S)+1)}$ is added to a sub-population S, the new sub-population is defined as $S \cup c_{(L(S)+1)}$. A first semi-order relation is given as

$$f(S) \geq f(S \cup c_{(L(S)+1)}) \tag{1}$$

   where $f(S)$ denotes the average fitness of the individuals in the sub-population S.

2. When the individual $c_{(L(S))}$ is replaced with the individual $c_{(L(S)+1)}$, the new population is defined as $(S - c_{L(S)}) \cup c_{(L(S)+1)}$. A second semi-order relation is given as

$$f(S) \geq f((S - c_{L(S)}) \cup c_{(L(S)+1)}) \tag{2}$$

3.2.2 Sub-population

The use of semi-order relation gives the following order of sub-populations.
Since it is obvious that the best sub-population is composed of the best individual c_1 alone, we have a first sub-population

$$S_1 = \{c_1\}.$$

When adding the individual c_2 to the sub-population $S_1 = \{c_1\}$ according to the semi-order relation (1), we have a second sub-population

$$S_2 = \{c_1, c_2\}.$$

When replacing the individual c_1 in the sub-population S_1 with the individual c_2 according to the semi-order relation (2), we have a third sub-population

$$S_3 = \{c_2\}.$$

When adding the individual c_3 to the sub-population $S_2 = \{c_1, c_2\}$ according to the semi-order relation (1), we have a fourth sub-population

$$S_4 = \{c_1, c_2, c_3\}$$

When replacing the individual c_2 in the sub-population S_2 with the individual c_3 according to the semi-order relation (2), we have

$$S_4 = \{c_1, c_3\}.$$

We can define the other sub-populations in the similar way.

3.3 Extracting common schemata

After defining the sub-populations, the common schemata are extracted as the common set of the chromosome of the individuals in the sub-populations (Fig.3).

3.4 Generating new individuals

The extracted schemata are composed of three characters; “0”, “1”, and “$*$”. So, the new individuals are defined by randomly replacing “$*$” by “0” or “1” (Fig.3).

4. Extended SSE and cross generational elitist selection SSE

4.1 Extended SSE (ESSE)

4.1.1 Algorithm

The ESSE algorithm is illustrated in Fig.4 and summarized as follows.

1. Construct an initial population by individuals with randomly defined chromosomes.
2. Estimate individual fitness.
3. Rank individuals according to the descending order of their fitness.
4. Generate sub-populations according to individual rank.
5. Extract common schemata from the individuals in sub-populations and register them to the common schemata list.
6. Update the common schemata list by applying the ESSE operations.
7. Generate offspring from the extracted schemata.
8. Go to step 2 unless convergence criterion is satisfied.

4.1.2 ESSE operations

The SSE algorithm often generates identical or very similar schemata from different sub-populations. Extended Stochastic Schemata Exploiter (ESSE) updates the common schemata list by applying the ESSE operations.
When the schema A is extracted from the sub-population S_A, the fitness of the schema A is defined as the average fitness of all individuals in the sub-population S_A, which is referred to as $f(S_A)$.
The following operations are applied for two schemata A and B.

4.1.2.1 ESSE Operation 1

If the schemata A and B are identical, the following processes are performed.

1. A is kept and B is deleted from the schemata list.
2. A common schema is extracted from $S_A \cup S_B$.

4.1.2.2 ESSE Operation 2

If the schema A is included into the schema B, it is considered that the schema B is grown up from the schema A. In this case, the following process is performed.

1. If $f(S_A) > f(S_B)$, A is kept and the common schema is extracted from $S_A \cup S_B$.
2. If $f(S_A) \leq f(S_B)$, B is kept and the common schema is extracted from $S_A \cup S_B$.

4.1.2.3 ESSE Operation 3

If the schema A and B are partially identical, the following processes are performed.

1. If $f(S_A) > f(S_B)$, A is kept. If not so, B is kept.
2. A common schema is extracted from $S_A \cup S_B$.

4.1.3 ESSE family

The ESSE is composed of the original SSE and one or more ESSE operations. So, we can define the seven ESSE algorithms according to the selection of ESSE operations. They are named as $c_1, c_2, \cdots$ and c_7.

c1: Original SSE with ESSE operation 1.
c2: Original SSE with ESSE operations 1 and 2.
c3: Original SSE with ESSE operation 2.

c4: Original SSE with ESSE operations 2 and 3.
c5: Original SSE with ESSE operation 3.
c6: Original SSE with ESSE operations 1 and 3.
c7: Original SSE with ESSE operations 1, 2 and 3.

4.2 Cross generational elitist selection SSE (cSSE)

4.2.1 Cross generational elitist selection

The cross generational elitist selection was presented by Eshelman(3). Algorithm of cross generational elitist selection is summarized as follows.

1. At generation $t-1$, offspring are generated from individuals in the population.
2. Populations of parents and offspring are referred to as $P(t-1)$ and $O(t-1)$, respectively.
3. $P(t-1)$ and $O(t-1)$ are merged to new population $P'(t-1)$. When the sizes of $P(t-1)$ and $O(t-1)$ are M, the size of $P'(t-1)$ is $2M$.
4. Individuals in $P'(t-1)$ are ranked according to their fitness.
5. The population $P(t)$ is generated by selecting M best individuals from $P'(t-1)$.

The original algorithm of cross-generational elitist selection allow multiple identical individuals to exist in a population. However, the algorithm of cross generational elitist selection in cSSE is modified so that identical individuals are deleted from the population.

4.2.2 Algorithm

The cSSE algorithm is illustrated in Fig.5 and summarized as follows.

1. Construct an initial population by individuals with M randomly defined chromosomes.
2. Estimate individual fitness.
3. Rank M individuals according to the descending order of their fitness.
4. Generate M sub-populations according to individual ranks.
5. Extract common schemata from the individuals in M sub-populations.
6. Generate M offspring from M extracted schemata.
7. Delete identical individuals from $2M$ individuals in the population composed of M parents and M offspring.
8. Select M better individuals from the remaining individuals to define new population.
9. Go to step 2 unless convergence criterion is satisfied.

In the above process, the steps 7 and 8 correspond to the cross generational elitist selection.

5. Numerical example

5.1 Test Problems

We would like to compare the algorithm performance in deception and knapsack problems(9).

5.1.1 Deception problem

The deception problem is defined as the summation of the 4-bit deception problems(9). The 4-bit deception problem is shown in Table 1. The objective function and the design variable of the problem is defined as

$$f_{deception} = \sum_{i=1}^{n} f_d(x_i) \tag{3}$$

$$x_i \in 0000, 0001, \cdots, 1111$$

where n denotes the number of 4-bit deception problem and $n = 10$.

$$f_d(1111) = 30 \; f_d(0000) = 28 \; f_d(0001) = 26 \; f_d(0010) = 24$$
$$f_d(0100) = 22 \; f_d(1000) = 20 \; f_d(0011) = 18 \; f_d(0101) = 16$$
$$f_d(0110) = 14 \; f_d(1001) = 12 \; f_d(1010) = 10 \; f_d(1100) = 8$$
$$f_d(1110) = 6 \quad f_d(1101) = 4 \quad f_d(1011) = 2 \quad f_d(0111) = 0$$

Table 1. Part solutions of deceptive problem

N_i	10	50	100
c1	298.24	300.00	300.00
c2	298.92	300.00	300.00
c3	297.32	300.00	300.00
c4	297.84	299.56	299.12
c5	298.12	300.00	300.00
c6	297.64	300.00	300.00
c7	291.40	290.52	290.80

Table 2. Final solutions on deception problem

N_i	10	50	100
c1	16107.1	16150.2	16153.5
c2	16094.2	16149.5	16154.0
c3	16066.3	16135.7	16134.3
c4	16074.4	16121.3	16128.5
c5	16090.9	16142.9	16147.7
c6	16103.5	16148.1	16153.7
c7	15738.6	15644.9	15663.4

Table 3. Final solutions on knapsack problem

5.1.2 Knapsack problem

When there are n bag gages in a knapsack, the knapsack problem is defined as the maximization of the value of the knapsack without exceeding the weight limit b. The problem is defined as

$$\begin{aligned} &\max_{\{x_i\}} \sum_{i=1}^{n} c_i x_i \\ &\text{subject to} \sum_{i=1}^{n} a_i x_i \leq b \\ &x_i \in 0, 1 \qquad (i = 1, \cdots, n) \end{aligned} \tag{4}$$

where the weight and the value of the bag i are referred to as a_i and c_i, respectively, which are randomly taken within $1 \leq a_i, c_i \leq 100$. Besides, $b = 10000$ and $n = 400$.

5.2 ESSE performance evaluation

A two-point crossover is adopted for the SGA and GA with MGG. The crossover rate is 1 (100%). A maximum number of the generation is 40,000 in the deception problem and 10,000 in the knapsack problem. The population size is $n_i = 10, 50$ or 100. Fifty simulations are performed for each problem from the different initial populations. Their average values are shown in figures.

5.2.1 Comparison of ESSE family

We notice from Tables 2 and 3 that the ESSE-c1, c2 and c6 have the better search performance than the others. The results show that ESSE operation 1 is effective because ESSE-c1, c2 and c6 have the ESSE operation 1. When focusing the ESSE-c2 which has ESSE operation 1 and 2, we notice that it has good search performance at the deception problem with $n_i = 10$ and the knapsack problem with $n_i = 100$. Therefore, the effectiveness of the ESSE operations 2 and 3 may depend on the problem to be solved.

5.2.2 Comparison of ESSE, SSE, SGA, and GA with MGG

The results in the previous section show that the ESSE-c1 is the best among the ESSE family. The ESSE-c1 is compared with SGA, GA with MGG and SSE.

5.2.2.1 Deception problem

Convergence history of the best individual fitness is shown in Fig.6. The figure is plotted with the generation as a horizontal axis and the best individual fitness as a vertical axis, respectively. Numbers following to the algorithm name, in the figures, denote the population size n_i. The population size is determined from numerical experiments as the best value for each algorithm. We notice from figure 6 that the convergence speed of SSE and ESSE-c1 are faster than SGA and GA with MGG and specially, that the ESSE-c1 is the fastest among them. The ESSE-c1 is compared with the SSE in the different population sizes. Convergence history of the average value of the best individuals at every iterations is shown in Fig.7. In case of the large population size (50 and 100 individuals), the convergence speed of the c1 is faster than the SSE.

5.2.2.2 Knapsack problem

Convergence history of the best individual fitness is shown in Fig.8. Figure is plotted with the generation as a horizontal axis and the best individual fitness as a vertical axis, respectively. We notice the similar results as the deception problem from the figures; i.e., the convergence speed of SSE and ESSE-c1 is faster than the others.
The ESSE-c1 is compared with the SSE in the different population sizes. Convergence history of the best individual fitness is shown in Fig.9. In all population sizes, the ESSE-c1 shows faster convergence speed than the SSE. Specially, the ESSE-c1 with 50 individuals can find slightly better solution than the SSE with 100 individuals.

5.3 cSSE Performance evaluation

We would like to compare cSSE with GA with MGG, BOA and SSE.
In GA with MGG, two-point crossover of crossover rate $= 1$ is employed and all individuals are replaced at every generations. In all algorithms, the best mutation rates are determined from numerical experiments.
Maximum generation is 40,000 for deception problem and 15,000 for knapsack problem, respectively. Population size is specified as $N_i = 10, 50, 100$ or 250 for GA with MGG, SSE and cSSE and $n_i = 20, 100, 200$ or 500 for BOA, respectively. Since BOA replaces half population size at every generations, computational cost of fitness function is half as much as the other algorithms. For equalizing the computational cost of all algorithms, the population size of BOA is twice as many as the other algorithms. Simulations are performed 50 times from different initial populations. The average values of the best individual fitness are shown.

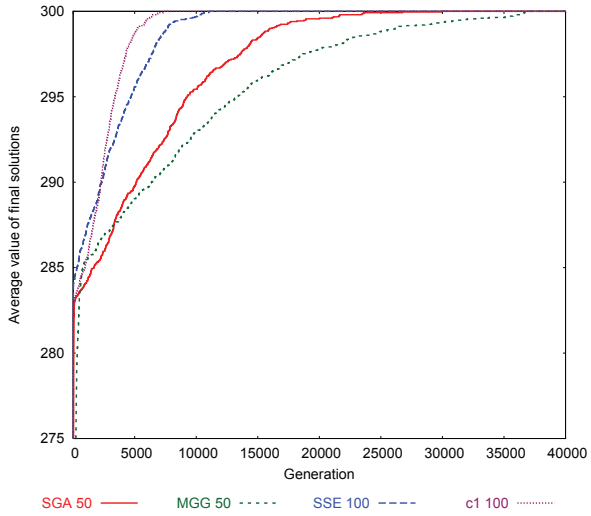


Fig. 6. Comparison of SGA, GA with MGG, SSE and ESSE-c1 in deception problem

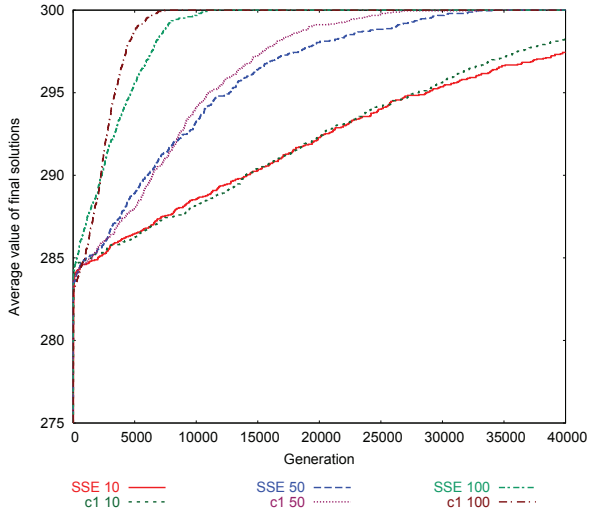


Fig. 7. Comparison of SSE and ESSE-c1 in deception problem

N_i (N_i for BOA)	GA(MGG)	BOA	SSE	cSSE
10 (20)	571.2	533.4	570.0	573.0
50 (100)	574.9	560.4	575.2	588.3
100 (200)	576.1	560.0	586.9	594.6
250 (500)	575.8	560.0	590.7	597.1

Table 4. Average values of final solutions (Deception problem)

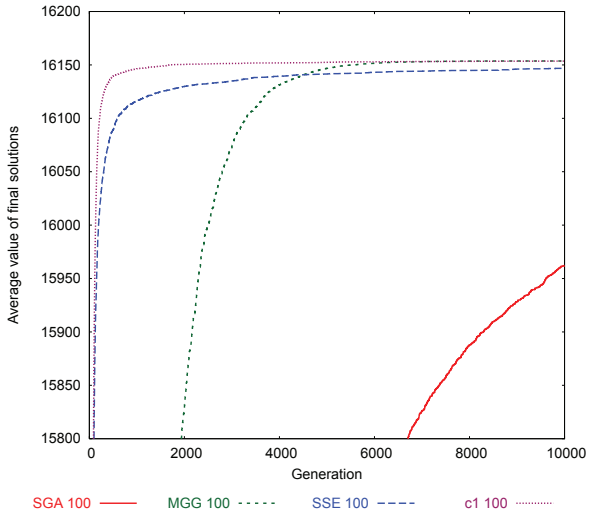


Fig. 8. Comparison of SGA, GA with MGG, SSE and ESSE-c1 in knapsack problem

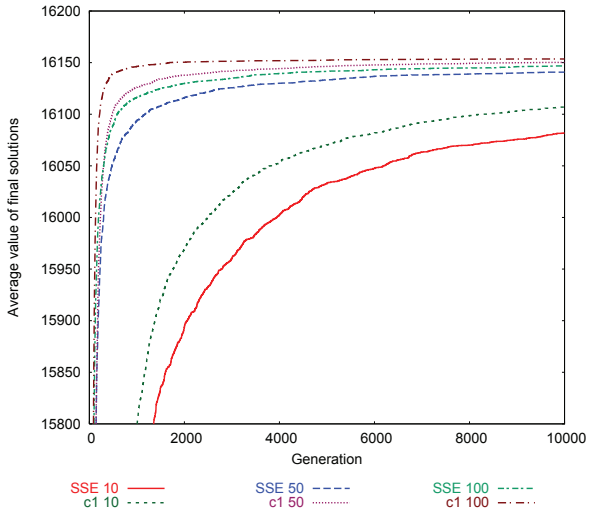


Fig. 9. Comparison of SSE and ESSE-c1 in knapsack problem

5.3.1 Deception problem

Convergence history of the best individual fitness is shown in Fig.10. Figure is plotted with the generation as the horizontal axis and the fitness value as the vertical axis, respectively. Table 2 shows the best individual fitness at final generation.

We notice from Table 4 that the final cSSE solution is the best among them for all cases of population size. Figure 10 illustrates that the cSSE is the fastest among them. Although BOA

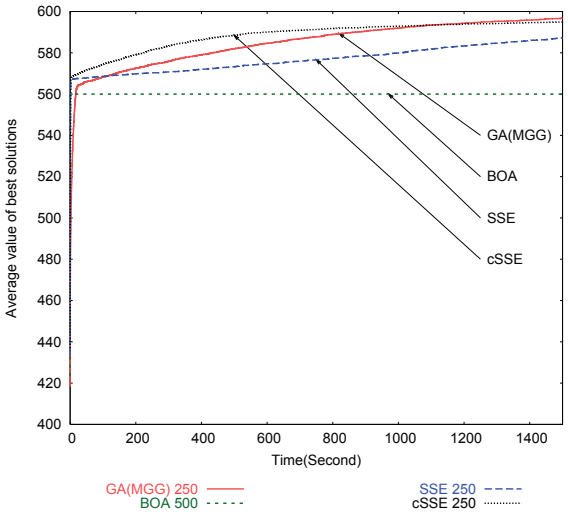


Fig. 10. Comparison of GA with MGG, BOA, SSE and cSSE in deception problem

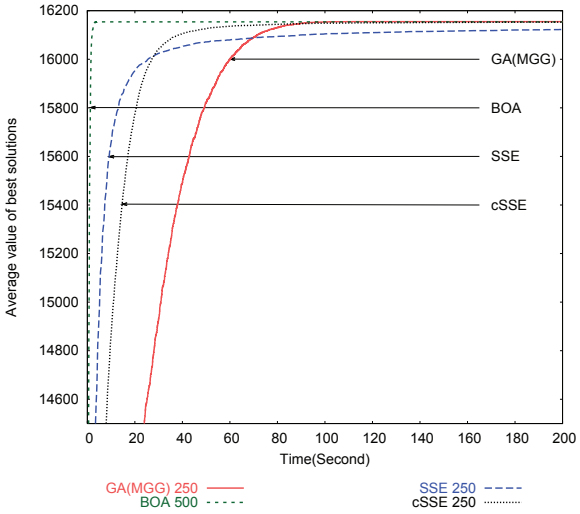


Fig. 11. Comparison of GA with MGG, BOA, SSE and cSSE in knapsack problem

has very fast convergence speed, it may be attracted to a local optimum solution because the final BOA solution is worse than the other.

5.3.2 Knapsack problem

Convergence history of the best individual fitness is shown in Fig.11. We notice that the BOA outperforms the SSE and the cSSE and that the BOA is the best among them from the view-point of both the computational time and the search performance. Remember that the convergence speed of the BOA is generally faster than the GA, SSE and cSSE. Therefore, the

N_i (N_i for BOA)	GA(MGG)	BOA	SSE	cSSE
10 (20)	16060.7	12754.9	16104.4	16141.0
50 (100)	16141.9	15829.3	16142.8	16154.1
100 (200)	16153.8	16111.9	16148.3	16154.8
250 (500)	16155.0	16153.9	16149.5	16155.0

Table 5. Average values of final solutions (Knapsack problem)

above results may denote that the solution space of the Knapsack problem is relatively simple and that there is only one optimal solution in the space. Table 5 shows the fitness value of the best solution at the final generation.
We notice from Table 5 that the final cSSE solution is the best among them. Note that the final cSSE solution at $N_i = 50$ is better than GA solution at $N_i = 100$, BOA solution at $N_i = 500$, and SSE at $N_i = 250$. Figure 11 illustrates that the convergence speed of MGG is the slowest among them.

6. Conclusions

In this chapter, we described stochastic schemata exploiter (SSE) and its improved algorithms such as Extended SSE (ESSE) and cross-generational elitist selection SSE (cSSE).
First, we compared seven ESSE algorithms in test problems. The ESSE-c1 algorithm, which was composed of SSE and ESSE operation 1, showed better search performance than the other ESSE algorithms in the test problems. We compared the ESSE-c1 with SGA, GA with MGG and SSE. The convergence speed of SSE and ESSE-c1 is faster than the others. In some cases, the speed of the ESSE-c1 is faster than the original SSE.
Next, we compared the cSSE with the other algorithms. From the view point of search performance, we notice that cSSE can find slightly or much better solution than the others in all examples. In comparing the convergence speed of algorithms, we notice that the cSSE as well as BOA is fastest among them.

7. References

[1] N. A. Aizawa. Evolving SSE: A stochastic schemata explointer. In *Proc. 1st IEE Conf. Evol. Comp.*, pages 525–529. IEEE, 1994.
[2] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, 1987.
[3] L. J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms 1991 (FOGA 1)*, pages 265–283. Morgan Kaufmann., 1991.
[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1 edition, 1989.
[5] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1 edition, 1975.
[6] I. Ono, S. Kobayashi, and K. Yoshida. Global and multi-objective optimization for lens design by real-coded genetic algorithms. *International Optical Design Conference*, 3482:110–121, 1998.
[7] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz. Boa: The bayesian optimization algorithm. In W. Banzhaf, Daida J, A. E. Elben, M. H. Garzon, V. Honavar, M. Jakiela,

and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-1999, San Fransisco, CA)*, pages 525–532. Morgan Kaufmann., 1999.

[8] H. Satoh, I. Ono, and S. Kobayashi. Minimal generation gap model for gas considering both exploration and exploitation. *Proc. IIZUKA'96*, pages 494–497, 1996.

[9] L. D. Whitley. Fundamental principles of deception in genetic search. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms 1991 (FOGA 1)*, pages 221–241. Morgan Kaufmann, 1991.

[10] D. Whitly. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive traials in best. In J. D. Shafer, editor, *Proc. 3rd Int. Conf. Genetic Algorithm*, pages 116–121. Morgan Kaufmann Pub., 1989.

4

In Vitro Fertilization Genetic Algorithm

Celso G. Camilo-Junior[1] and Keiji Yamanaka[2]
[1]*Informatics Institute, Federal University of Goias*
[2]*School of Electrical Engineering, Federal University of Uberlandia*
Brazil

1. Introduction

Some techniques are applied to the optimization problems. However, a few achieve satisfactory performance when the problem is complex, for example, multimodal or multiobjective. The Mathematical Programming algorithms which use the gradient as a search guide have difficulty and do not often reach the optimum in multimodal problems. The metaheuristics, on the other hand, don't ensure the global optimum but they have good results and, therefore, are quite used in these scenarios.

Metaheuristics could be classified by the number of potential solutions that are used: solution-based metaheuristics, such as Hill-Climbing, Simulated Annealing and Tabu search; population-based metaheuristics, such as the Evolutionary Algorithms; or hybrids metaheuristics. The population-based algorithms start the search with several points in the search space and, through the interactions of among points, try to find a new point with the highest value of the objective function. This strategy, therefore, explores the search space in several places simultaneously (Oriented Exploration). The solution-based algorithms are based on a single point to exploit the search space (Oriented Exploitation). They often use techniques to escape from the local optimum. The hybrid strategy tries to join the intensification and diversification to improve both quality of computed solutions and the robustness of solvers. All strategies show good results depending on the approached problem. As an example of metaheuristic, we can cite: Simulated Anneling, Tabu Search, Grasp, VND, VNS and Ant Colony. Among them, Evolutionary Algorithms, especially the Genetic Algorithms, show excellent results and thus they are one of the most popular among researchers.

Genetic algorithms are optimization methods inspired by the mechanisms of alive beings' evolution Goldberg (1989). The algorithms based on this technique follow the principle of natural selection and survival of the fittest (Charles Darwin).

One of the genetic algorithm advantages is the simplification in the formulation and solution of optimization problems. Simple GAs usually work with coded descriptions formed by bits strings of fixed size. Other types of GAs can work with bits strings of variable size, e.g. GAs used for Genetic Programming FERREIRA (2001a;b); RODRIGUES (2003).

The GA is indicated for the solution of complex optimization problems, such as the real problems, that involve a large number of variables and, consequently, high dimensional search spaces. Moreover, in many cases where other optimization strategies fail in finding a solution, GAs are good options. However, in some cases, the performance is not satisfactory GEN & CHENG (1997); MICHALEWICZ (1996). Therefore, some works are developed with the aim to improve the performance CHAINATE & PONGCHAROEN (2007); MUSIL et al.

(1999); PARK et al. (2000); RAJAN et al. (2002); RONG-LONG & KOZO (2005); RUTTKAY et al. (1995); WU et al. (2004); YANG & DOUGLAS (1998); YEN et al. (1998). Some works focus in the convergence speed, though, most analyzes the efficacy. The preservation of genetic diversity to avoid premature convergence is a frequently researched topic MAHFOUD (1992; 1995); SHIMODAIRA (2002); TACKETT & CARMI (1994).
Thus, in this paper we propose the In Vitro Fertilization Module (IVFm), to assist the GAs evolution, avoiding the loss of information during evolution. The IVFm is an assistant algorithm that runs in a parallel flow of GA's flow and that recombines chromosomes in order to do the information mining with individuals, which are proceeding from the populations created by GAs or generated by the IVFm operators.
To test the efficacy, defined here as the capacity to reach the global optimum, and the efficiency, defined as the capacity to accelerate the GA evolution, we opted for a benchmark minimization problem that is multimodal with several local optimum. This problem was used to establish the test scenarios.
The results from these three experiments show that the IVFm, especially with the operator EAR-N, has excellent performance and contributes substantially to quick convergence of GA to the global optimum.
The chapter will be organized as follows. One section about canonical GA. One section about the IVFm and operators are described. One section about the premature convergence. One section about the experimental results. Finally, the conclusion and suggestions future works.

2. IVFm: In Vitro Fertilization module

The GAs evolutionary process is composed by a cyclical flow, in which each iteration creates a new generation. For each new generation, individuals are generated and introduced into the population to replace some of the existing ones, which are discarded. However, many of these eliminated individuals contain genes with information that is important for the search. Nevertheless, they do not even go through the reproduction process before being discarded. In other words, they were generated and eliminated and did not contribute to the evolution. Therefore, we can say that in every iteration some information is lost, discarded without analysis.
Although this process mimics the evolution of species, in which an individual could be born and die without generating progeny, it is well-known that there is some loss of information. The information lost may assume two possibilities. It might not go back to the population, causing in some cases a reduction in efficacy and characterizing the loss of opportunity, or it might return by mutation process with low probability. On last case, the efficiency is lower because is needed much time until that information returns and is processed. Therefore, we propose the In Vitro Fertilization Module (IVFm) that recombines the chromosomes from population of GA and new individuals to better exploit the information presented.
Assisted Reproduction is a set of techniques of reproduction where the generation of a life is manipulated and assisted by experts in order to ensure the success of the process. In Vitro Fertilization (IVF) is one of Assisted Reproduction technique that collects egg cells from the ovary of the mother and fertilizes them with prepared sperm from the father, forming the pre-embryos. After manipulation, two pre-embryos are selected and transferred to the mother. The first human baby generated by IVF born in 1978. From 1978 until today, many people have been generated by the IVF.
Analogous to IVF the IVFm assesses various combinations and selects the individual according its quality. IVFm is an algorithm executed in a parallel flow, which receives as input a portion of the GA population and as output it returns an individual, which may be better

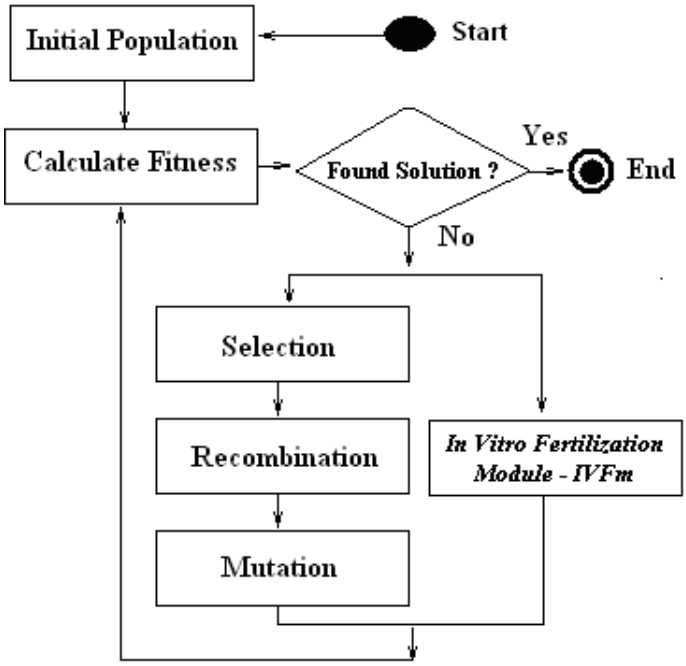


Fig. 1. GA with the IVFm

than the best current (Figure 1). Thus, the IVFm could supply the GA of good individuals, improving and accelerating the GA evolutionary process.

The ideal way to absorb all the information in individuals would be to recombine them gene-gene, until you find the best combination, which would be computationally unfeasible, especially for larger chromosomes. Therefore, we opted to generate children from the recombination of chromosome parts of some individuals with the best one (section 2.4). If the process generates a better individual, this replaces the current best. Otherwise, if the operator does not produce a better individual, there is no interference in the population.

There are two groups of operators for IVFm until this moment. The first one with operators that use as genetic material only the population generated by GA and the second one with operators that alter chromosomes that will be recombined. The second group strategy is to improve the IVFm's population with information that may be beneficial for the recombination process. The operator AR (Assisted Recombination) is a member of the first group and the operators EAR-T, EAR-P and EAR-N are members of the second group.

The IVFm is detailed in the following sections. Section 2.1 describes the execution flow, section 2.2 describes the division of genetic material, section 2.3 shows the operators and section 2.4 describes the process of recombination.

2.1 The IVFm execution flow

At the beginning of the GA execution, the division of the genetic material is defined (section 2.2) as well as the number of individuals that will be used by IVFm (NuIndiv). This configuration is only done in the beginning of the execution of the algorithm.

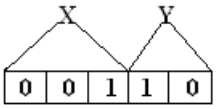


Fig. 2. Two groups selected as the genetic material to exchange

After receiving the current population of GA, each IVFm generation executes the following tasks:

1. Find the fittest individual and label it as Father;
2. Receive as input the following parameters: the N GA individuals (NuIndiv), which are handled in the recombination process;
3. On the EAR strategy, from N selected individuals, the half (N / 2) changes the chromosomes, forming N'. On the AR strategy N' = N;
4. The fittest (father) and the N' individuals are recombined. As there is only one father for each recombination process, the generated children are siblings;
5. If IVFm generates good individuals, they are inserted in the new population, replacing elected individuals, chosen by the elitism process. Without elitism, individuals replace any others aleatorily.

Task 5 above, explains the interference form of IVFm in the population, generated by GA. The interference only happens when good individuals, better than the best available previously, are generated by the proposed algorithm.

2.2 Division of Genetic Material

Before the recombination (exchange of genetic material) the chromosome is divided into groups of genes or gene to gene. This is called the Genetic Material Division to be exchanged. This is an important step of the algorithm, because we believe that knowledge of the problem helps the designer to choose the best way to divide the chromosome, and so, help the algorithm to solve the problem.

One option is to divide the chromosome according to the coded variables, e.g. if the first 3 genes represent the variable x and the past 2 the y, then the chromosome is divided into two groups, the first one containing 3 genes and the other one 2 (Figure 2). Thus, in recombination process the exchanges are among the variable values (decoded values) and not among encoded values. Therefore, this process provides the designer with a tool for knowledge transferring.

2.3 IVFm's operators

Four independent operators are proposed for the IVFm as strategies to help the GA to explore the search space. The AR recombines the chromosomes of the current unchanged population. On the EAR-T, all the genes of some chromosomes suffer mutation before recombination process. On the EAR-P, one group of genes, formed by Genetic Material Division, suffer mutation before recombination and EAR-N generates new individuals to participate in the recombination process. All EAR operators (section 2.3.2) produce some kind of change in population before the recombination, while the AR operator (section 2.3.1) uses the current population without changes as genetic material. The operators are independent and the combined use of these has not been tested yet.

2.3.1 Assisted Recombination operator

Because of the low utilization by the GA of genetic material, we propose the Assisted Recombination (AR) - an operator of the IVFm prepared to better exploit the genetic information in chromosomes of the population generated by the GA. It is hoped, therefore, that the operator contributes to genetic improvement.

The operator is composed of a single phase, called recombination (see details in section 2.4). In this phase, the operator receives part of the GA population and, without making changes to parents, produces children who are analyzed. If the operator finds better individual, this replaces the current best. If the operator does not produce better individuals, there is no interference in the population.

One of the principal AR features is the capability of assisting the recombinations and finding individuals with good fitness, sometimes better than the fittest.

2.3.2 Exploratory Assisted Recombination operators

The Exploratory Assisted Recombination (EAR) operators are based on Assisted Recombination and, therefore, have similarities. For instances, they use the N individuals of the GA population in the improvement process, they have capacity to guide the recombination and they have capacity to identify better individuals than the best current.

Like the Assisted Recombination, the aim of the EAR operators is to better exploit the information produced by GA. The difference of this new class of operators is the ability to make global search through the added exploratory characteristic.

The EAR operators have two phases. The first one, called search space exploration, changes some chromosomes received from the GA and the second one that recombines the chromosomes. These phases are the same to all operators of IVFm (section 2.4).

During the first phase (exploration of search space), the population of N selected individuals is divided in half (N/2) and the individuals of last portion are totally (EAR-T) or partially (EAR-P) changed.

The operator EAR-P applies mutation to the raffled part of the chromosome, divided by the genetic material division. The operator EAR-T applies mutation to all the chromosomes and the EAR-N operator randomly generates new N/2 individuals. After the modifications, the changed individuals replace the N / 2 of the population.

Figure 3 describes the process from the choice of N / 2 individuals until the new changed population (N '), which will be used during the recombination (section 2.4). Is established for this example the division of the genetic material into two parts. The first, with 3 genes and the second with 2 genes. The number of individuals from the GA is 12 (qtdIndv), so the last 6 (N / 2) are changed.

2.4 The recombination

After establishing the groups of genes (division of genetic material) and the population N', the next step is recombination. In this phase, the chromosome is represented by a vector and every element of this is a group of genes, defined by the division of genetic material. This process is similar to the one applied in the Jung's paper Jung (2003).

Analyzing the population N', the best individual is reserved as father and the others as mothers, being the number of mothers limited by the parameter NuIndiv. To create a son, the mother gives an element of its vector and the father supplements the son's chromosome with the other elements of its own vector. This process repeats for all mothers, generating a group of offspring. Other groups are generated by the exchange of the element donated by the mother and the father's chromosome complement, thus, the number of groups of offspring is equal to the number of elements of the father's vector.

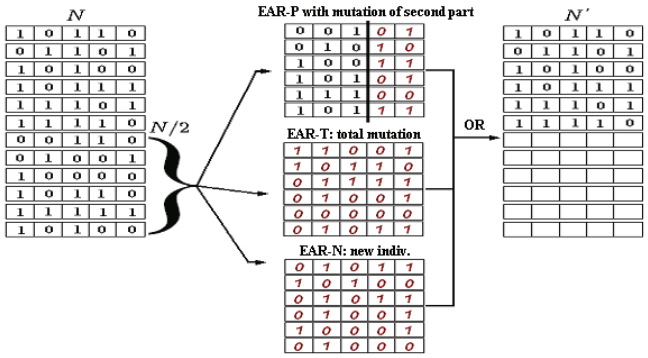


Fig. 3. EAR's exploration process

After the generation of all groups of offspring, the best child is considered the best Super Individual and it is compared to the father. If better, the son replaces the father in the IVFm's population and the recombination process restarts. If not, the algorithm interrupts the loop and inserts the current father in the GA population.
As an example of a recombination iteration, Figure 4 shows:

- The divided chromosome (division of genetic material) into two parts. Group 1 with 3 genes and group 2 with 2 genes;
- Three mothers established by the parameter NuIndiv = 3, in which the mothers are the three individuals after the father;
- The first group of offspring generated by the donation of the first element of the mother vector and a complement of the father's. For example, the second child from the "Sons of Group 1" (01110) is the result of the union of the first element of the second mother (011) and of the father's complement, i.e. second element of the father vector (10);
- The second group of offspring. In this group, the mother donates the second vector element and the father completes it with its first vector element. For example, the third child from the "Sons of Group 2" (00100) is the result of the union of the second element of the third mother (00) and of the father's complement, i.e. first element of the father's vector (001). In this case, contrary to the "Sons of Group 1", the child is formed by the father's first part and the mother's second part;
- The best individual from the group of offspring is considered the super individual of the group. The best among the super individuals is compared to the father. If it is better than the father, the best super individual replaces the father in the next iteration. If not, the algorithm interrupts the loop and returns the current father.

3. The premature convergence

One of the main features of the IVFm is the ability to better exploit the information present in the population (information mining), generating good individuals who can improve and accelerate the genetic evolution.
However, when we try to accelerate the GA's genetic evolution a problem arise, the premature convergence Goldberg (1989); SULTAN et al. (2007), defined as a fast loss of genetic diversity in the population that harms the final solution. Despite being considered a problem when

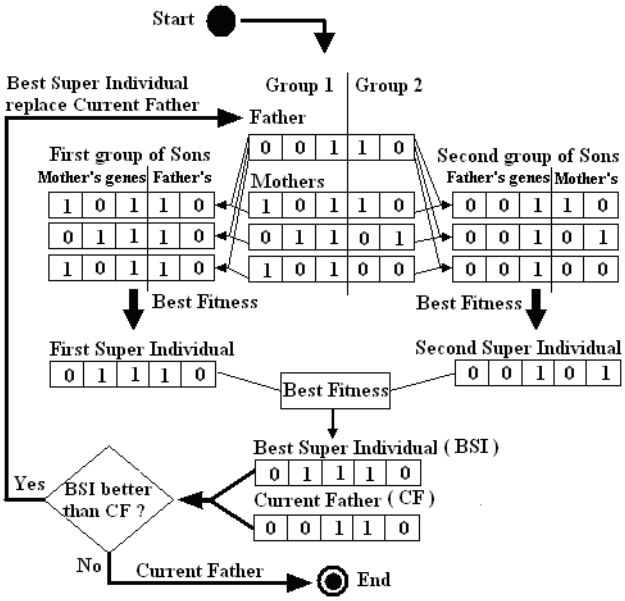


Fig. 4. Example of recombination process

the convergence is to a local optimum, the fast convergence is desirable if it is to the global optimum.

The discussion about premature convergence goes through other concepts such as exploration, global exploration of search space, and exploitation, local exploitation of the search space EIBEN & SCHIPPERS (1998); Spears (1992). Thus, we can define premature convergence as little exploration and premature exploitation. Usually, the algorithms that use both strategies apply the global search at the beginning to identify promising regions, and local search later to find the best solution in the good regions.

The GA is an algorithm which begins its execution focusing on exploration strategy, when their genetic diversity is high, and at a certain time it focuses in the exploitation, when a representative portion of individuals have similarities.

To avoid premature convergence to local optimum, it is important to make a good exploration. In the Canonical GA's case, the exploration is done by mutation, with a low probability, and by the recombination of solutions generated by the beginning of the evolutionary process, when the population usually has high genetic diversity. However, this process is slow, considering that several generations are needed to prioritize one of the promising regions.

Allowing the GA to have a better use of its individuals (information mining), we believed that the genetic evolution can be improved without increasing the probability of the premature convergence.

4. Experiments

To test the performance of the proposed algorithm (IVFm) and its operators, we chose a benchmark multimodal problem. The Rastrigin's function (Equation 1) is widely used and one of the hardest benchmark because of multimodality and its behavior with many peaks and valleys, featuring several local optimum. Figure 5 shows the function.

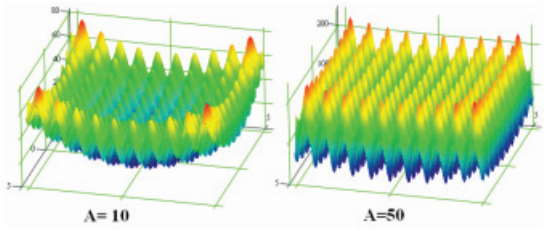


Fig. 5. Bidimensional Rastrigin function with A=10 and A=50

$$f(x) = nA + \sum_{i=1}^{n}(x_i^2 - A\cos(2\pi x_i)); \forall i \in [1..n], x_i \in [-5.12, 5.12] \quad (1)$$

Given the function, the objective is to minimize yet still respecting the restrictions: $-5.12 \leq X_i \leq 5.12$ and values with four decimal places. Considering the restrictions, the minimum point of the function is reached when Xi = 0. At this point the solution is global optimum, generating a $f(X_i) = 0$.
This work presents 3 experiments. In the first experiment, the objective is to examine the efficacy of the IVFm to minimize (Scenario 1.1) and its behavior when the parameter A is changed (Scenario 1.2). In the second one, the aim is to analyze the behavior of IVFm when the number of individuals is changed (genetic diversity) in the population. Hence, scenario 2.1 was created with 50 individuals; scenario 2.2 with 10 individuals, and 2.3 with 5 individuals. In the third experiment, the impact of the dimensions increase (variable) is analyzed in IVFm. Scenario 3.1 runs with 2 variables, and scenario 3.2 with 10 variables.
Table 1 shows the configurations of the experiments 1, 2 and 3. For better visualization, some acronyms were used. Like MP for mutation probability, CP for crossover probability, NuG for number of genes, GO for global optimum, DGM for division of genetic material, SC for stopping criterion, G for generations and NuIndiv for number of IVFm's individuals. The CP is 0.65 and MP is 0.01 for all experiments.

4.1 Experimental results

The experiments aim to measure the efficiency - the capacity to accelerate the GA's evolution - and efficacy - the capacity to reach the global optimum. In this work, the efficiency is measured by the average of NG and efficacy is analyzed by NGO%, success rate.
Figures 6, 7 and 8 show, respectively, the results of experiments 1, 2 and 3 of algorithms GA, IVFm/AR (AR), IVFm/EAR-P (EAR-P), IVFm/EAR-T (EAR-T) and IVFm/EAR-N (EAR-N).

4.2 Analysis of results

Analyzing the results of scenario 1.1, Table 4 and Figure 7, about the efficacy, we can identify a great performance of the IVFm and its operators because all operators reached the global optimum on all the 20 executions. On the other hand, the GA without the aid of the IVFm did not obtain good results, the canonical GA reached the global optimum on only 4 of the 20 executions (20%). On Scenario 1.2 the function parameter A was increased, characterizing deepest valleys. Nevertheless, the IVFm and its operators continued with excellent results, 100% of efficacy. Already the GA presents on average performance with 50% of efficiency.
Analyzing the efficiency on experiment 1 - scenario 1.1, the EAR-N has the best result, followed by EAR-T, EAR-P and AR. Scenario 1.2 identifies a small efficiency loss of the IVFm. The GA without IVFm presents a lower efficiency, even if considering only the average of executions on which the global optimum was reached.

Experiment 1							
Scenario	GA			IVFm		Function	
1.1	**Pop**	**NuG**	**SC**	**DGM**	**NuIndiv**	**Variables**	**A**
	50	34	GO or 250 G	4	45	2	10
1.2	**Pop**	**NuG**	**SC**	**DGM**	**NuIndiv**	**Variables**	**A**
	50	34	GO or 250 G	4	45	2	50
Experiment 2							
Scenario	GA			IVFm		Function	
2.1	**Pop**	**NuG**	**SC**	**DGM**	**NuIndiv**	**Variables**	**A**
	30	34	GO or 250 G	2	27	2	10
2.2	**Pop**	**NuG**	**SC**	**DGM**	**NuIndiv**	**Variables**	**A**
	10	34	GO or 250 G	2	9	2	10
2.3	**Pop**	**NuG**	**SC**	**DGM**	**NuIndiv**	**Variables**	**A**
	5	34	GO or 250 G	2	4	2	10
Experiment 3							
Scenario	GA			IVFm		Function	
3.1	**Pop**	**NuG**	**SC**	**DGM**	**NuIndiv**	**Variables**	**A**
	50	34	GO or 250 G	4	45	2	10
3.2	**Pop**	**NuG**	**SC**	**DGM**	**NuIndiv**	**Variables**	**A**
	50	170	GO or 250 G	10	45	10	10

Table 1. Experiments configuration

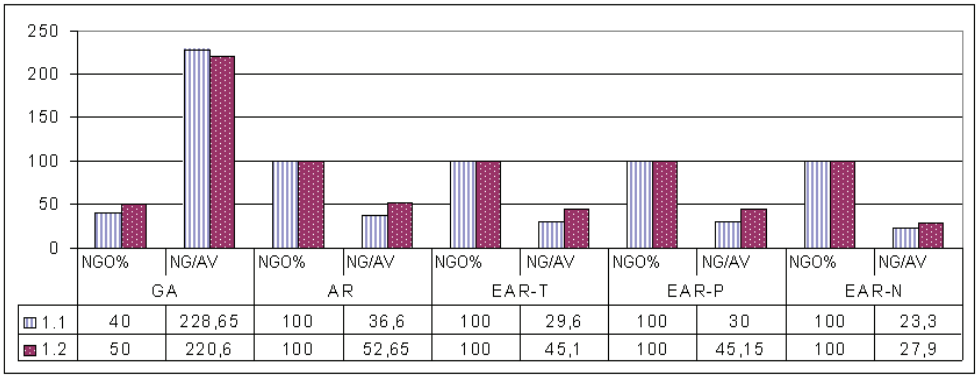
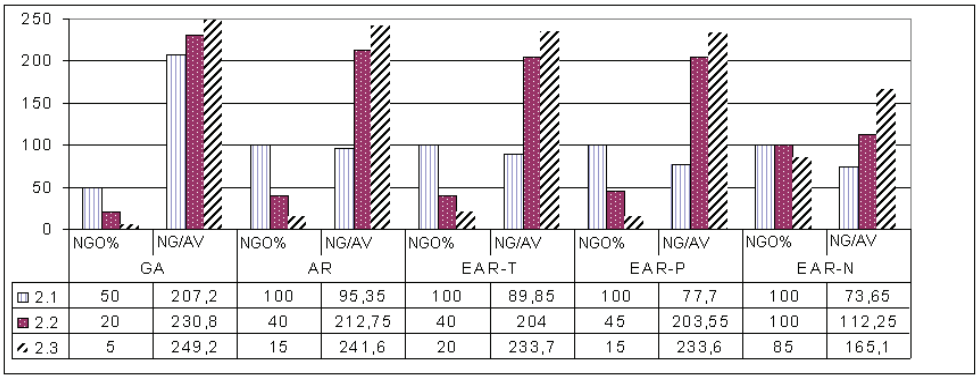


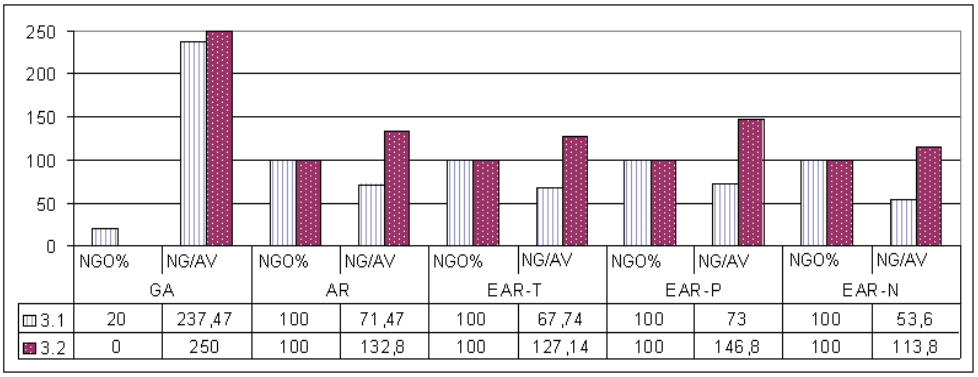
Fig. 6. Experiment 1 results

On experiment 2 we can measure the impact of the reduction of individuals (genetic diversity) on the proposed algorithm. On scenario 2.1 with 30 individuals, all operators of IVFm showed 100% of efficacy and the GA 50%. On scenario 2.2 with 10 individuals, we can observe a loss



	GA		AR		EAR-T		EAR-P		EAR-N	
	NGO%	NG/AV	NGO%	NG/AV	NGO%	NG/AV	NGO%	NG/AV	NGO%	NG/AV
2.1	50	207,2	100	95,35	100	89,85	100	77,7	100	73,65
2.2	20	230,8	40	212,75	40	204	45	203,55	100	112,25
2.3	5	249,2	15	241,6	20	233,7	15	233,6	85	165,1

Fig. 7. Experiment 2 results



	GA		AR		EAR-T		EAR-P		EAR-N	
	NGO%	NG/AV	NGO%	NG/AV	NGO%	NG/AV	NGO%	NG/AV	NGO%	NG/AV
3.1	20	237,47	100	71,47	100	67,74	100	73	100	53,6
3.2	0	250	100	132,8	100	127,14	100	146,8	100	113,8

Fig. 8. Experiment 3 results

of performance by the EAR-T (40%), EAR-P (40%) and AR (40%). The EAR-N maintained an excellent efficacy of 100% and GA fell to 20%. On scenario 2.3 with only 5 individuals, we can prove the thesis that the operators EAR-P, EAR-T and AR, besides the canonical GA, suffer an impact with the reduction of genetic material, because the success rates were 20%, 15%, 15% and 5% respectively. The EAR-N also showed reduction of efficiency on scenario 2.3, though returned surprising result of efficiency (85%), as even with very low genetic diversity.

Besides the direct dependence of the IVFm efficacy with the number of individuals, experiment 2 shows the contribution of EAP to the GA on scenarios with a few genetic material.

Finally, on experiment 3 we can identify the behavior of algorithms when we increased the problem dimension, hence the complexity. On scenario 3.1 with 2 dimensions, variables X and Y, the IVFm showed, once again, 100% of efficacy. On the other hand, the GA without the IVFm showed only 20% of efficacy. Looking at the average of generations to reach the global optimum (NG/AV), we can identify a better efficiency of EAR-N (53.6), followed by EAR-T (67.73), AR (71.47) and EAR -P (73). On scenario 3.2 with 10 dimensions, the IVFm keeps the 100% of efficacy, but we can observe a decrease in efficiency because of the NG/AV: 113.8 (EAR-N), 127.13 (EAR-T), 132.8 (AR) and 146.8 (EAR-P). Maintaining the effectiveness of 100% on scenario 3.2, IVFm showed its robustness to Rastrigin multidimensional problem.

On this scenario, the GA presents its worst performance, because it did not reach the global optimum at any time, demonstrating the difficulty to evolve on highly multimodal and multidimensional problems.

5. Conclusion

Some papers RAJAN et al. (2002); SINGH & DEB (2006); YANG & DOUGLAS (1998) propose new algorithms that, independently, have better results than the GA. On some benchmarks, however, this study suggests a joining algorithm (IVFm) that allows a better use of information generated by the GA or by the IVFm.

Tests with the benchmark Rastrigin were executed to measure the effectiveness and efficiency of the proposed IVFmGA. The established scenarios were divided in to 3 experiments with different purposes. In the first one, the capacity of the IVFm was measured to reach the global optimum in a multimodal context. The IVFm proved to be effective and efficient, because it reached the global optimum in few generations, even with the increase of the function parameter A that increased a larger complexity to the problem.

In the second experiment, some scenarios were set up to analyze the impact of the individuals' reduction in the population. This reduction decreases the material and the genetic diversity, which is usually very damaging to the evolutionary algorithms. With 30 individuals, scenario 2.1, IVFm was 100% effective, already with 10 and 5 the operators EAR-T, EAR-P and AR suffered a reduction of effectiveness and efficiency. The operator EAR-N obtained better results with 100% in scenarios 2.1 and 2.2 and 85% in scenario 2.3. These are considered excellent performances because although it only had 5 individuals, it was capable of reaching the global optimum in 85% of the cases.

In the experiment 3, the impact of the dimension increase in the algorithms was measured. The IVFm had 100% of efficacy and a low efficiency reduction when the dimension increased to 10. This result shows a robustness of the IVFm on dimensionality. The operators EAR-T, EAR-P and AR had a good efficacy and robustness too.

Analyzing all tests done, we can observe that the IVFm was able to considerably increase the efficacy and efficiency of GA.

Among the IVFm operators, the EAR-N was more effective and efficient than the others, followed by EAR-T, EAR-P and AR. The good performance of the EAR-N is justified by the way that it balances the exploration of the search area, generating new information, and exploitation of promising areas, promoted by the recombination process.

Given the tests presented, we can notice that the IVFm has a great potential to help various kinds of binary-coded evolutionary algorithms. Therefore, it is suggested as future work the joining of the IVFm in different evolutionary algorithms for different types of applications.

6. References

CHAINATE, W. T. & PONGCHAROEN, P. (2007). A new heuristic for improving the performance of genetic algorithm, *Academy of Science, Engineering and Technology* 21(1): 217–220.

EIBEN, A. E. & SCHIPPERS, C. A. (1998). On evolutionary exploration and exploitation, *Fundamenta Informaticae* 35: 35–50.

FERREIRA, C. (2001a). Gene expression programming: A new adaptive algorithm for solving problems, *Complex Systems* 13(2): 87–129.

FERREIRA, C. (2001b). Gene expression programming in problem solving, *Tutorial*, Universidade dos Açores, Portugal.
URL: *http://www.propesq.ufpe.br/anais/anais.htm*

GEN, M. & CHENG, R. (1997). *Genetic Algorithms and Engineering Design*, EDA.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional.

Jung, S. H. (2003). Queen-bee evolution for genetic algorithms, *Electronics Letters* 39(6): 575–576.

MAHFOUD, S. (1992). Crowding and preselection revisited, *in* R. Manner & B. Manderick (eds), *Parallel Problem Solving from Nature 2*, Amsterdan, pp. 27–36.

MAHFOUD, S. (1995). Niching methods for genetic algorithms, *Tech report*, Department of General Engineering, University of Illinois at Urbana-Champaign.

MICHALEWICZ, Z. (1996). *Genetic algorithms + data structures = evolution programs*, Springer-Verlag, Berlin.

MUSIL, M., WILMUT, M. J. & CHAPMAN, R. (1999). A hybrid simplex genetic algorithm for estimating geoacoustic parameters using matched-field inversion, *IEEE Journal of Oceanic Engineering* 24(3): 358–369.

PARK, J.-B., PARK, Y.M. abd WON, J. & LEE, K. (2000). An improved genetic algorithm for generation expansion planning, *Power Systems* 15(1): 916–922.

RAJAN, C., MOHAN, M. & MANIVANNAN, K. (2002). Improved genetic algorithm solution to unit commitment problem, *Transmission and Distribution Conference and Exhibition*, IEEE/PES, Asia Pacific, pp. 255–260.

RODRIGUES, E. (2003). Genetic programming in the optimization of digital circuits (in portuguese), *IV Encontro Nacional de Inteligência Artificial*, Campinas, Brazil.

RONG-LONG, W. & KOZO, O. (2005). Solving facility layout problem using an improved genetic algorithm, *Fundamentals of Electronics, Communications and Computer Sciences* E88-A(2): 606–610.

RUTTKAY, Z., EIBEN, A. & RAUE, P. (1995). Improving the performances of gas on a ga-hard csp, *Workshop on Studying and Solving Really Hard Problems*, Cassis, França, pp. 157–171.

SHIMODAIRA, H. (2002). An empirical performance comparison of niching methods for genetic algorithms, *IEICE Trans Inf Syst* E85-D(11): 1872–1880.

SINGH, G. & DEB, K. (2006). Comparison of multi-modal optimization algorithms based on evolutionary algorithms, *8th annual conference on Genetic and evolutionary computation*, Seattle, USA, pp. 1305–1312.

Spears, W. M. (1992). Crossover or mutation?, *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, pp. 221–237.

SULTAN, A. B. M., MAHMUD, R. & S., S. M. (2007). Reducing premature convergence problem through numbers structuring in genetic algorithm, *International Journal of Computer Science and Network Security* 7(4): 215–217.

TACKETT, W. A. & CARMI, A. (1994). The unique implications of brood selection for genetic programming, *IEEE World Congress on Computational Intelligence*, Orlando, FL, USA, pp. 160–165.

WU, A. S., YU, H., JIN, S., LIN, K. C. & SCHIAVONE, G. (2004). An incremental genetic algorithm approach to multiprocessor scheduling, *Parallel and Distributed Systems* 15(9): 824–834.

YANG, R. & DOUGLAS, I. (1998). Simple genetic algorithm with local tuning: Eficient global optimizing technique, *J. Optim. Theory Appl.* 98(2): 449–465.

YEN, J., LIAO, J. C., LEE, B. & RANDOLPH, D. (1998). A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method, *IEEE Trans. on Syst., Man, and Cybern.* 28(2): 173–191.

5

Bioluminescent Swarm Optimization Algorithm

Daniel Rossato de Oliveira[1], Rafael S. Parpinelli[2] and Heitor S. Lopes[3]
[1,2,3]*Bioinformatics Laboratory, Federal University of Technology Paraná (UTFPR), Curitiba (PR), 80230-901*
[2]*Department of Computer Science, Santa Catarina State University (UDESC), Joinville (SC), 89223-100*
Brazil

1. Introduction

Evolutionary Computation (EC) is a research area of metaheuristics mainly applied to real-world optimization problems. EC is inspired by biological mechanisms such as reproduction, mutation, recombination, natural selection and collective animal behavior. Two branches of EC can be highlighted: Evolutionary Algorithms (EA) comprising Genetic Algorithms (Goldberg, 1989), Genetic Programming (Koza, 1992), Differential Evolution (Storn & Price, 1997), Harmony Search (Geem et al., 2001), and others; and Swarm Intelligence (SI) comprising Ant Colony Optimization (ACO) (Dorigo & Stützle, 2004) and Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 2001; Poli et al., 2007) and others. The ACO metaheuristic[1] is inspired by the foraging behavior of ants. On the other hand, the PSO metaheuristic[2] is motivated by the coordinated movement of fish schools and bird flocks. Both ACO and PSO approaches have been applied successfully in a vast range of problems (Clerc, 2006; Dorigo & Stützle, 2004).

In recent years, new SI algorithms were proposed. They have in common biological inspirations, such as bacterial foraging (Passino, 2002), slime molds life cycle (Monismith & Mayfield, 2008), various bees behaviors (Karaboga & Akay, 2009), cockroaches infestation (Havens et al., 2008), mosquitoes host-seeking (Feng et al., 2009), bats echolocation (Yang, 2010), and fireflies bioluminescense (Krishnanand & Ghose, 2005; 2009; Yang, 2009).

This work proposes a new swarm-based evolutionary approach based on the bioluminescent behavior of fireflies, called Bioluminescent Swarm Optimization (BSO) algorithm. The BSO uses two basic characteristics of the Glow-worm Swarm Optimization (GSO) algorithm proposed by (Krishnanand & Ghose, 2005): the luciferin attractant, and the stochastic neighbor selection. However, BSO goes further introducing new features such as: stochastic adaptive step sizing, global optimum attraction, leader movement, and mass extinction. Besides, the proposed algorithm is hybridized with two local search techniques: local unimodal sampling and single-dimension perturbation. All these features makes BSO a powerful algorithm for hard optimization problems.

Experiments were done to analyze the sensitivity of the BSO to control parameters. Later, extensive experiments were performed using several benchmark functions with high

[1] ACO repository: http://iridia.ulb.ac.be/∼mdorigo/ACO/
[2] PSO Repository: http://www.particleswarm.info

dimensionality and different degrees of complexity. The performance of the BSO was compared with a well-known SI method, Particle Swarm Optimization (PSO).
The remainder of the chapter is organized as follows. Section 2 reports the existent bioluminescence inspired algorithms, focusing on the Glowworm Swarm Optimization algorithm (GSO), in which our proposed algorithm is mainly based. Section 3 shows in details the BSO algorithm and its main differences to the GSO algorithm. Section 4 explain how our experiments were done, as well as the parameter tuning process that lead to default parameter values. The results of these experiments are shown and discussed in Section 5. Conclusions and future works are presented in Section 6.

2. Bioluminescense inspired algorithms

Lampyridae is a family of insects (order *Coleoptera*) that are capable to produce natural light (bioluminescense) to attract a mate or a prey. They are commonly called fireflies or lightning bugs. In the species *Lampyris noctiluca* the fireflies are also known as glow-worms and, despite of the name, they are not worms. In this species, it is always the female who glows, and only the male has wings. In other species, *Luciola lusitanica*, both male and female firefly may emit light and both have wings (Fraga, 2008; Shimomura, 2006).
If a firefly is hungry or looks for a mate, its light glows to make the attraction of insects or mates more effective. The brightness of the bioluminescent light depends on the available amount of a pigment called *luciferin*[3].
Two optimization algorithms reported in the literature were inspired by the bioluminescent behavior of *Lampyridae* insects. The Firefly Algorithm (FA) was proposed by (Yang, 2009) as a general purpose optimization algorithm and it was applied to the optimization of benchmark functions. The Glow-worm Swarm Optimization Algorithm (GSO) that was designed to capture multiple peaks in multimodal functions, without the aim of finding the global best.

2.1 Glow-worm Swarm Optimization algorithm

The Glow-worm Swarm Optimization (GSO) algorithm was first presented by (Krishnanand & Ghose, 2005) to model the collective behavior in robotics. In this algorithm, each glow-worm uses a probabilistic mechanism to select a neighbor that has an associated luciferin value, and moves towards it. Glow-worms are attracted to neighbors that glow brighter. The movements are based only on local information and interactions with selected neighbor. This enables the swarm of glow-worms to divide themselves into disjoint subgroups that eventually converge to multiple local optima of a given multimodal function. The GSO algorithm is memoryless and the glow-worms do not retain any information about the search space.
The original GSO is shown in Algorithm 1. It starts by randomly placing in the search space a population of n glow-worms of dimension d. Each solution $\vec{x}_i = [x_{i1}, x_{i2}, ..., x_{id}]$ is evaluated by a fitness function $f(\vec{x}_i)$, $i = 1, ..., n$. At the beginning, all the glow-worms contain the same amount of luciferin l_0 and the same neighborhood range decision r_0. Each iteration consists of a luciferin update phase, followed by a movement phase based on a transition rule. Other running parameters of the algorithm are: the luciferin decay constant (ρ), the luciferin enhancement constant (γ), the step size (s), the number of neighbors (n_t), the sensor range (r_s), and a constant value (β). However, the authors observed that only two parameters have significant influence in the behavior of the algorithm: n and r_s.

[3] See the UK Glow worm survey home page at: http://www.glowworms.org.uk

Set parameters: n, l_0, r_0, ρ, γ, β, s, r_s, n_t
Randomly generate the population of glow-worms $\vec{x}_i$
for $i = 1$ **to** n **do**
    Initialize luciferin $l_i(0) = l_0$
    Initialize neighborhood range $r_d^i(0) = r_0$
end for
$t = 1$
while stop condition not met **do**
    for each glow-worm i **do** $\{update\ luciferin\}$
        $l_i(t+1) = (1-\rho) \cdot l_i(t) + \gamma \cdot f(x_i(t))$
    end for
    for each glow-worm i **do** $\{movement\ phase\}$
        Find neighbors $N_i(t)$
        for each glow-worm $j \in N_i(t)$ **do**
            Compute probability $P_{ij}(t) = \frac{l_j(t)-l_i(t)}{\sum_{k \in N_i(t)} l_k(t)-l_i(t)}$
        end for
        Select glow-worm j using P_{ij}
        Update glow-worm position with $x_i(t+1) = x_i(t) + s[\frac{x_j(t)-x_i(t)}{\|x_j(t)-x_i(t)\|}]$
        Update decision range:
            $r_d^i(t+1) = min\{r_s,\ max\{0,\ r_d^i(t) + \beta \cdot (n_t - |N_i(t)|)\}\}$
    end for
    $t = t + 1$
end while

Algorithm 1: The Glow-worm Swarm Optimization Algorithm (GSO)

The GSO algorithm was used by (Krishnanand & Ghose, 2009) to find multiple optima in multimodal benchmark functions. This work also conducted detailed parameter tuning experiments. When compared with a niched-PSO, GSO showed better results concerning the number of peaks found in almost all test functions. GSO was also applied to hazard sensing in ubiquitous environments (Krishnanand & Ghose, 2008). It should be stressed that in all applications the objective is to find multiple peaks of multimodal functions.

3. The Bioluminescent Swarm Optimization algorithm

In this section we present in detail the proposed swarm-inspired algorithm for global optimization named the Bioluminescent Swarm Optimization (BSO) algorithm. The section is divided into five subsections, namely, a general description of BSO, stochastic adaptive step sizing, global optimum attraction, mass extinction, and local search procedures.

3.1 General description of BSO algorithm

The BSO algorithm, shown in Algorithm 2, can be loosely seen as a hybrid between PSO and GSO, but with some unique features. As GSO and many other algorithms, the first step is initializing n particles in the d-dimensional search space. All particles, defined by $\vec{x}_i = [x_{i1}, x_{i2}, ..., x_{id}]$, are evaluated by a fitness function $fit(\vec{x}_i)$, $i = 1, ..., n$. BSO uses luciferin-based attraction instead of fitness-based attraction between the particles, as

proposed by the GSO. This process is controlled by the parameters ρ and γ, which are the luciferin decay constant and the luciferin enhancement constant, respectively. It also uses stochastic step size, similarly to PSO, instead of fixed step as in the GSO. This step size also varies for each particle, according to its luciferin value, and controlled by the c_s parameter. This is shown in line 18 of Algorithm 2, later explained in Section 3.2.

1: Set parameters: n, ρ, γ, s_0, c_g, c_s, lR, eT
2: Randomly generate the bioluminescent particle population $\vec{x}_i$
3: **for** $i = 1$ **to** n **do**
4: Initialize luciferin $l_i(0) = 0$
5: **end for**
6: Find the global best $g(t)$
7: $t = 1$
8: **while** stop condition not met **do**
9: **for** each particle i **do** $\{update\ luciferin\}$
10: $l_i(t+1) = (1-\rho) \cdot l_i(t) + \gamma \cdot f(x_i(t))$
11: **end for**
12: **for** each glow-worm i **do** $\{movement\ phase\}$
13: Find neighbors $N_i(t)$
14: **for** each particle $j \in N_i(t)$ **do**
15: Compute probability $P_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)}$
16: **end for**
17: Select glow-worm j using P_{ij}
18: Update particle step size with $s = s_0 \cdot \frac{1}{1 + c_s \cdot l_i(t)}$
19: Update glow-worm position with
$x_i(t+1) = x_i(t) + rand \cdot s \cdot [\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}] + c_g \cdot rand \cdot s \cdot [\frac{g(t) - x_i(t)}{\|g(t) - x_i(t)\|}]$
20: Find the global best $g(t)$
21: **if** $t\%lR = 0$ **then** $\{LocalSearchProcedures\}$
22: Perform strong local search on $g(t)$
23: **else**
24: Perform weak local search on $g(t)$
25: **end if**
26: **if** iterations without a new $g(t) = eT$ **then** $\{MassExtinction\}$
27: Reinitialize all particles but g(t)
28: **end if**
29: **end for**
30: $t = t + 1$
31: **end while**

Algorithm 2: The Bioluminescent Swarm Optimization Algorithm (BSO)

While the concept of global optimum does not exist in the GSO algorithm, every particle in BSO is attracted to the global optimum, like PSO. This attraction is part of the equation in line 19 of the Algorithm 2, controlled by the parameter c_g, and discussed later in Section 3.3. We also propose a mass extinction mechanism shown in line 27. It is controlled by the parameter eT, and explained in Section 3.4.

In the GSO, particles without neighbors, that is, particles that are the best in its vicinity, do not move. It is assumed that the most promising regions of the search space are those regions in which the best solutions were found and, therefore, they should be deeply explored. Consequently, we propose two local search schemes. The first it a weak local search, shown in line 24, and meant to be the default movement for the global best. The second one is a strong local search, shown in line 22. This procedure is supposed to take place at each lR iterations. Both methods will be discussed in Section 3.5.

A weakness of the GSO algorithm is the computational overhead due to frequent distance measurements. It is necessary to know the distance between each particle at each iteration. Therefore, the number of distance computations at each iteration is $\frac{n^2-n}{2}$, where n is the number of particles. This means that the time spent only in this phase of the algorithm varies quadratically with the population size. Consequently, this fact leads to a significant overhead when using large populations.

To avoid this problem, BSO considers an infinite radius for the neighborhood. This means that the neighborhood of a given particle is composed by all other particles having luciferin value larger than its own. Experiments reported in Section 4 showed that this approach leads to a huge improvement in processing time, with almost no degradation of the quality of solutions. Consequently, the BSO algorithm has four more parameters: c_s to control the adaptive step sizing, c_g, to control the global best attraction, eT, to control the mass extinction, and lR, to control the strong local search. On the other hand, it does not have the parameters β, r_s, r_0, and n_t, present in the GSO algorithm. This is due to the infinite radius adopted by the BSO for the neighborhood of each particle, since these four parameters were used to control the radius of each particle.

3.2 Stochastic adaptive step sizing

The BSO algorithm uses Equation 1 to compute the next position of a given particle:

$$x_i(t+1) = x_i(t) + rand \cdot s \cdot [\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}] + c_g \cdot rand \cdot s \cdot [\frac{g(t) - x_i(t)}{\|g(t) - x_i(t)\|}] \quad (1)$$

where $x_i(t)$ is the current particle position, $rand$ is a random number in the [0, 1] interval, s is the current step size of the particle, c_g is the global best attraction constant, and $g(t)$ is the global best position.

Unlike GSO that uses a fixed step size, BSO changes the step size stochastically, in the same way as in PSO. Besides, the maximum step is adaptive, according to Equation 2:

$$s = s_0 \cdot \frac{1}{1 + c_s \cdot l_i(t)} \quad (2)$$

where s_0 is the maximum step, $l_i(t)$ is the amount of luciferin of the particle, and c_s is a slowing constant. This means that a particle with a high level of luciferin will move slower, while a particle with a low level of luciferin will move faster. Since the level of luciferin is an information related to the fitness of the current and previous positions, particles laying in promising regions of the search space will perform a more thorough search, with small steps. Conversely, particles laying in regions of low quality will move faster, searching for better regions.

The step sizes resulting from Equation 2 must be between 0 and s_0. Therefore, one must assure that $l_i(t)$ is always non-negative, what can be achieved by guaranteeing non-negative fitness.

This is also needed due to the roulette-wheel selection method, which will be explained further.
The parameter c_s is the slowing constant. It adjusts how much a particle will slow down due to its luciferin value. Large values of c_s lead to particles with low luciferin levels to perform a thorough search, while small c_s values makes only particles with very high luciferin levels to perform this search.

3.3 Global optimum attraction

Although the concept of global optimum does not exist in the GSO, each particle in the BSO algorithm is attracted both to the selected neighbor and to the current global optimum, as in the PSO. This is shown in the third term of the Equation 1, where the constant c_g means the force of this attraction. This constant is usually very low, but the results are still noticeably affected. Experiments reported on Section 4 show the effects.

3.4 Mass extinction

The GSO algorithm usually shows a fast convergence to multiple local optima. In the BSO, the goal is to provide global optimization, not multiple peak finding. Although BSO usually shows a slow and smooth convergence, as reported in Section 5, an early stagnation may occur. Therefore, it is important to detect stagnation as soon as possible and take corrective measures to avoid it. This is done through a mechanism known as Mass Extinction, Decimation and Hot-Boot or, simply, Explosion. This method is frequently used in PSO, genetic algorithms and other EC algorithms (Benítez & Lopes, 2010; Hembecker et al., 2007; Kalegari & Lopes, 2010), although not used in the original GSO.
Explosion works by reinitializing all or part of the particles, but keeping the main information gathered so far, like global and local optima. In the BSO, since there is no intrinsic local optimum information, we maintain just the best particle. This occurs when there is no improvement of the best solution for a given number of iterations, represented by the eT parameter. In the Algorithm 2, this verification is done in line 26. If the condition is true, that is, the last improvement of the global best $g(t)$ took place more than eT iterations before, all particles are reinitialized, except $g(t)$, as shown in line 27.

3.5 Local search procedures

Due to the dynamic grouping nature of the BSO particles, this algorithm performs strong exploration. However, experiments have shown that its exploitation capability is not intrinsically good. We propose a corrective measure using two local search procedures: a weak one, to be executed on the best individual at each iteration; and a strong one, to be executed on the global best found so far, at each lR iterations. In our experiments, Local Unimodal Sampling (LUS) (Pedersen, 2010) was used every iteration, and a Single-Dimension Perturbation Search (SDPS) at each lR iterations. The difference between these local search procedures is not on quality or overhead of the algorithm, but only the computational effort applied in each one. The strong one is meant to use much more function evaluations than the weak.
Since the neighborhood of a particle is composed by the particles with higher luciferin levels than itself, the best particle has no neighbors and, thus, does not move. Spending computational effort with the worse particles, while leaving the best one behind is not interesting, and so the weak local search is meant to correct this. Therefore, the weak local

search is the default movement for the best particle. It is allowed one turn off the strong local search, but the weak local search is an essential part of the BSO algorithm.

3.5.1 Local Unimodal Sampling

The Local Unimodal Sampling (LUS), shown in Algorithm 3, was proposed by (Pedersen, 2010) as a simple and fast method for numerical optimization that is adequate to unimodal search spaces. It works by randomly sampling a position within a radius from the base position, and decreasing this radius exponentially. Its parameters are the initial position $\vec{x_0}$, initial sampling radius r_{0w}, the decrease rate q, and the iteration limit n_w.

1: Set parameters: $n_w, r_{0w}, q, \vec{x_0}$
2: $\vec{x} = \vec{x_0}$
3: $r = r_{0w}$
4: **for** $i = 1$ **to** n_w **do**
5: Randomly generate the movement vector $\vec{a}$ within r of $\vec{x}$
6: **if** $f(\vec{x} + \vec{a}) > f(\vec{x})$ **then**
7: $\vec{x} = \vec{x} + \vec{a}$
8: **else**
9: $r = q \cdot r$
10: **end if**
11: **end for**

Algorithm 3: Local Unimodal Sampling (LUS) algorithm.

3.5.2 Single-Dimension Perturbation Search

The strong local search procedure (SDPS), shown in Algorithm 4, is just a random perturbation in a single random dimension in each iteration. It is similar to the LUS, but the main differences are: single-dimension instead of multi-dimension movement, and linear decay of search space radius, instead of exponential. Its parameters are the initial position $\vec{x_0}$, initial sampling radius r_{0s}, and the iteration limit n_s.

1: Set parameters: n_s, $r_{0s}, \vec{x_0}$
2: $\vec{x} = \vec{x_0}$
3: **for** $i = 1$ **to** n_s **do**
4: $\vec{c} = \vec{x}$ $\{CandidateSolution\}$
5: $r = r_{0s} \cdot \frac{n_s - i}{n_s}$
6: Randomly choose a dimension j to perturbate
7: Compute $step = rand \cdot r$ $\{rand \in [-1, 1]\}$
8: $\vec{c}[j] = \vec{c}[j] + step$
9: **if** $f(\vec{c}) > f(\vec{x})$ **then**
10: $\vec{x} = \vec{c}$
11: **end if**
12: **end for**

Algorithm 4: Single-Dimension Perturbation Search (SDPS)

4. Experiments

All experiments were done using a desktop computer with a 2.8GHz quad-core processor, 2GB of RAM, running a minimal installation of Arch Linux. The application software was development ANSI-C programming language. Since the BSO is a stochastic algorithm, all

experiments were repeated 100 times with different random seeds. The performance of each approach takes into account the average best solution found in each run and the average processing time.

The BSO algorithm was applied to four well-known benchmark functions, extensively used in the literature for the evaluation of metaheuristics (Digalakis & Margaritis, 2002). The definition of the functions, the corresponding range of values and the minimum value known are shown in Table 1.

Function	Ranges	Minimum value
$f_1(\vec{x}) = \sum_{i=1}^{d}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$-5.12 \leq x_i \leq 5.12$	$f_1(\vec{0}) = 0$
$f_2(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	$-600 \leq x_i \leq 600$	$f_2(\vec{0}) = 0$
$f_3(\vec{x}) = \sum_{i=1}^{d-1}\left(0.5 + \frac{\sin^2\left(\sqrt{x_{i+1}^2 + x_i^2}\right) - 0.5}{\left(0.001\left(x_{i+1}^2 + x_i^2\right) + 1\right)^2}\right)$	$-100 \leq x_i \leq 100$	$f_3(\vec{0}) = 0$
$f_4(\vec{x}) = \sum_{i=1}^{d-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$-30 \leq x_i \leq 30$	$f_3(\vec{1}) = 0$

Table 1. Numerical benchmark functions

The first function ($f_1(\vec{x})$) is the Rastrigin function that is a multimodal function and is based on the Sphere function, with the addition of cosine modulation to produce many local minima. The locations of the minima are regularly distributed. The function was originally proposed for two dimensions, and later generalized to d dimensions (Mühlenbein et al., 1991). The main difficulty in finding optimal solutions to this function is that an optimization algorithm can be easily trapped in a local optimum on its way towards the global optimum. $\vec{x}$ is defined in the range of $[-5.12, 5.12]$ and the global minimum value for $f_1(\vec{x})$ is 0 and the corresponding global optimum solution is $\vec{x}_{opt} = (x_1, x_2, \ldots, x_d) = (0, 0, \ldots, 0)$.

The second function ($f_2(\vec{x})$) is the Griewank function (Griewank, 1981) that is strongly multimodal, because the number of local optima increases exponentially with the dimensionality (Cho et al., 2008). $\vec{x}$ is defined in the range of $[-600, 600]$ and the global minimum value for $f_2(\vec{x})$ is 0 and the corresponding global optimum solution is $\vec{x}_{opt} = (x_1, x_2, \ldots, x_d) = (0, 0, \ldots, 0)$.

The third function ($f_3(\vec{x})$) is the generalized Schaffer function F6 (Floudas & Pardalos, 1990) that is also strongly multimodal. $\vec{x}$ is defined in the range of $[-100, 100]$ and the global minimum value for $f_3(\vec{x})$ is 0 and the corresponding global optimum solution is $\vec{x}_{opt} = (x_1, x_2, \ldots, x_d) = (0, 0, \ldots, 0)$.

The fourth function ($f_4(\vec{x})$) is the Rosenbrock function (Rosenbrock, 1960), which has a long and narrow parabolic flat valley, where the global minimum is located . $\vec{x}$ is defined in the range of $[-30, 30]$ and the global minimum value for $f_4(\vec{x})$ is 0 and the corresponding global optimum solution is $\vec{x}_{opt} = (x_1, x_2, \ldots, x_d) = (1, 1, \ldots, 1)$.

Since the BSO is a maximization algorithm and these four functions are meant to be minimized, a conversion is needed. Simply multiplying the fitness by -1 will not work, because the roulette wheel selection method accepts only non-negative fitness values. Therefore, the following transformation shown in Equation 3 was applied to it.

$$fit(\vec{x}) = \frac{k}{k + f(\vec{x})} \tag{3}$$

where $fit(\vec{x})$ is the corrected fitness to be maximized, $f(\vec{x})$ is the function raw value (which we want to minimize), and k is a given constant. In our experiments, functions f_1, f_2 and f_3 used $k = 1$, and f_4 used $k = 100$.

Should be noted that Equation 3 guarantees fitness values in the interval [0,1] for the tested functions, which is needed for the step sizing process. In fact, if the function to be minimized can produce negative fitness values, a more elaborate transformation will be needed. As mentioned before, the fitness values must always be in the [0,1] interval.
The BSO was compared to the PSO algorithm using cognitive and social constants $\phi_1 = \phi_2 = 1.8$, and the inertia weight was set to $\omega = 0.6$, as recommended in (Kennedy & Eberhart, 2001; Vesterstrom & Thomsen, 2004).
The number of variables (dimensions) for all functions in our experiments was set to 10, 30 and 50. In all experiments, the maximum number of function evaluations and swarm size were set to 500,000 and 500 particles, respectively. These two parameters were used both in the BSO and the PSO algorithm. For all models, the stop condition is reaching the maximum number of evaluations or a 10^{-5} error.
The number of iterations for each local search procedure in the BSO were set to $n_w = 10$ and $n_s = 100$, meaning that the strong local search is ten times heavier than the weak local search. The initial radius for each one was set to $r_{0w} = 0.1$ and $r_{0s} = 1.0$, and the sampling radius decrease rate of the LUS was set to $q = 0.6$.

4.1 Parameter tuning

We have also investigated the behavior of BSO to different settings of some control parameters. The objective is to suggest default values, whenever possible.
Parameters ρ and γ were fixed using the default values from GSO, that is, $\rho = 0.4$ and $\gamma = 0.6$. The other parameters were tested using a factorial experiment (Box et al., 2005) with the following values: $n = [50; 300; 500; 1000]$, $s_0 = [0.3; 1.0; 3.0]$, $lR = [0; 1; 5; 10; 50]$, $c_g = [0.01; 0.03; 0.1]$, $c_s = [1; 5; 10]$, and $eT = [100; 200]$. A total of 1080 possible combinations of these parameter values can be arranged. Experiments were done with all these combinations of parameters, for each function of each dimension. Each parameter set was tested 20 times, in independent runs, and the average best result found for each test was used later to define a default parameter set.
In almost all problems, the bigger the population size n , the better the result, with $n = 1000$ achieving the best solutions, in average. However, in f_2 and f_4 for $d = 30$ and $d = 50$, the best results were found using $n = 50$. This happened because the maximum number of function evaluations was reached. The BSO has a very slow convergence, and the number of iterations in which the limit of function evaluations was reached with large populations was not enough to converge. In fact, even with a small population, the convergence curve still showed some room to improvement, meaning that this limited number of evaluations was too small for these instances of the problems. Therefore, we set as the default value $n = 500$, since it lead to results almost as good as those found with $n = 1000$ in most problems, but had much better solutions than this value when more time was needed by the algorithm.
The parameter s_0 showed to be strongly dependent of the problem. For each instance of the problems, a different value of s_0 leaded to better solutions. This was expected, since this parameter has a direct influence in the navigation steps in the search space. In search spaces with different topologies, different values of s_0 will be needed. The range used in our experiments achieved good results for all functions. Therefore, s_0 should be usually set in the range $[0.3..3.0]$. In our experiments, we used $s_0 = 1.0$ as the default value.
The global attraction, controlled by the c_g parameter, lead to better results when set to the lower values. The only exception was in f_4. However, even in this function, small values also yielded good solutions. Consequently, we defined $c_g = 0.03$ as the default value for this

parameter, since a good performance in all instances of the problems was achieved with this value.
The stochastic step sizing, which controls the thoroughness of the search that each particle performs, showed better results when the exploitation was privileged. This means higher values of c_s. As explained in Section 3.2, higher values of c_s lead particles to explore the space with slow movements, even in regions with not so high fitness levels. This parameter leads to a trade-off between exploration and exploitation, and the user should tune it according to what it is wanted to emphasize. In almost all experiments, good results were found using $c_s = 5$, and this value is set as the default for this parameter.
Our experiments also showed that the eT parameter has low influence in the quality of the final results, but lower values had a slight advantage. Since good results were achieved in all functions using $eT = 100$, we consider this as the default value.
Usually, the strong local search procedure (see Section 3.5.2) showed better results with lower values. Lower values mean more local search, not the contrary, and so, the more effort spent in local search, the better the results. However, with $lR = 1$, too much function evaluations were spent in the local search, and the algorithm had few time to explore the search space. With small populations the effect was more pronounced, since the number of function evaluations used by SDPS was proportionally larger. When testing the functions with lower dimensions, this was not a problem, since the algorithm needed much less evaluations than the upper limit set. However, when using $d = 50$, higher values of lR lead to better solutions, but the quality dropped again when lR was set too high. Considering that $lR = 5$ leaded to good results for all problems, this is defined as the default.
Table 2 summarizes the default values for the control parameters of BSO. Notice that the maximum number of iterations $max1te$, as well as the maximum number of evaluations cannot be considered parameters of the BSO. They only control the stop criterion, which can be different for each problem and application of the algorithm. One can set the stop criterion to a predefined error value, number of fitness evaluations or processing time spent, for instance.

Symbol	Name	Default Value
n	Population size	500
ρ	Luciferin decay constant	0.4
γ	Luciferin gain constant	0.6
s_0	Maximum step size	[0.3 .. 3.0]
eT	Number of stagnated iterations to enable explosion	100
lR	Period of local search procedures	5
c_g	Attraction to global best	0.03
c_s	Slowing constant	5

Table 2. Default values for the control parameters of BSO

5. Results and discussion

In this section we present results of our experiments and a comparison of performance between PSO and BSO, as well as a study of the convergence behavior of each algorithm.

5.1 Numerical results

The statistical results of 100 independent runs obtained by BSO and PSO with default parameters are shown in Tables 3 to 6. BSO was tested with and without the strong local

search (SDPS). In the tables, besides the full BSO algorithm, it is also shown wBSO, a version of BSO without the strong local search. Each column shows the average of the best values obtained in each run followed by the respective standard deviations, and the mean of the elapsed time in seconds by each algorithm for each function.
In Table 3, the results for function f_1 indicate that the BSO algorithm has an overhead significantly larger than PSO for small dimensions, but this overhead grows much slower in higher dimensions. With $d = 10$, PSO was almost 4 times faster than BSO. On the other hand, they took almost the same time to execute when $d = 50$. The quality of the solutions found by the BSO were also much better than the PSO. The reason for this, further explained in Section 5.2, is the slower convergence of BSO, avoiding getting trapped into local maxima.

	$d = 10$		$d = 30$		$d = 50$	
Model	Quality	Time (s)	Quality	Time (s)	Quality	Time (s)
BSO	0.00005 ± 0.00004	2.34	0.14368 ± 0.38712	2.84	0.32219 ± 0.80927	3.38
wBSO	0.00688 ± 0.00201	2.45	0.16968 ± 0.10514	3.04	0.88320 ± 0.59278	3.61
PSO	7.79839 ± 3.69998	0.61	27.8135 ± 7.41229	1.74	88.7282 ± 17.0144	3.04

Table 3. Statistical results obtained by all approaches - Function f_1

The results for function f_2, shown in Table 4, have the same time characteristics found in f_1. That is, large overhead for small dimensions, but equivalent performance for high dimensions. The quality of the results were close between PSO and BSO, with better solutions found by PSO than wBSO when $d = 30$ and $d = 50$.

	$d = 10$		$d = 30$		$d = 50$	
Model	Quality	Time (s)	Quality	Time (s)	Quality	Time (s)
BSO	0.03465 ± 0.02183	2.54	0.02628 ± 0.02542	3.15	0.02919 ± 0.01673	3.81
wBSO	0.08019 ± 0.02994	2.64	0.43223 ± 0.11680	3.39	0.74025 ± 0.09198	4.15
PSO	0.09003 ± 0.03284	0.67	0.23263 ± 0.09442	1.97	0.66629 ± 0.10810	3.28

Table 4. Statistical results obtained by all approaches - Function f_2

Concerning function f_3, Table 5 shows that BSO again outperformed PSO, and the difference between the time spent by each algorithm also decreased for higher dimensions.

	$d = 10$		$d = 30$		$d = 50$	
Model	Quality	Time (s)	Quality	Time (s)	Quality	Time (s)
BSO	0.07870 ± 0.02445	2.50	0.50525 ± 0.18516	3.39	1.39567 ± 0.55424	4.26
wBSO	0.09584 ± 0.02414	2.60	1.21714 ± 0.33930	3.52	3.36662 ± 1.10405	4.41
PSO	0.53988 ± 0.20318	0.62	5.60456 ± 0.90357	1.85	12.7432 ± 1.35718	3.10

Table 5. Statistical results obtained by all approaches - Function f_3

Results for function f_4, shown in Table 6, demonstrate that the time spent by PSO when $d = 50$ was higher than the time spent by the BSO. Overall, this indicates that BSO can be faster than PSO if the number of dimensions is high enough. Here, the quality of the solutions found degraded fastly for PSO, comparing with BSO, as the number of dimensions increased.

5.2 Convergence analysis

An important feature of the BSO algorithm is a very slow convergence, when compared, for instance, with PSO. To show that, we ran a set of tests without the strong local search (using

	$d = 10$		$d = 30$		$d = 50$	
Model	Quality	Time (s)	Quality	Time (s)	Quality	Time (s)
BSO	0.72827 ± 1.52126	2.16	27.0083 ± 1.75217	2.33	47.0415 ± 0.79140	2.48
wBSO	2.06372 ± 1.75578	2.23	33.0884 ± 10.4737	2.39	79.7785 ± 36.1792	2.56
PSO	0.92752 ± 2.07380	0.64	67.6840 ± 30.3727	1.86	290.274 ± 171.253	3.10

Table 6. Statistical results obtained by all approaches - Function f_4

wBSO), since the weak local search is considered the default movement for the best particle, and the PSO do not have a specific local search procedure in this implementation.
The results of these experiments are in Figures 1 to 4. In these figures, the fitness for both methods are normalized by the best fitness found by each one. This means that there is no information at all about the quality of the solutions found by each approach in these figures, only about the convergence. The quality of the solutions was previously discussed in Section 5.1. These figures shows that BSO usually has a very slow start, then accelerate its convergence speed, but still much slower than PSO. In f_1, the BSO converged around iteration 400, while PSO converged around iteration 100, see Figure 1. This slower convergence led to a much better final result, as shown in Table 3.

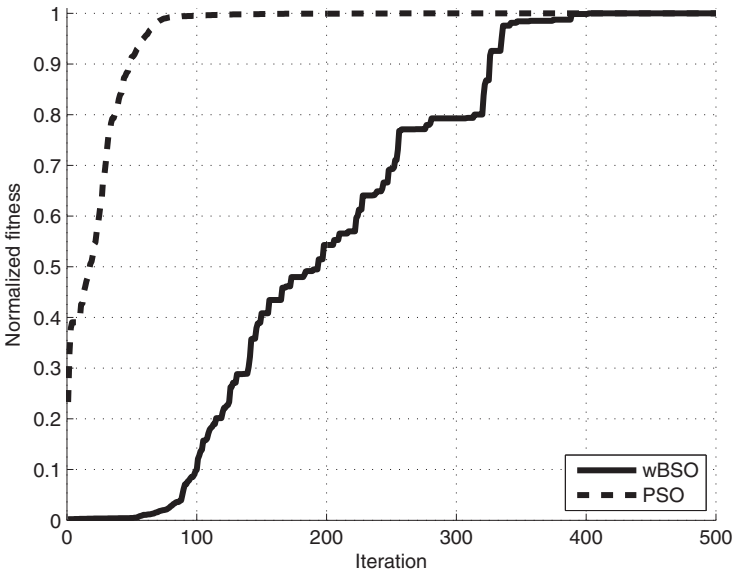


Fig. 1. Convergence curve for wBSO and PSO in f_1, d=50

The convergence curves for function f_2, shown in Figure 2, shows that BSO and PSO converged around iteration 700 and 200, respectively. However, the quality of the solution was not that good, as shown in Table 4.
Concerning function f_3, the curves shown in Figure 3 indicate that convergence ocurred when BSO reached approximately 500 iterations, and PSO found its final solution as soon as iteration 100, what lead to very bad results.

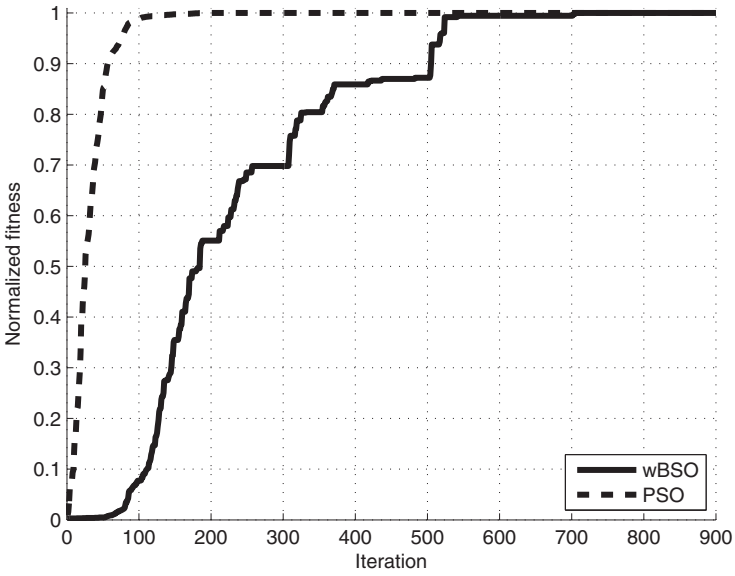


Fig. 2. Convergence curve for wBSO and PSO in f_2, d=50

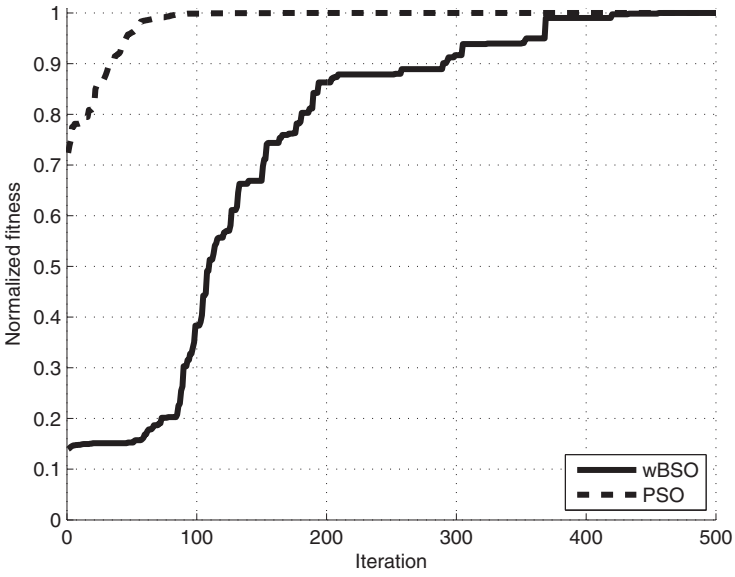


Fig. 3. Convergence curve for wBSO and PSO in f_3, d=50

For function f_4, Figure 4 shows that the BSO algorithm reached 900 iterations with presumably some potential to improve the solution, while the PSO stopped near 250 iterations.

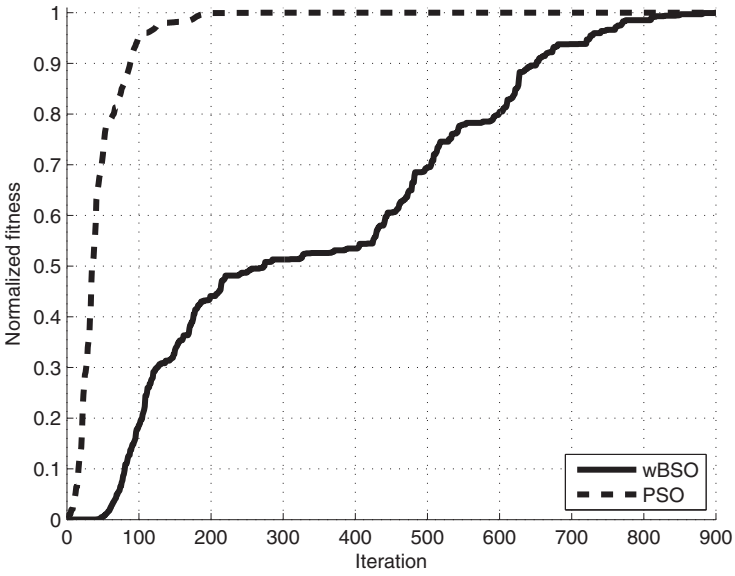


Fig. 4. Convergence curve for wBSO and PSO in f_4, d=50

In these four studies, using functions with 50 dimensions, BSO converged between 4 to 5 times slower than PSO. As consequence, this lead to very good solutions, since fast convergence usually makes the algorithm to get trapped into local maxima. The results were similar for the other dimensions studied (not shown here), always with a smooth convergence in the BSO algorithm.

6. Conclusions and future work

In this work we proposed a swarm-based algorithm for global optimization named Bioluminescent Swarm Optimization (BSO) algorithm. The algorithm is based both on research on the behavior of real *Lampyridae* family of insects and on Swarm Intelligence concepts and principles.

The BSO algorithm has a very good exploration capability, avoiding getting trapped into local maxima, while its local search procedures improved the otherwise somewhat weak exploitation. However, the introduction of local search did not remove the slow convergence characteristic, leading to high-quality final results with a smooth convergence.

The experiments concerning the BSO robustness suggested that the algorithm has low sensitivity to parameter tuning. Hence, in most cases, the default parameters proposed here should lead to good results just by tuning the s_0 parameter and the population size n.

We have compared the performance of BSO and one of the most widely used swarm intelligence method (PSO) in four benchmark functions, using three different number of dimensions for each one. Results obtained by the BSO algorithm were much better than those

obtained by the other approach, indicating that the proposed algorithm is an interesting and promising strategy for global optimization of complex problems.
Future work will address the selection system, self-tuning of the parameters, and multi-objective optimization. We will also apply the BSO algorithm to other benchmark functions, and real-world problems. We will investigate other local search methods to be used as the strong local search procedure, since this approach proved to be very promising. In the same way, hybridizing the proposed algorithm with other evolutionary computation methods may lead to improved performance.

7. Acknowledgements

This work was partially supported by the Brazilian National Research Council (CNPq) under research grant no. 309262/2007-0. Authors would like to thank UDESC (Santa Catarina State University) and the FUMDES program for the financial support.

8. References

Benítez, C. & Lopes, H. (2010). A master-slave parallel genetic algorithm for protein structure prediction using the 3D-HP side-chain model, *Journal of the Brazilian of Computer Society* 16(1): 69–78.

Box, G., Hunter, W. & Hunter, J. (2005). *Statistics for Experimenters: Design, Innovation, and Discovery*, 2nd edn, Wiley, New York.

Cho, H., Olivera, F. & Guikema, S. (2008). A derivation of the number of minima of the Griewank function, *Applied Mathematics and Computation* 204(2): 694–701.

Clerc, M. (2006). *Particle Swarm Optimization*, Wiley-ISTE Press, London.

Digalakis, J. G. & Margaritis, K. G. (2002). An experimental study of benchmarking functions for evolutionary algorithms, *International Journal of Computer Mathematics* 79(4): 403–416.

Dorigo, M. & Stützle, T. (2004). *Ant Colony Optimization*, MIT Press, Cambridge.

Feng, X., Lau, F. C. M. & Gao, D. (2009). A new bio-inspired approach to the traveling salesman problem, *in* X. Feng, F. C. Lau & D. Gao (eds), *Complex Sciences*, Vol. 5 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Berlin/Heidelberg, pp. 1310–1321.

Floudas, C. A. & Pardalos, P. M. (1990). *A collection of test problems for constrained global optimization problems*, Vol. 455 of *Lecture Notes in Computer Science*, Springer.

Fraga, H. (2008). Firefly luminescence: A historical perspective and recent developments, *Journal of Photochemical & Photobiological Sciences* 7: 146–158.

Geem, Z., Kim, J. & Loganathan, G. (2001). A new heuristic optimization algorithm: Harmony search, *Simulation* 76(2): 60–68.

Goldberg, D. E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*, Addison Wesley, Reading.

Griewank, A. (1981). Generalized descent for global optimization, *Journal of Optimization Theory and Applications* 34(1): 11–39.

Havens, T., Spain, C., Salmon, N. & Keller, J. (2008). Roach infestation optimization, *IEEE Swarm Intelligence Symposium* pp. 1–7.

Hembecker, F., Godoy Jr., W. & Lopes, H. (2007). Particle swarm optimization for the multidimensional knapsack problem, *Lecture Notes in Computer Science* 4331: 358–365.

Kalegari, D. & Lopes, H. (2010). A differential evolution approach for protein structure optimization in a 2-D off-lattice protein model, *International Journal of Bio-Inspired Computation* 2(3/4): 242–250.

Karaboga, D. & Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence, *Artificial Intelligence Review* 31(1/4): 61–85.

Kennedy, J. & Eberhart, R. (2001). *Swarm Intelligence*, Morgan Kaufmann, San Francisco.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge.

Krishnanand, K. & Ghose, D. (2005). Detection of multiple source locations using a glowworm metaphor with applications to collective robotics, *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 84–91.

Krishnanand, K. & Ghose, D. (2008). Glowworm swarm optimization algorithm for hazard sensing in ubiquitous environments using heterogeneous agent swarms, *in* B. Prasad (ed.), *Soft Computing Applications in Industry*, Vol. 226 of *Studies in Fuzziness and Soft Computing*, Springer-Verlag, Heidelberg, pp. 165–187.

Krishnanand, K. & Ghose, D. (2009). Glowworm swarm optimization for simulataneous capture of multiple local optima of multimodal functions, *Swarm Intelligence* 3(2): 87–124.

Mühlenbein, H., Schomisch, D. & Born, J. (1991). The parallel genetic algorithm as function optimizer, *Parallel Computing* 17(6-7): 619–632.

Monismith, D. & Mayfield, B. (2008). Slime mold as a model for numerical optimization, *IEEE Swarm Intelligence Symposium*, pp. 1–8.

Passino, K. (2002). Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Systems Magazine* 22(3): 52–67.

Pedersen, M. E. H. (2010). *Tuning & Simplifying Heuristical Optimization*, Phd thesis, School of Engineering Sciences, University of Southampton, UK.

Poli, R., Kennedy, J. & Blackwell, T. (2007). Particle swarm optimization: an overview, *Swarm Intelligence* 1(1): 33–57.

Rosenbrock, H. (1960). An automatic method for finding the greatest or least value of a function, *The Computer Journal* 3: 175–184.

Shimomura, O. (2006). *Bioluminescense: Chemical Principles and Methods*, World Scientific Publishing, Singapore.

Storn, R. & Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization* 11: 341–359.

Vesterstrom, J. & Thomsen, R. (2004). A comparative study of differential evolution particle swarm optimization and evolutionary algorithms on numerical benchmark problems, *IEEE Congress on Evolutionary Computation* 3: 1980–1987.

Yang, X. (2010). A new metaheuristic bat-inspired algorithm, *Nature Inspired Cooperative Strategies for Optimization*, Vol. 284 of *Studies in Computational Intelligence*, Heidelberg, pp. 65–74.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization, *Lecture Notes in Computer Sciences* 5792: 169–178.

6

A Memetic Particle Swarm Optimization Algorithm for Network Vulnerability Analysis

Mahdi Abadi and Saeed Jalili
Tarbiat Modares University
Tehran, Iran

1. Introduction

As computer networks continue to grow, it becomes increasingly more important to automate the process of evaluating their vulnerability to attacks. Despite the best efforts of software architects and developers, network hosts inevitably contain a number of vulnerabilities. Hence, it is not feasible for a network administrator to remove all vulnerabilities present in the network hosts. Therefore, the recent focus in security of such networks is on analysis of vulnerabilities globally, finding exploits that are more critical, and preventing them to thwart an intruder.

When evaluating the security of a network, it is rarely enough to consider the presence or absence of isolated vulnerabilities. This is because intruders often combine exploits against multiple vulnerabilities in order to reach their goals (Abadi & Jalili, 2005). For example, an intruder might exploit the vulnerability of a particular version of FTP to overwrite the .rhosts file on a victim host. In the next step, the intruder could remotely log in to the victim. In a subsequent step, the intruder could use the victim host as a base to launch another exploit on a new victim, and so on.

(Phillips & Swiler, 1998) proposed the concept of attack graphs, where each node represents a possible attack state. Edges represent a change of state caused by a single action taken by the intruder. (Sheyner et al., 2002) used a modified version of the model checker NuSMV (NuSMV, 2010) to produce attack graphs. (Ammann et al., 2002) introduced a monotonicity assumption and used it to develop a polynomial algorithm to encode all of the edges in an attack graph without actually computing the graph itself. These attack graphs are essentially similar to (Phillips & Swiler, 1998), where any path in the graph from an initial node to a goal node shows a sequence of exploits that an intruder can launch to reach his goal.

(Noel et al., 2005) presented a number of techniques for managing attack graph complexity through visualization. (Mehta et al., 2006) presented a ranking scheme for the nodes of an attack graph. Rank of a node shows its importance based on factors like the probability of an intruder reaching that node. Given a ranked attack graph, the system administrator can concentrate on relevant subgraphs to figure out how to start deploying security measures.

(Ou et al., 2006) presented logical attack graphs, which directly illustrate logical dependencies among attack goals and configuration information. Their attack graph generation tool builds upon MulVAL (Ou et al., 2005), a network security analyzer based on logical programming.

The aim of minimization analysis of network attack graphs is to find a minimum critical set of exploits that completely disconnect the initial nodes and the goal nodes of the graph.

(Sheyner et al., 2002) and (Jha et al., 2002) showed this problem is in fact *NP*-hard. They proposed an approximation algorithm, ApproxNAG, that can find an approximately-optimal set of exploits, which must be prevented to thwart an intruder. (Abadi & Jalili, 2006) and (Abadi & Jalili, 2008) presented an ant colony optimization algorithm, AntNAG, and a genetic algorithm, GenNAG, for minimization analysis of network attack graphs.
While it is currently possible to generate very large and complex network attack graphs, relatively little work has been done for analysis of them.
Particle swarm optimization (PSO) (Kennedy & Eberhart, 1995) is a population based stochastic optimization algorithm that was inspired by social behaviour of flocks of birds when they are searching for food.
It has been shown in many empirical studies that global optimization algorithms lack exploitation abilities in later stages of the optimization process. This is also true for the basic PSO as shown in (Shi & Eberhart, 1999); (Hendtlass & Randall, 2001); (Braendler & Hendtlass, 2002), however, it provides mechanisms to balance exploration and exploitation through proper settings of the inertia weight, acceleration coefficients and velocity clamping. Many variations of the basic PSO have been proposed to address this problem (Engelbrecht, 2005). Most of them first allow the algorithm to explore new regions, and when a good region is located, allow the algorithm to exploit the search space to refine solutions. This is a sequential approach to balancing exploration and exploitation (Engelbrecht, 2005).
Another approach is to embed a local optimizer in between the iterations of the global search heuristics. By doing this, exploration and exploitation occur in parallel (Engelbrecht, 2005). Such hybrids of local and global search heuristics have been studied elaborately in the evolutionary computation paradigm (Eiben & Smith, 2003), and are generally referred to as *memetic algorithms* (Krasnogor et al., 2006). While evolutionary algorithms take inspiration from biological evolution, memetic algorithms mimic cultural evolution. The term *meme* refers to a unit of cultural information that can be transmitted from one mind to another after reinterpretation and improvement that in the context of combinatorial optimization corresponds to local search.
In this paper, we present a memetic PSO algorithm, called ParticleNAG, for minimization analysis of large-scale network attack graphs (NAGs). We also compare the performance of ParticleNAG with ApproxNAG (Sheyner et al., 2002); (Jha et al., 2002), AntNAG (Abadi & Jalili, 2006), and GenNAG (Abadi & Jalili, 2008) for minimization analysis of several large-scale network attack graphs.
The remainder of this paper is organized as follows: Section 2 provides an overview of PSO, Section 3 introduces our network security model, and Section 4 describes the process of minimization analysis of network attack graphs. Section 5 presents ParticleNAG. Section 6 reports the experimental results and finally Section 7 draws some conclusions.

2. Particle swarm optimization

Particle swarm optimization (PSO) is a population based stochastic optimization. It was inspired by social behaviour of flocks of birds when they are searching for food. In PSO, the potential solutions, called *particles*, fly through the problem space exploring for better regions. The position of a particle is influenced by the best position visited by itself and the position of the best particle in its neighbourhood. When the neighbourhood of a particle is the entire swarm, the best position in the neighbourhood is referred to as the *global best particle*, and the

resulting algorithm is referred to as a *gbest* PSO. When smaller neighbourhoods are used, the algorithm is generally referred to as a *lbest* PSO (Kennedy et al., 2001).

The performance of each particle is measured using a predefined fitness function, which is related to the problem to be solved. Each particle in the swarm has a current position, x_i, a velocity (rate of position change), v_i, and a personal best position, y_i. The personal best position of particle *i* shows the best fitness reached by that particle at a given time. Let *f* be the objective function to be maximized. Then the personal best position of a particle at iteration or time step *t* is updated as

$$y_i(t) = \begin{cases} y_i(t-1) & \text{if} \quad f(x_i(t)) \le f(y_i(t-1)) \\ x_i(t) & \text{if} \quad f(x_i(t)) > f(y_i(t-1)) \end{cases} \tag{1}$$

For the *gbest* model, the best particle is determined from the entire swarm by selecting the best personal best position. This position is denoted as $\hat{y}$. The equation that manipulates the velocity is called the *velocity update equation* and is stated as

$$\begin{aligned} v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)) + \\ c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t)) \end{aligned} \tag{2}$$

where $v_{ij}(t+1)$ is the velocity updated for the *j*th dimension, $j = 1, 2, \ldots, d$. c_1 and c_2 are the acceleration constants, where the first moderates the maximum step size towards the best personal of the particle, while the second moderates the maximum step size towards the global best particle in just one iteration. $r_{1j}(t)$ and $r_{2j}(t)$ are two random values in the range [0,1] and give the PSO algorithm a stochastic search property.

Velocity updates on each dimension can be clamped with a user defined maximum velocity $V_{\max}$, which would prevent them from exploding, thereby causing premature convergence (Eberhart et al., 1996); (Shi, 2004). Each particle updates its position using the following equation:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3}$$

In swarm terminology, particle *i* is flying to its new position $x_i(t+1)$. After the new position is calculated for each particle, the iteration counter increases and the new particle positions are evaluated. This process is repeated until some convergence criteria is satisfied.

(Kennedy & Eberhart, 1997) have adapted PSO to search in binary spaces. For binary PSO, the elements of x_i, y_i and $\hat{y}$ can only take the values 0 and 1. The velocity v_i is interpreted as a probability to change a bit from 0 to 1, or from 1 to 0 when updating the position of particles. Therefore, the velocity vector remains continuous-valued. Since each v_{ij} is a real value, a mapping needs to be defined from v_{ij} to a probability in the range [0,1]. This is done by using a sigmoid function to squash velocities into a [0,1] range. The sigmoid function is defined as

$$sig(v) = \frac{1}{1 + e^{-v}} \tag{4}$$

The equation for updating positions is then replaced by the following probabilistic update equation:

$$x_{ij}(t+1) = \begin{cases} 0 & \text{if} \quad r_{3j}(t) \geq sig(v_{ij}(t+1)) \\ 1 & \text{if} \quad r_{3j}(t) < sig(v_{ij}(t+1)) \end{cases} \tag{5}$$

where $r_{3j}(t)$ is a random value in the range [0,1].
In binary PSO, the meaning and behaviour of velocity clamping differ substantially from real-valued PSO. With the velocity interpreted as a probability of change, velocity clamping, $V_{\max}$, sets the minimal probability for a bit to change its value from 0 to 1, or from 1 to 0 (Engelbrecht, 2005).
In this paper, we use the *gbest* model of binary PSO for minimization analysis of network attack graphs.

3. Network security model

Our network security model is a tuple (*S*, *H*, *C*, *T*, *E*, *M*, *R*), where *S* is a set of services, *H* is a set of hosts connected to the network, *C* is a relation expressing connectivity between hosts, *T* is a relation expressing trust between hosts, *E* is a set of individual known exploits that intruder can use to construct attack scenarios, *M* is a set of countermeasures that must be implemented to prevent exploits, and *R* is a model of intruder.

Services

Each service $s \in S$ is a pair (svn, p), where svn is the service name and p is the port on which the service is listening.

Hosts

Each host $h \in H$ is a tuple $(id, svcs, plvl, vuls)$, where id is a unique host identifier, $svcs$ is a set of services running on the host, $plvl$ is the level of privilege that the intruder has on the host, and $vuls$ is a set of host-specific vulnerable components. For simplicity, we only consider three privilege levels: *none*, *user*, and *root*.

Network Connectivity

Network connectivity is modelled as a relation $C \subseteq H \times H \times P$, where P is a set of port numbers. Each network connectivity $c \in C$ is a triple (h_s, h_t, p), where h_s is the source host, h_t is the target host, and p is the target port number. Note that the connectivity relation incorporates network elements such as firewalls that restrict the ability of one host to connect to another.

Trust Relationships

Trust relationships are modelled as a relation $T \subseteq H \times H$, where $T(h_t, h_s)$ indicates that a user may log in from host h_s to host h_t without authentication.

Exploits

Each exploit $e \in E$ is a tuple $(pre, h_s, h_t, post)$, where pre is a list of conditions that must hold before launching the exploit, h_s is the host from which the exploit is launched, h_t is the host targeted by the exploit, and $post$ specifies the effects of exploit on the network. An exploit $e \in E$ is *inevitable* if its prevention is not feasible or incurs high cost. The set of inevitable exploits is denoted by I.

Countermeasures

To prevent an exploit $e \in E$, the security analyst must implement a suitable countermeasure $m \in M$, such as

- changing the firewall configuration
- patching the vulnerability that made this exploit possible
- deploying a host-based or network-based intrusion detection and prevention system
- modifying the configuration of network services and applications
- deleting user accounts
- changing access rights
- setting up a virtual private network (VPN)

Intruder

The intruder has some knowledge about the target network, such as known vulnerabilities, user passwords, and information gathered with port scans. The intruder's knowledge is modelled as a relation $R \subseteq ID \times PW \times VUL \times INF$, where ID is a set of host identifiers, PW is a set of user passwords, VUL is a set of known vulnerabilities, and INF is a set of information gathered through port scans and operating system identification techniques.

4. Minimization analysis of network attack graphs

Let $E = \{e_1, e_2, ..., e_n\}$ be the set of exploits, $I \subseteq E$ be the set of inevitable exploits, $M = \{m_1, m_2, ..., m_p\}$ be the set of countermeasures, and $prv : M \to 2^{E \setminus I}$ be a function. An exploit $e_j \in prv(m_i)$ if and only if implementing the countermeasure m_i prevents the exploit e_j.

A network attack graph is a tuple $G=(V, A, V_0, V_f, L)$, where V is the set of nodes, A is the set of directed edges, $V_0 \subseteq V$ is the set of initial nodes, $V_f \subseteq V$ is the set of goal nodes, and $L : A \to E$ is a labelling function, where $L(a) = e_j$ if and only if an edge $a = (v, v')$ corresponds to an exploit $e_j \in E$. A path π in G is a sequence of nodes $v_1, v_2, ..., v_m$, such that $v_i \in V$ and $(v_i, v_{i+1}) \in A$, where $1 \le i < m$. The label of path π is a subset of the set of exploits E. Each attack scenario corresponds to a complete path that starts from an initial node and ends in a goal node.

Let $S = \{S_1, S_2, ..., S_l\}$ be the set of attack scenarios represented by the network attack graph G. The attack scenario $S_k \in S$ is hit by the exploit $e_j \in E$ if $e_j \in S_k$.

Definition 1. Total Hit Value

For each exploit $e_j \in E$, the total hit value $hv_t(e_j)$ is defined to be the number of attack scenarios that are hit by e_j.

$$hv_t(e_j) = \left| \left\{ S_k \in S \mid e_j \in S_k \right\} \right| \tag{6}$$

Definition 2. Redundant Exploit

Let $U \subseteq E$ be a subset of exploits and $hs(U)$ be the set of attack scenarios hit by the exploits in U.

$$hs(U) = \left\{ S_k \in S \mid e_j \in S_k \text{ for some } e_j \in U \right\} \tag{7}$$

An exploit e_j is redundant with respect to U if $hs(U \setminus \{e_j\}) = hs(U)$.

Definition 3. Partial Hit Value

Let $U \subseteq E$ be a subset of exploits. For each exploit $e_j \notin U$, the partial hit value $hv_p(e_j, U)$ is defined to be the number of attack scenarios that are hit by e_j, but that are not hit by any exploit in U.

$$hv_p(e_j, U) = \left| \left\{ S_k \in S \mid e_j \in S_k \wedge S_k \notin hs(U) \right\} \right| \tag{8}$$

Definition 4. Exclusive Hit Value

Let $U \subseteq E$ be a subset of exploits. For each exploit $e_j \in U$, the exclusive hit value $hv_x(e_j, U)$ is defined to be the number of attack scenarios that are hit by e_j, but that are not hit by any exploit in $U \setminus \{e_j\}$.

Definition 5. Critical Set of Exploits

A subset of exploits $CE \subseteq E \setminus I$ is critical if and only if all attack scenarios are hit by the exploits in it. Equivalently, CE is critical if and only if every complete path from an initial node to a goal node of the network attack graph G has at least one edge labelled with an exploit $e_j \in CE$.

Definition 6. Minimal Critical Set of Exploits

A critical set of exploits CE is minimal if it contains no redundant exploit.

Definition 7. Minimum Critical Set of Exploits

A critical set of exploits CE is minimum if there is no critical set of exploits CE' such that $|CE'| < |CE|$.

Definition 8. Critical Set of Countermeasures

A subset of countermeasures $CM \subseteq M$ is critical if and only if all attack scenarios are prevented by implementing the countermeasures in it. Equivalently, CM is critical if and only if every complete path from an initial node to a goal node of the network attack graph G has at least one edge labelled with an exploit $e_j \in es(CM)$, where $es(CM)$ is the set of exploits prevented by implementing the countermeasures in CM.

$$es(CM) = \bigcup_{m_i \in CM} prv(m_i) \tag{9}$$

Definition 9. Minimal Critical Set of Countermeasures

A critical set of countermeasures CM is minimal if it contains no redundant countermeasure.

Definition 10. Minimum Critical Set of Countermeasures

A critical set of countermeasures CM is minimum if there is no critical set of countermeasures CM' such that $|CM'| < |CM|$.

In general, there can be multiple minimum critical set of exploits/countermeasures. We can now state formally two problems: MCEP and MCCP (Sheyner et al., 2002); (Jha et al., 2002).

Definition 11. Minimum Critical Set of Exploits Problem (MCEP)

Given a network attack graph G and a set of exploits E, find a minimum critical subset of exploits $CE \subseteq E \setminus I$ for G.

Definition 12. Minimum Critical Set of Countermeasures Problem (MCCP)

Given a network attack graph G, a set of exploits E, and a set of countermeasures M, find a minimum critical subset of countermeasures $CM \subseteq M$ for G.

There is a trivial reduction from MCEP to MCCP, and vice versa. Given an instance (G, E) of MCEP, we can construct an instance (G, E, M) of MCCP where $M = \{\{e_j\} \mid e_j \in E\}$.
A typical process for solving MCEP or MCCP is shown in Fig. 1. First, vulnerability scanning tools, such as Nessus (Deraison, 2010), determine vulnerabilities of individual hosts. Using this vulnerability information along with exploit templates, intruder's goals, and other information about the network, such as connectivity between hosts, a network attack graph is generated. In this directed graph, each complete path from an initial node to a goal node corresponds to an attack scenario. The minimization analysis of the network attack graph determines a minimum critical set of exploits/countermeasures that must be prevented/implemented to guarantee no attack scenario is possible.

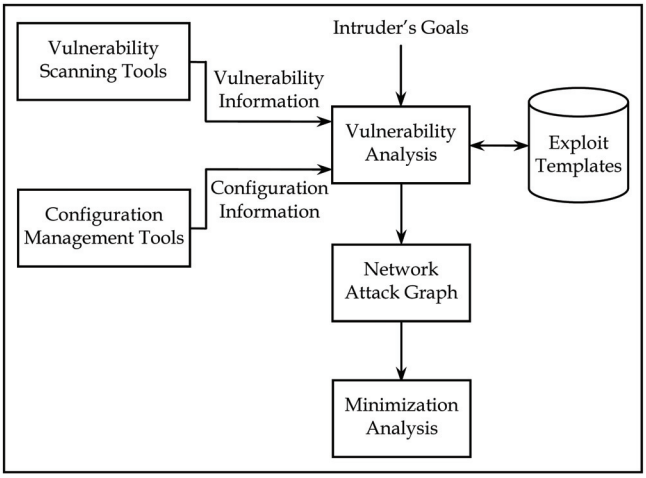


Fig. 1. Minimization analysis of network attack graphs

4. ParticleNAG

In this section, we present ParticleNAG, a memetic particle swarm optimization algorithm for minimization analysis of large-scale network attack graphs. The aim of minimization analysis of network attack graphs is to find a minimum critical set of exploits/ countermeasures. This problem is in fact a constrained optimization problem in which the objective is to find a solution with minimum cardinality and the constraint is that the solution must be critical (i.e., it must hit all attack scenarios).
Fig. 2 shows the pseudo-code of ParticleNAG. The first step is to initialize the swarm and control parameters. Then repeated iterations of the algorithm are executed until some termination condition is met (e.g., a maximum number of iterations is reached). Within each iteration, if each particle's current position x_i does not represent a critical set of exploits, a greedy repair algorithm is applied to it. Then redundant exploits of x_i are eliminated. After that, x_i is improved by a local search heuristic procedure. Then the particle's personal best position y_i is updated using equation (1). The global best position $\hat{y}$ is then determined from the entire swarm by selecting the best personal best position. Finally, the velocity and the position of each particle are updated using equations (2) and (5).

```
procedure ParticleNAG
    Set parameters, create and initialize the swarm
    while termination condition not met do
        for each particle i do
            if x_i does not represent a critical set of exploits then
                Apply the greedy repair procedure to x_i ;
            end if
            Eliminate redundant exploits of x_i ;
            Apply the local search heuristic to x_i ;
            Update the personal best position y_i ;
        end for
        Update the global best position ŷ ;
        for each particle i do
            Update the velocity v_i ;
            Update the position x_i ;
        end for
    end while
end ParticleNAG
```

Fig. 2. The ParticleNAG algorithm

5.1 Problem representation

Let $E = \{ e_1, e_2, ..., e_n \}$ be the set of preventable exploits. Each particle position x_i corresponds to an n-bit vector $(x_{i1}, x_{i2}, ..., x_{in})$ and represents a subset of exploits $E_i \subseteq E$ in which the exploit $e_j \in E_i$ if and only if the element $x_{ij} = 1$.

$$E_i = \{ e_j \in E \mid x_{ij} = 1 \} \tag{10}$$

Let $S = \{ S_1, S_2, ..., S_l \}$ be the set of attack scenarios represented by the network attack graph G. The attack scenario $S_k \in S$ is hit by the particle position x_i if $S_k \cap E_i \neq \varnothing$.

The particle position x_i represents a critical set of exploits if all attack scenarios are hit by it. The aim of minimization analysis of network attack graphs is to find a minimum critical set of exploits. So ParticleNAG uses the following fitness function to evaluate the quality of x_i:

$$f(x_i) = |E| - |E_i| \tag{11}$$

5.2 Greedy repair

The set of exploits represented by a particle position x_i may not be critical. In other words, it may not hit all attack scenarios.

Let E_i be the set of exploits represented by a particle position x_i. As shown in Fig. 3, the greedy repair algorithm chooses at each step an exploit $e_k \in E$ such that $e_k \notin E_i$ and it maximizes the partial hit value $hv_p(e_k, E_i)$. It then adds e_k to E_i and changes its corresponding element x_{ik} to 1. This is repeated until a critical set of exploits is obtained.

procedure GreedyRepair (x_i)
  $E_i = \{e_j \in E \mid x_{ij} = 1\}$;
  while x_i does not represent a critical set of exploits **do**
    Choose an exploit $e_k \in E$ such that $e_k \notin E_i$ and it maximizes the partial hit value $hv_p(e_k, E_i)$;
    $E_i = E_i \bigcup \{e_k\}$;
    $x_{ik} = 1$;
    $v_{ik} = V_{\max}$;
  end while
  return x_i;
end GreedyRepair

Fig. 3. The greedy repair procedure

5.3 Greedy elimination

The critical set of exploits represented by a particle position x_i may contain redundant exploits, which must be eliminated. Let E_i be the critical set of exploits represented by x_i. The exploit e_j is called *candidate redundant* with respect to E_i if $hv_x(e_j, E_i) = 0$. The set of candidate redundant exploits of E_i is denoted by R_i.

$$R_i = \{ e_j \in E_i \mid hv_x(e_j, E_i) = 0 \} \tag{12}$$

For each candidate redundant exploit $e_j \in R_i$, the *selection value* $sv(e_j, E_i)$ is calculated as

$$sv(e_j, E_i) = \sum_{e_k \in E_i \setminus \{e_j\}} hv_x(e_k, E_i \setminus \{e_j\}) \tag{13}$$

The selection value is used to evaluate candidate redundant exploits of a critical set of exploits in order to choose a candidate redundant exploit to be removed from it.

procedure GreedyElimination (x_i)
  $E_i = \{ e_j \in E \mid x_{ij} = 1 \}$;
  $R_i = \{ e_j \in E_i \mid hv_x(e_j, E_i) = 0 \}$;
  while $R_i \neq \varnothing$ do
    Choose an exploit $e_k \in R_i$ that maximizes the selection value $sv(e_k, E_i)$;
    $E_i = E_i \setminus \{e_k\}$;
    $x_{ik} = 0$;
    $v_{ik} = -V_{\max}$;
    $R_i = \{ e_j \in E_i \mid hv_x(e_j, E_i) = 0 \}$;
  end while
  return x_i;
end GreedyElimination

Fig. 4. The greedy elimination procedure

In Fig. 4 an algorithm is presented, which can be used to eliminate redundant exploits of x_i. Let E_i be the critical set of exploits represented by x_i. The algorithm is based on the idea that it is good to remove an exploit e_k from E_i if e_k is a candidate redundant exploit and hits attack scenarios that are hit by too many other exploits in E_i. Hence, at each step, the algorithm chooses a candidate redundant exploit e_k from R_i that maximizes the selection value $sv(e_k, E_i)$. It then removes e_k from E_i and changes its corresponding element x_{ik} to 0. This is repeated until a minimal critical set of exploits is obtained.

5.4 Local search heuristic

Combining global and local search is a strategy used by many successful global optimization approaches.

In ParticleNAG, a local search heuristic is applied to the current position of each particle to improve them before their personal best positions are updated. The local search heuristic is based on the following idea: given a particle position x_i and its corresponding critical set of exploits E_i, suppose there is an exploit $e_j \in E$ such that $e_j \notin E_i$ and $E_i \cup \{e_j\}$ contains at least two exploits other than e_j, say $e'_1, ..., e'_r$, with $r \geq 2$ that are redundant. Then we conclude that $(E_i \setminus \{e'_1, ..., e'_r\}) \cup \{e_j\}$ is a better critical set of exploits than E_i. The gain of the exploit e_j with respect to E_i is $g(e_j, E_i) = l - 1$. In this case, we call e_j a *candidate dominant* exploit.

---

procedure LocalSearch(x_i)
  $E_i = \{ e_j \in E \mid x_{ij} = 1 \}$;
  while improvement is possible **do**
    Choose an exploit $e_k \in E$ such that $e_k \notin E_i$ and $g(e_k, E_i) > 0$;
    $E_i = E_i \cup \{e_k\}$;
    $x_{ik} = 1$;
    $v_{ik} = V_{\max}$;
    Eliminate redundant exploits of x_i ;
  end while
  return x_i ;
end LocalSearch

Fig. 5. The local search heuristic procedure

As shown in Fig. 5, the local search heuristic first chooses a candidate dominant exploit e_k and changes its corresponding element x_{ik} to 1. It then eliminates the redundant exploits of the new position using the algorithm already presented in Section 5.3 for eliminating redundant exploits. This process is repeated until no further improvement is possible.

6. Experiments

In order to evaluate the performance of ParticleNAG, we performed our experiments over a sample network attack graph and several randomly generated large-scale network attack graphs.

6.1 Sample network attack graph

Consider the network shown in Fig. 6. There are three target hosts called *RedHat*, *Windows* and *Fedora* on an internal network, and a host called *PublicServer* on an isolated demilitarized zone (DMZ) network. One firewall separates the internal network from the DMZ and another firewall separates the DMZ from the rest of the Internet. A number of services are running on each of the hosts of *RedHat*, *Windows*, *Fedora*, and *PublicServer*. Also, each of the above hosts has a number of vulnerabilities. Vulnerability scanning tools such as Nessus (Deraison, 2010) can be used to find the vulnerabilities of each host.

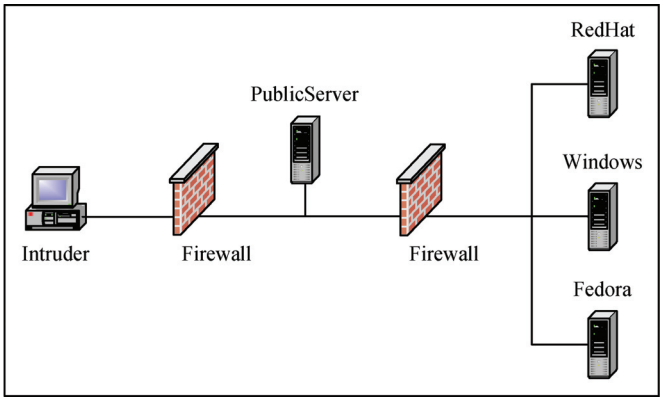


Fig. 6. An example network

Different types of services and vulnerabilities available on the network hosts are introduced in Table 1.

iis_bof(*h*)	IIS web server has buffer overflow vulnerability on host *h*
exchange_ivv(*h*)	Exchange mail server has input validation vulnerability on host *h*
squid_conf(*h*)	Squid web proxy is misconfigured on host *h*
licq_ivv(*h*)	LICQ client has input validation vulnerability on host *h*
sshd_bof(*h*)	SSH server has buffer overflow vulnerability on host *h*
scripting(*h*)	HTML scripting is enabled on host *h*
ftp(*h*)	FTP service is running on host *h*
wdir(*h*)	FTP home directory is writable on host *h*
fshell(*h*)	FTP user has executable shell on host *h*
xterm_bof(*h*)	*xterm* program has buffer overflow vulnerability on host *h*
at_bof(*h*)	*at* program has buffer overflow vulnerability on host *h*
database(*h*)	database service is running on host *h*

Table 1. Types of services and vulnerabilities running on the network hosts

The *RedHat* host on the internal network is running FTP and SSH services. The *Fedora* host is running several services: LICQ chat software, Squid web proxy, FTP and a database. The LICQ client lets Linux users exchange text messages over the Internet. The Squid web proxy is a full-featured web proxy cache. Web browsers can then use the local Squid cache as a proxy server, reducing access time as well as bandwidth consumption. The *PublicServer* host on the DMZ network is running IIS and Exchange services.
The connectivity information among the network hosts is shown in Table 2. In this Table, each entry corresponds to a pair of (h_s, h_t) in which h_s is the source host and h_t is the target host. Every entry has five boolean values. These values are 'T' if host h_s can connect to host h_t on the ports of *http*, *licq*, *ftp*, *ssh*, and *smtp*, respectively.

Host	Intruder	PublicServer	RedHat	Windows	Fedora
Intruder	F,F,F,F,F	T,F,F,F,T	F,F,F,F,F	F,F,F,F,F	F,F,F,F,F
PublicServer	F,F,F,F,F	T,F,F,F,T	F,F,T,T,F	F,F,F,F,F	T,T,T,F,F
RedHat	F,F,F,F,F	T,F,F,F,T	F,F,T,T,F	F,F,F,F,F	T,T,T,F,F
Windows	F,F,F,F,F	T,F,F,F,T	F,F,T,T,F	F,F,F,F,F	T,T,T,F,F
Fedora	F,F,F,F,F	T,F,F,F,T	F,F,T,T,F	F,F,F,F,F	T,T,T,F,F

Table 2. Network connectivity information

The intruder launches his attack starting from a single host, *Intruder*, which lies on the outside network. His goal is to disrupt the database service on the host *Fedora*. To achieve this goal, the intruder should gain the *root* privilege on this host.
There are *wdir*, *fshell*, and *sshd_bof* vulnerabilities on the *RedHat* host, scripting vulnerability on the *Windows* host, *wdir*, *fshell*, *squid_conf*, and *licq_ivv* vulnerabilities on the *Fedora* host, and *iis_bof* and *exchange_ivv* on the *PublicServer* host. Also, *at* and *xterm* programs on the *RedHat* and *Fedora* are vulnerable to buffer overflow. The intruder can use ten generic exploits, described as follows:

- *iis_r2r*
  Buffer overflow vulnerability in the Microsoft IIS web server allows remote intruders to gain root shell on the target host.
- *exchange_r2u*
  The OLE component in the Microsoft Exchange mail server does not properly validate the lengths of messages for certain OLE data, which allows remote intruders to execute arbitrary code.
- *squid_ps*
  The intruder can use a misconfigured Squid web proxy to conduct unauthorized activities such as port scanning.
- *licq_r2u*
  The intruder can send a specially crafted URL to the LICQ client to execute arbitrary commands on the target host.
- *script_r2u*
  Microsoft Internet Explorer allows remote intruders to execute arbitrary code via malformed Content-Disposition and Content-Type header fields that cause the

application for the spoofed file type to pass the file back to the operating system for handling rather than raise an error message.

- *ssh_r2r*
  Buffer overflow vulnerability in the SSH server allows remote intruders to gain root shell on the target host.
- *ftp_rhosts*
  Using FTP vulnerability, the intruder creates a .rhosts file in the FTP home directory, creating a remote login trust relationship between his host and the target host.
- *rsh_r2u*
  Using an existing remote login trust relationship between two hosts, the intruder logs in from one machine to another, getting a user shell without supplying a password.
- *xterm_u2r*
  Buffer overflow vulnerability in the *xterm* program allows local users to gain root shell on the target host.
- *at_u2r*
  Buffer overflow vulnerability in the *at* program allows local users to gain root shell on the target host.

In Table 3, each generic exploit is represented by its preconditions and postconditions. More information about each of the exploits is available in (NVD, 2010). Before an exploit can be used, its preconditions must be met. Each exploit will increase the network vulnerability if it is successful. Among the ten generic exploits shown in Table 3, the first eight generic exploits require a pair of hosts and the last two generic exploits require only one host. Therefore, there are 8 * 5 * 4 + 2 * 4 = 168 exploits in total, which the intruder can try. Each attack scenario for the above network consists of a subset of these 168 exploits. For example, consider the following attack scenario:

1. *iis_r2r(Intruder, PublicServer)*
2. *squid_ps(PublicServer, Fedora)*
3. *licq_r2u(PublicServer, Fedora)*
4. *xterm_u2r(Fedora, Fedora)*

The intruder first launches the *iis_r2r* exploit to gain *root* privilege on the *PublicServer* host. Then he uses the *PublicServer* host to launch a port scan via the vulnerable Squid web proxy running on the *Fedora* host. The scan discovers that it is possible to gain *user* privilege on the *Fedora* host with launching the *licq_r2u* exploit. After that, a simple local buffer overflow gives the intruder *root* privilege on the *Fedora* host. The attack graph for the above network consists of 164 attack scenarios. Each attack scenario consists of between 4 to 9 exploits.

Experimental Results

We applied ParticleNAG for minimization analysis of the above network attack graph. To evaluate the performance of the algorithm, we performed several experiments.

In the first experiment, we assumed that all exploits are preventable. Therefore, the aim was to find a minimum critical set of exploits among 168 exploits. Using ParticleNAG, the following minimum critical set of exploits was found:

$$CE = \{\ iis\_r2r(Intruder, PublicServer),\ exchange\_r2u(Intruder, PublicServer)\ \}$$

Exploit	Preconditions	Postconditions
$iis\_r2r(h_s, h_t)$	$iis\_bof(h_t)$ $C(h_s, h_t, http)$ $plvl(h_s) \geq user$ $plvl(h_t) < root$	$\neg iis(h_t)$ $plvl(h_t) := root$
$exchange\_r2u(h_s, h_t)$	$exchange\_ivv(h_t)$ $C(h_s, h_t, smtp)$ $plvl(h_s) \geq user$ $plvl(h_t) = none$	$plvl(h_t) := user$
$squid\_ps(h_s, h_t)$	$squid\_conf(h_t)$ $\neg scan$ $C(h_s, h_t, http)$ $plvl(h_s) \geq user$	$scan$
$licq\_r2u(h_s, h_t)$	$licq\_ivv(h_t)$ $scan$ $C(h_s, h_t, licq)$ $plvl(h_s) \geq user$ $plvl(h_t) = none$	$plvl(h_t) := user$
$script\_r2u(h_s, h_t)$	$scripting(h_t)$ $C(h_t, h_s, http)$ $plvl(h_s) \geq user$ $plvl(h_t) = none$	$plvl(h_t) := user$
$sshd\_r2r(h_s, h_t)$	$sshd\_bof(h_t)$ $C(h_s, h_t, ssh)$ $plvl(h_s) \geq user$ $plvl(h_t) < root$	$\neg ssh(h_t)$ $plvl(h_t) := root$
$ftp\_rhosts(h_s, h_t)$	$ftp(h_t)$ $wdir(h_t)$ $fshell(h_t)$ $\neg T(h_t, h_s)$ $C(h_s, h_t, ftp)$ $plvl(h_s) \geq user$	$T(h_t, h_s)$
$rsh\_r2u(h_s, h_t)$	$T(h_t, h_s)$ $plvl(h_s) \geq user$ $plvl(h_t) = none$	$plvl(h_t) := user$
$xterm\_u2r(h_t, h_t)$	$xterm\_bof(h_t)$ $plvl(h_t) = user$	$plvl(h_t) := root$
$at\_u2r(h_t, h_t)$	$at\_bof(h_t)$ $plvl(h_t) = user$	$plvl(h_t) := root$

Table 3. Exploit templates

In the second experiment, we assumed that the generic exploits *iis_r2r*, *exchange_r2u*, and *xterm_u2r* are inevitable, i.e., the prevention of them is not feasible or incurs high cost. Therefore, the aim was to find a minimum critical set of exploits among 124 exploits. Using ParticleNAG, the following minimum critical set of exploits was found:

$$CE = \{\ licq\_r2u(PublicServer, Fedora),\ licq\_r2u(RedHat, Fedora),\ script\_r2u(PublicServer, Windows),\ ftp\_rhosts(PublicServer, Fedora),\ ftp\_rhosts(RedHat, Fedora)\ \}$$

It should be mentioned that the exact cardinality of the minimum critical set of exploits for this network attack graph is 5, so the above critical set of exploits found by ParticleNAG is minimum. While using ApproxNAG (Sheyner et al., 2002); (Jha et al., 2002), the following minimum critical set of exploits was found:

$$CE = \{\ script\_r2u(PublicServer, Windows),\ at\_u2r(Fedora, Fedora),\ sshd\_r2u(PublicServer, RedHat),\ ftp\_rhosts(PublicServer, RedHat),\ squid\_ps(PublicServer, Fedora),\ ftp\_rhosts(PublicServer, Fedora)\ \}$$

The second experiment shows ParticleNAG can find a critical set of exploits with less cardinality.

In the experiments, the parameters were set to $c_1 = 2$, $c_2 = 2$, and $V_{max} = 4$, which are values commonly used in the binary PSO literature. The swarm size was set to $m = 10$ and the maximum number of iterations was set to $t_{max} = 50$.

6.2 Large-scale network attack graphs

A large computer network builds upon multiple platforms, runs different software packages and supports several modes of connectivity. Despite the best efforts of software architects and developers, each network host inevitably contains a number of vulnerabilities.

Several factors can make network attack graphs larger so that finding a minimum critical set of exploits/countermeasures becomes more difficult. An obvious factor is the size of the network under analysis. Our society has become increasingly dependent on networked computers and the trend towards larger networks will continue. For example, there are enterprises today consisting of tens of thousands of hosts. Also, less secure networks clearly have larger network attack graphs. Each network host might have several exploitable vulnerabilities. When considered across an enterprise, especially given global internet connectivity, network attack graphs become potentially large (Ammann et al., 2005).

In order to further evaluate the performance of ParticleNAG, we randomly generated 14 large-scale network attack graphs, denoted by NAG_1, NAG_2, ..., NAG_{14}. For each network attack graph, we considered different values for the cardinalities of E and S, where E is the set of preventable exploits and S is the set of attack scenarios represented by the network attack graph.

In NAG_1, ..., NAG_7, attack scenarios consists of between 3 to 9 exploits, while in NAG_8, ..., NAG_{14}, attack scenarios consists of between 3 to 12 exploits. Table 4 shows the cardinality of the set of preventable exploits, the cardinality of the set of attack scenarios, and the average cardinality of attack scenarios for each generated large-scale network attack graph.

Network Attack Graph	Cardinality of the Set of Exploits ($\|E\|$)	Cardinality of the Set of Attack Scenarios ($\|S\|$)	Average Cardinality of Attack Scenarios
NAG_1	200	2000	6.01
NAG_2	400	4000	5.99
NAG_3	400	6000	5.99
NAG_4	600	6000	6.03
NAG_5	600	8000	5.95
NAG_6	800	8000	6.01
NAG_7	1000	10000	6.05
NAG_8	200	2000	7.55
NAG_9	400	4000	7.52
NAG_{10}	400	6000	7.48
NAG_{11}	600	6000	7.53
NAG_{12}	600	8000	7.55
NAG_{13}	800	8000	7.48
NAG_{14}	1000	10000	7.47

Table 4. Large-scale network attack graphs

Experimental results

We applied ParticleNAG for minimization analysis of the above large-scale network attack graphs. We performed 10 runs of the algorithm with different random seeds and reported the best cardinality and the average cardinality of critical sets of exploits obtained from these 10 runs. We also applied ApproxNAG (Sheyner et al., 2002); (Jha et al., 2002), AntNAG (Abadi & Jalili, 2006), and GenNAG (Abadi & Jalili, 2008) for minimization analysis of the above network attack graphs. As shown in Table 5, ParticleNAG outperforms all the algorithms referenced above and finds a critical set of exploits with less cardinality. On average, the cardinalities of critical sets of exploits found by ParticleNAG, AntNAG, GenNAG are, respectively, 10.77, 9.21, and 8.95 percent less than the cardinality of critical set of exploits of exploits found by ApproxNAG. Accordingly, we conclude that ParticleNAG is more efficient than ApproxNAG, AntNAG, and GenNAG.

In ParticleNAG experiments, the parameters were set to $c_1 = 2$, $c_2 = 2$, and $V_{max} = 4$, which are values commonly used in the binary PSO literature. The swarm size was set to $m = 20$ and the maximum number of iterations was set to $t_{max} = 100$.

Network Attack Graph	ParticleNAG		AntNAG		GenNAG		ApproxNAG
	Best	Average	Best	Average	Best	Average	
NAG_1	87	87.3	88	88.6	87	88.8	98
NAG_2	175	176.5	177	178.9	176	179.0	197
NAG_3	194	196.6	197	199.6	197	200.2	221
NAG_4	264	265.9	268	270.7	264	271.3	296
NAG_5	287	288.4	291	293.7	291	293.8	317
NAG_6	351	352.8	356	360.9	358	361.3	397
NAG_7	439	442.8	448	451.7	449	453.9	503
NAG_8	80	80.8	81	82.1	81	82.0	91
NAG_9	158	159.6	159	161.9	161	162.5	182
NAG_{10}	178	179.4	179	181.9	180	182.8	200
NAG_{11}	239	240.8	242	244.7	244	245.6	267
NAG_{12}	257	259	262	264.4	263	265.6	293
NAG_{13}	322	323.6	325	329.1	327	331.2	362
NAG_{14}	401	404	409	413.1	410	414.9	450

Table 5. The cardinality of critical set of exploits found by ParticleNAG, AntNAG, GenNAG, and ApproxNAG

Figures 7 to 10 show the progress of the average cardinality of the global best position of ParticleNAG, the global best solution of AntNAG, and the best chromosome of GenNAG in the experiments for minimization analysis of NAG_4, NAG_7, NAG_{12}, and NAG_{14}, respectively. As it can be seen in these figures, ParticleNAG is able to quickly converge to a good solution for large-scale network attack graphs and can maintain the balance between the exploration and exploitation reasonably well in comparison to AntNAG and GenNAG.

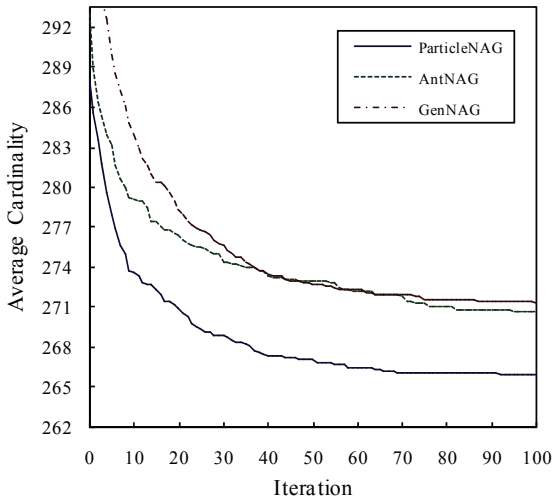


Fig. 7. Comparison of the performance of ParticleNAG, AntNAG, and GenNAG for minimization analysis of NAG_4

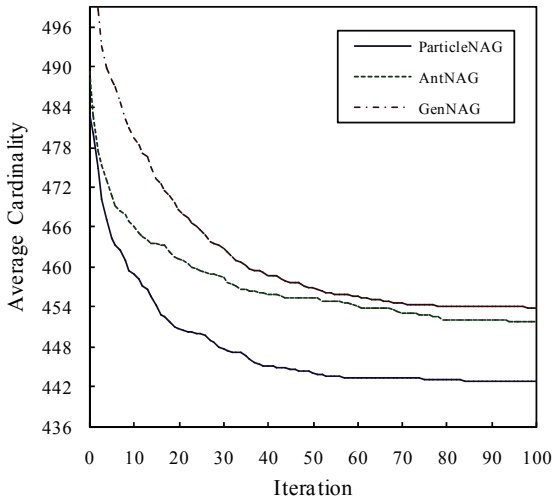


Fig. 8. Comparison of the performance of ParticleNAG, AntNAG, and GenNAG for minimization analysis of NAG_7

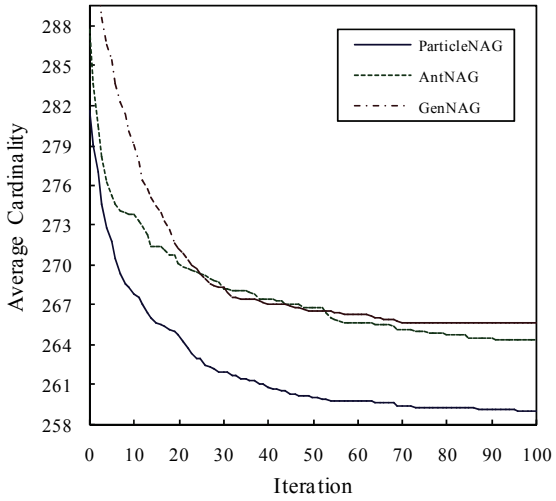


Fig. 9. Comparison of the performance of ParticleNAG, AntNAG, and GenNAG for minimization analysis of NAG_{12}

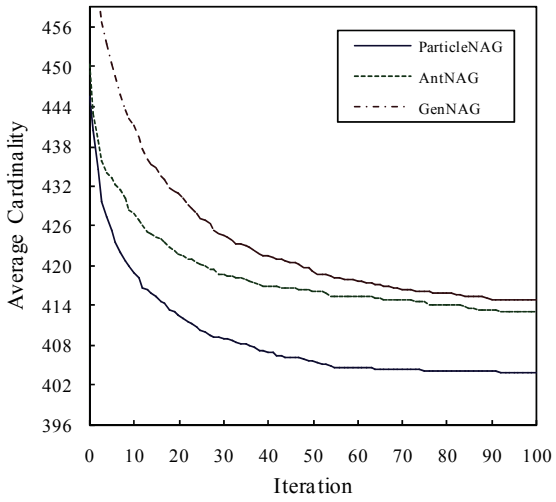


Fig. 10. Comparison of the performance of ParticleNAG, AntNAG, and GenNAG for minimization analysis of NAG_{14}

6.3 Algorithm parameters

We performed experiments to analyze the effect of different settings of parameters on the performance of ParticleNAG.

The effect of using the local search heuristic on the performance of ParticleNAG was analyzed by comparing the results of running the algorithm with and without the local search heuristic. Figures 11 and 12 show the progress of the average cardinality of the global

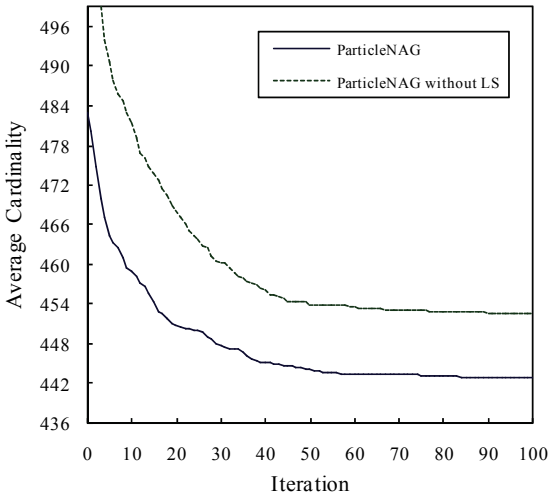


Fig. 11. Comparison of the performance of ParticleNAG and ParticleNAG without the local search heuristic for minimization analysis of NAG_7

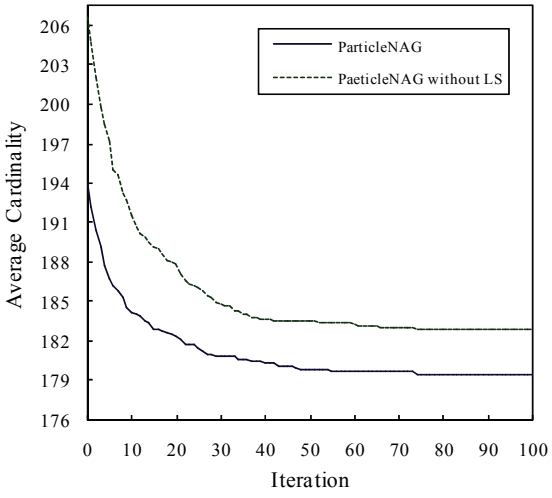


Fig. 12. Comparison of the performance of ParticleNAG and ParticleNAG without the local search heuristic for minimization analysis of NAG_{10}

best position, obtained from 10 runs of ParticleNAG and 10 runs of ParticleNAG without the local search heuristic in the experiments for minimization analysis of NAG_7 and NAG_{10}, respectively.

As the figures show, ParticleNAG significantly performs better than ParticleNAG without the local search heuristic and finds a critical set of exploits with less cardinality. This is because before updating the personal best position of a particle, its current position is improved by the local search heuristic. Hence, the personal best position of the particle shows a locally optimized solution.

To analyze the effect of the swarm size on the performance of ParticleNAG, the algorithm was run with the parameter settings from Section 6.2 but this time with the swarm size, m, set to 2, 5, 15, and 20, respectively.

As it can be seen in Table 6, when using a very small number of particles, ParticleNAG shows a poor performance. This is because the fewer the number of particles, the less the

Network Attack Graph	SwarmNAG			
	$m=2$	$m=5$	$m=15$	$m=20$
NAG_1	89.1	88.6	87.8	87.3
NAG_2	179.7	178.1	176.9	176.5
NAG_3	201.0	198.1	197.0	196.6
NAG_4	271.6	267.8	265.7	265.9
NAG_5	294.1	290.4	288.7	288.4
NAG_6	361.8	355.1	354.2	352.8
NAG_7	451.1	446.0	442.8	442.8
NAG_8	82.7	81.8	81.5	80.8
NAG_9	163.2	160.6	160.4	159.6
NAG_{10}	184.2	181.2	179.1	179.4
NAG_{11}	245.0	242.2	241.3	240.8
NAG_{12}	263.8	261.6	259.9	259.0
NAG_{13}	330.5	326.9	323.8	323.6
NAG_{14}	413.1	408.1	404.7	404.0

Table 6. Effect of the swarm size on the performance of ParticleNAG

exploration ability of the algorithm, and consequently the less information about the search space is available to all particles.

7. Conclusions

Each attack scenario is a sequence of exploits launched by an intruder for a particular goal. To prevent an exploit, the security analyst must implement a suitable countermeasure such as the firewall configuration or patch the vulnerabilities that made this exploit possible. The collection of possible attack scenarios in a computer network can be represented by a directed graph, called network attack graph. In this directed graph, each path from an initial node to a goal node corresponds to an attack scenario.

The aim of minimization analysis of network attack graphs is to find a minimum critical set of exploits/countermeasures so that by preventing/implementing them the intruder cannot reach his goal using any attack scenarios. This problem is in fact a constrained optimization problem in which the objective is to find a solution with minimum cardinality and the constraint is that the solution must be critical.

Several factors can make network attack graphs larger so that finding a minimum critical set of exploits/countermeasures becomes more difficult. An obvious factor is the size of the network under analysis. Our society has become increasingly dependent on networked computers and the trend towards larger networks will continue. Also, less secure networks clearly have larger network attack graphs. Each network host might have several exploitable vulnerabilities. When considered across an enterprise, especially given global internet connectivity, network attack graphs become potentially large.

Particle swarm optimization (PSO) is a population based stochastic optimization algorithm that was inspired by social behaviour of flocks of birds when they are searching for food.

While evolutionary algorithms take inspiration from biological evolution, memetic algorithms mimic cultural evolution. The term meme refers to a unit of cultural information that can be transmitted from one mind to another after reinterpretation and improvement that in the context of combinatorial optimization corresponds to local search.

In this paper, we presented a memetic particle swarm optimization algorithm, called ParticleNAG, for minimization analysis of network attack graphs. A greedy repair method was used to convert the constrained optimization problem into an unconstrained one. We reported the results of applying ParticleNAG for minimization analysis of 14 large-scale network attack graphs. We also applied an approximation algorithm, ApproxNAG (Sheyner et al., 2002); (Jha et al., 2002), an ant colony optimization algorithm, AntNAG (Abadi & Jalili, 2006), and a genetic algorithm, GenNAG (Abadi & Jalili, 2008), for minimization analysis of the above large-scale network attack graphs.

On average, the cardinality of critical sets of exploits found by ParticleNAG was 10.77 percent less than the cardinality of critical sets of exploits found by ApproxNAG. Also, ParticleNAG performed better than AntNAG and GenNAG in terms of convergence speed and accuracy.

We performed experiments to analyze the effect of swarm size and local search heuristic on the performance of ParticleNAG. The results of experiments showed that ParticleNAG significantly performs better than ParticleNAG without the local search heuristic.

8. References

Abadi, M. & Jalili, S. (2005). Automatic discovery of network attack scenarios using SPIN model checker, *Proceedings of the International Symposium on Telecommunications (IST 2005)*, pp. 81–86, Shiraz, Iran, September 2005

Abadi, M. & Jalili, S. (2006). An ant colony optimization algorithm for network vulnerability analysis. *Iranian Journal of Electrical & Electronic Engineering (IJEEE)*, Vol. 2, Nos. 3 & 4, pp. 106–120, July 2006

Abadi, M. & Jalili, S. (2008). Minimization analysis of network attack graphs using genetic algorithms. *International Journal of Computers and Their Applications (IJCA)*, Vol. 14, No. 4, pp. 263–273, December 2008

Ammann, P.; Wijesekera, D. & Kaushik, S. (2002). Scalable, graph-based network vulnerability analysis, *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 217–224, Washington, DC, USA, November 2002

Ammann, P.; Pamula, J.; Ritchey, R. & Street, J. (2005). A host-based approach to network attack chaining analysis, *Proceedings of the 2005 Annual Computer Security Applications Conference (ACSAC 2005)*, pp. 72–84, Tucson, AZ, USA, December 2005

Braendler D. & Hendtlass T. (2002). The suitability of particle swarm optimisation for training neural hardware, *Proceedings of the 15th International Conference on Industrial and Engineering, Applications of Artificial Intelligence and Expert Systems*, pp. 190–199, Cairns, Australia, June 2002

Deraison, R. (2010). Nessus Vulnerability Scanner. http://www.nessus.org

Eberhart, R. C.; Simpson P. & Dobbins R. (1996). *Computational Intelligence PC Tools*, Academic Press Professional, San Diego, CA, USA

Eiben, A. E. & Smith, J. E. (2003). *Introduction to Evolutionary Computing*, Springer-Verlag, Berlin, Germany

Engelbrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, Hoboken, NJ, USA

Hendtlass, T. & Randall, M. (2001). A survey of ant colony and particle swarm meta-heuristics and their application to discrete optimization problems, *Proceedings of the Inaugural Workshop on Artificial Life*, pp. 15–25, Adelaide, Australia, December 2001

Jha, S.; Sheyner, O. & Wing, J. M. (2002). Two formal analyses of attack graphs, *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, pp. 49–63, Cape Breton, Nova Scotia, Canada, June 2002

Kennedy, J. & Eberhart, R. C. (1995). Particle swarm optimization, *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, 1995

Kennedy, J. & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm, *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4104–4109, Orlando, FL, USA, October 1997

Kennedy, J.; Eberhart, R. C. & Shi Y. (2001). *Swarm Intelligence*, Morgan Kaufmann, San Mateo, CA, USA

Krasnogor, N.; Aragon, A. & Pacheco J. (2006). Memetic Algorithms, In: *Metaheuristic Procedures for Training Neural Networks*, Enrique Alba & Rafael Martí (Eds.), Springer-Verlag, Berlin, Germany

Mehta, V.; Bartzis, C.; Zhu, H.; Clarke, E. M. & Wing, J. M. (2006). Ranking attack graphs, *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, pp. 127–144, Hamburg, Germany, September 2006

Noel, S.; Jacobs, M.; Kalapa, P. & Jajodia, S. (2005). Multiple coordinated views for network attack graphs, *Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC 2005)*, pp. 99–106, Minneapolis, Minnesota, USA, October 2005

NuSMV. (2010). A New Symbolic Model Checker. http://afrodite.itc.it:1024/~nusmv/

NVD. (2010). National Vulnerability Database. http://nvd.nist.gov/

Ou, X.; Govindavajhala, S. & Appel, A. W. (2005). MulVAL: A logic-based network security analyzer, *Proceedings of the 14th USENIX Security Symposium*, pp. 8–8, Baltimore, MD, USA, August 2005

Ou, X.; Boyer, W. F. & McQueen, M. A. (2006). A scalable approach to attack graph generation, *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006)*, pp. 336–345, Alexandria, VA, USA, October 2006

Phillips, C. & Swiler, L. P. (1998). A graph-based system for network-vulnerability, *Proceedings of the New Security Paradigms Workshop*, pp. 71–79, Charlottesville, VA, USA, September 1998

Sheyner, O.; Haines, J. W.; Jha, S.; Lippmann, R. & Wing, J. M. (2002). Automated generation and analysis of attack graphs, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 273–284, Berkeley, CA, USA, May 2002

Shi, Y. & Eberhart, R. C. (1999). Empirical study of particle swarm optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1945–1950, Washington, DC, USA, July 1999

Shi, Y. (2004). Particle swarm optimization. *IEEE Connections*, Vol. 2, No. 1, pp. 8–13, February 2004

7

Quantum-Inspired Differential Evolutionary Algorithm for Permutative Scheduling Problems

Tianmin Zheng[1] and Mitsuo Yamashiro[2]
[1]*SoftAgency Co., Ltd*
[2]*Ashikaga Institute of Technology*
Japan

1. Introduction

In the world of combinatorial optimization, a wide range of combinatorial problems exist for which the solution is permutative, such as knapsack problem, traveling salesman problem (TSP), vehicle routine problem (VRP), quadratic assignment problem (QAP), dynamic pick-and-place (DPP) model of placement sequence and magazine assignment in robots and various types of scheduling problems. The objectives for these problems are usually to find the best sequence to realize the optimal, for example, to minimize the maximum completion time of jobs on machines or to find the shortest routing between several cities. These problems are very different from the continuous space problem because we are interested in sequence such as [1 2 3 4 5] which is permutative in nature. The solutions include a permutation-based sequence as well as the fitness. It is not feasible to solve this kind of problem using the approaches that solve only continuous problems.

As an important part of the permutation-based combinatorial optimization problems, the permutative scheduling problems (PSP) account for a large proportion of the production scheduling which is the core content of advanced manufacturing system. Among them, the flow shop scheduling problem (FSP) and job shop scheduling problem (JSP) may be the best known permutation-based scheduling problems. Both of FSP and JSP have earned a reputation for being a typical strongly NP-complete combinatorial optimization problem (Garey, *et al*., 1976) and have been studied by many workers due to their importance both in academic and engineering fields.

By now, the meta-heuristic algorithms achieve global or sub-optimal optima within acceptable time range are most popular for dealing with the permutation-based scheduling optimization problem like flow shop and job shop scheduling. These approaches are initiated from a set of solutions and try to improve these solutions by using some strategies or rules. The meta-heuristics for PSP include genetic algorithm (GA) (Reeves & Yamada, 1998; Goncalves, *et al*., 2005), immune algorithm (IA) (Doyen, *et al*., 2003; Xu & Li, 2007), tabu search (TS) (Nowicki & Smutnicki, 1996; Pezzella & Merelli, 2000), simulated annealing (SA) (Hisao, *et al*., 1995), ant colony optimization (ACO) (Ying & Liao, 2004; Zhang, *et al*., 2006), particle swarm optimization (PSO) (Tasgetiren, *et al*., 2004; Liao, *et al*., 2007; Xia & Wu, 2006), local search (Stützle, 1998), iterated greedy algorithm (Rubén & Stützle, 2007), differential evolution (Pan, *et al*., 2008), and other hybrid approaches (Zheng & Wang, 2003; Qian, *et al*., 2008; Hasan, *et al*., 2009). We should notice that these meta-heuristics can obtain satisfactory solutions, while

require more computation time and vary dramatically according to their structure and parameters. Recently, Han and Kim (2000, 2002, 2004) proposed some quantum-inspired evolutionary algorithms (QEAs) for the knapsack problem. However, due to its encoding and decoding scheme, the QEA can't directly be applied to permutation-based scheduling problems and the research of production scheduling problems based on QEA is just at beginning. Research of PFSP for minimizing the makespan of jobs based on QEA is first proposed by Wang, *et al*. (2005a, 2005b) and he made the simulations and proved that the QEA has better performance than NEH algorithm. Quite recently, Gu, *et al*. (2008) proposed a quantum genetic based scheduling algorithm for stochastic flow shop scheduling problem with the random breakdown and Niu, *et al*. (2009) put forward a quantum-inspired immune algorithm for hybrid flow shop with makespan criterion.

In our study, a novel quantum-inspired evolutionary algorithm called quantum-inspired differential evolutionary algorithm (QDEA) is applied to deal with the FSP and JSP. This chapter is divided into the following sections: section 2 presents the two different shop scheduling problems; in section 3, the basic quantum-inspired evolutionary algorithm is introduced and we put forward the algorithm framework based on QEA for solving the permutation-based scheduling problem. Then, each part of proposed algorithm framework is implemented by developing the novel QDEA which will be presented in section 4. In section 5, we make the simulation and comparisons of proposed QDEA with other algorithms for FSP and JSP. Finally, section 5 concludes the research.

2. Permutative scheduling problems

A flow shop is characterized by continuous and uninterrupted flow of jobs through multiple machines in series and the solution for FSP is the processing sequence of jobs. A FSP containing the same processing sequence of jobs for all machines is called as permutation FSP (PFSP). In the permutation FSP with J jobs and M machines, each job is to be sequentially processed on machine j = 1,2,…,n. At any time, each machine can process at most one job and each job can be processed on at most one machine. The sequence in which the jobs are to be processed is the same for each machine. Here we suppose $\pi = \{J_1, J_2,\ldots,J_n\}$ to be any a processing sequence of all jobs and suppose $c(J_i, k)$ and $t(J_i, k)$ to be the completion time and the processing time of job J_i on machine k, respectively. After initializing $c(J_1, 1) = t(J_1, 1)$, the mathematical formulae for the permutation FSP can be described as follows:

$$c(J_1, k) = c(J_1, k - 1) + t(J_1, k),\ k=2,\ldots,m \tag{1}$$

$$c(J_i, 1) = c(J_{i-1}, 1) + t(J_1, 1),\ i=2,\ldots,n \tag{2}$$

$$c(J_i, k) = max\{c(J_{i-1}, k), c(J_i, k-1)\} + t(J_i, k),\ i=2,\ldots,n,\ k=2,\ldots,m \tag{3}$$

so the scheduling objective such as minimizing the maximum completion time (makespan) which is most widely adopted can be described as:

$$C_{max} = c(J_n, m) \tag{4}$$

The PFSP with the makespan criterion is to find the permutation π^* in the set of all permutations $\prod$ satisfies the following criterion:

$$C_{max}(\pi^*) \le C_{max}(\pi)\ \ \forall\ \pi \in \prod \tag{5}$$

For a deterministic $n \times m$ JSP, the n job $J = \{J_1, J_2, \ldots J_n\}$ must be processed exactly once on each of m machines $M=\{M_1, M_2, \ldots M_n\}$. The processing of a job J_i on one machine M_j is called an operation $O_{i,j}$, and each opeation mush have an integral processing time $p_{i,j}$ ($p_{i,j} > 0$). There are two important constraints make the JSP different from other scheduing problems like PFSP: the operation precedence constraint and the machine processing constraint. The operation constraint means that once order of operations of job is fixed then the processing of an operation cannot be interrupted and concurrent; and the machine constraint is that only a single job can be processed at the same time on the same machine. We denote $C_{i,j}$ as the completion time for operation of job J_i on machine M_j, so the value $C_{i,j} = C_{i,k} + p_{i,j}$ is a completion time of $O_{i,j}$ in relation of which $O_{i,k}$ precedes to $O_{i,j}$ in processing order. For deterministic JSP, the $p_{i,j}$ is pre-set and the goal of scheduling in this study is to find the completion time $C_{i,j}$ for all $O_{i,j}$ to minimize the value of $C_{max} = Max(C_{i,k} + p_{i,j})$, in which the $O_{i,k}$ precedes to $O_{i,j}$ and C_{max} stands for the time used in completing all operations required.

3. Quantum-inspired evolutionary framework for permutative optimization

3.1 The basic quantum-inspired evolutionary algorithm

The quantum-inspired evolutionary algorithm (QEA) is based on the concept and principles of quantum computing, such as the quantum bit and the superposition of states. QEA can explore the search space with a smaller number of individuals and exploit the search space for a global solution within a short span of time. However, QEA is not a quantum algorithm, but a novel evolutionary algorithm (EA). Like any other EAs, QEA is also characterized by the representation of the individual, the evaluation function, and the population diversity. Inspired by the concept of quantum computing, QEA is designed with a novel Q-bit representation, a Q-gate as a variation operator, and an observation process based on Q-bits. QEA uses a novel Q-bit representation which is a kind of probabilistic representation. A Q-bit may be in the "1" state, in the "0" state, or in a linear superposition of two states. A Q-bit individual as a string of Q-bits is defined as:

$$q = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{bmatrix} \tag{6}$$

where $|a_i| + |\beta_i| = 1$, $i=1,2,\ldots,m$ (Han & Kim, 2000, 2002, 2004). If there is, for instance, a two Q-bits system with two pairs of amplitudes such as

$$q = \begin{bmatrix} -1/\sqrt{2} & 1/2 \\ 1/\sqrt{2} & -\sqrt{3}/2 \end{bmatrix} \tag{7}$$

then the states of the system can be represented as

$$-\frac{1}{2\sqrt{2}}|00\rangle + \frac{\sqrt{3}}{2\sqrt{2}}|01\rangle + \frac{1}{2\sqrt{2}}|10\rangle - \frac{\sqrt{3}}{2\sqrt{2}}|11\rangle \tag{8}$$

The above result means that the probabilities to represent the states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are 1/8, 3/8, 1/8, 3/8 respectively. By consequence, the two Q-bits system contains the information of four states. Evolutionary computing with Q-bit representation can provide a better characteristic of population diversity than other representations, since it can represent

linear superposition of states probabilistically. Only one Q-bit individual is enough to represent four states, but in binary representation at least four strings (00), (01), (10), (11) are needed. By observing the states of Q-bit, the Q-bit will collapse into $|0\rangle$ or $|1\rangle$. Then, a quantum chromosome with a length of 2 will become a binary string by a certain measurement method and we can use the binary string to solve the problem required.
The basic QEA includes 4 main steps (Han & Kim, 2000, 2002): initialization, observation, evaluation and updating. The procedure of basic QEA is described in the Figure 1:

```
procedure of QEA{
        t ← 0
        initialize Q(t)
        observe Q(t) and produce P(t)
        evaluate P(t)
        store the best solution b among P(t)
        do{
            t ← t + 1
            observe Q(t -1) and produce P(t)
            evaluate P(t)
            update Q(t)
            store the best solution b among P(t)
        }while(t< MAX_GEN))
}
```

Fig. 1. The pseudo code of basic QEA

When we adopt the QEA to deal with some problems, we firstly initialize the quantum population Q0 = [q0,1, q0,2,...,q0,n], where n is population size and the individual q0,i = [q0,i,1, q0,i,2,...,q0,i,m] is the quantum chromosome represented by two-state Q-bits, where m is the dimension of the problem. Then the step of observation makes binary solutions in P(t) by observing the states of Q(t), where P(t) = [xt,1, xt,2,...,xt,n]. One binary solution xt,j, j=1,2,…,n is a binary string of length, which is formed by selecting either 0 or 1 for each bit using the probability amplitudes. The procedure of evaluation is similar to other EAs by which each binary solution xt,j is evaluated to give a measure of its fitness and the optimum individual b will be stored. In this step of updating, Q-bit individuals in Q(t) are updated by applying quantum rotating gate defined as a variation operator of QEA, the following rotation gate is usually used as a basic Q-gate in QEA:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = U(\theta_i)\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{9}$$

$\theta_i = s(a_i, \beta_i)\Delta\theta_i$, where $\Delta\theta_i$ is the value of the rotation angle and $s(a_i, \beta_i)$ is the rotation direction.
As a conclusion, we can see in the basic QEA, the population provides plenty of diversity even in the small population and it can be easily mixed with other algorithms. In the QEA, the observation and updating are the core of the evolution. The observation operation determines the solution for the specific problem and updating leads the search towards the optimal. In this chapter, we propose a common algorithm framework for the permutative scheduling based on QEA and give an implement to this algorithm framework for solving FSP and JSP in the next section.

3.2 Quantum-inspired evolutionary framework for permutative scheduling problem

This study adopts the QEA to solve the permutative scheduling problem. As for the evolution-based algorithm like QEA, the population of individuals needs to be initialized according to the mechanism of QEA firstly, and effective updating operator also should be adopted to perform the evolution. Then, since the quantum chromosomes can not be used to represent the job permutations directly, a conversion rule is necessary to determine the representation of solutions which is permutative. At last, in order to improve the solution obtained through global search performed by QEA, some neighborhood based search can be performed on the permutative solution to get satisfactory results. The algorithm framework for permutation-based scheduling problems is shown in Figure 2. We will make a brief discussion about the four important parts shown in Figure 2 before implement the algorithm.

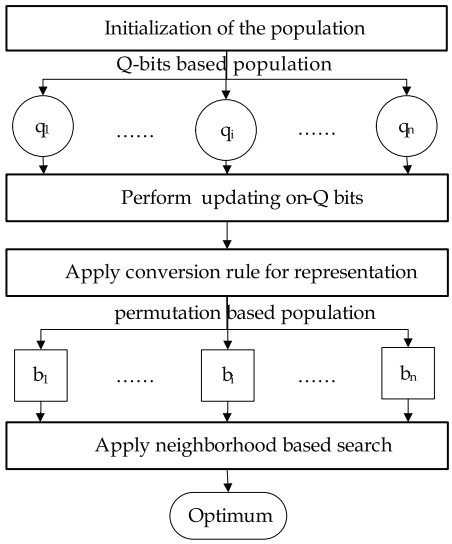


Fig. 2. The algorithm framework for permutation-based scheduling problems

1. The encoding scheme for Q-bits based population
   About how to encode the chromosome composed of several Q-bit, the equation 6 is most widely adopted for solving the scheduling problems (Wang, *et al*., 2005a, 2005b; Gu, *et al*., 2008; Niu, *et al*., 2009). In these researches, only one string of the probability amplitude is used to make the solution and the opposite one is just ignored. While, when we update the quantum individual, according to the normalization condition, these two probability amplitude strings should both be operated. If we just consider one string of probability amplitudes, the other one is a kind of waste. This study will introduce an encoding scheme by which we can make full use of quantum probability information and meet the normalization condition without any unnecessary operation.
2. The updating strategy for quantum evolution
   The updating strategy is the core of the QEA, and Han and Kim (2000, 2002, 2004) use the lookup table to perform the updating of quantum gate. In the lookup table, the value and direction of the rotating angle are determined by making the comparisons between the

current solution and the global best solution. We notice the change of rotating angle is a constant for all the problems, so it is easy to fall into local optimal since different problems has different characteristic. Also, the lookup operation costs excessive search time on the lookup table. The lookup table and its variants are also adopted by the researches (Wang, *et al*., 2005a, 2005b; Gu, *et al*., 2008; Niu, *et al*., 2009). This study adopts a new updating strategy which is more effective to replace the lookup table.

3. The conversion rule for representation of solution
   The decoding scheme is based on the encoding scheme and the specific problem. In the researches (Wang, *et al*., 2005a, 2005b; Gu, *et al*., 2008), the quantum chromosome is converted to job sequence for FSP by using the random key representation proposed by Bean (1994). In their approaches, the quantum chromosome is converted to the binary chromosome firstly, then to the decimal chromosome, and to the job order at last. Although this approach can convert the representation of quantum chromosome into job order, however, with the increasing of the problem scale, the length of corresponding quantum chromosome and the binary chromosome increases rapidly. In this study, we will introduce a simple but efficient way for conversion.
4. The neighbourhood based search for exploitation on job sequence
   Two schedules are neighbours if one can be obtained through a well-defined modification of the other. Neighbourhood search methods provide good solutions and offer possibilities to be enhanced when combined with other heuristics. These techniques continue to add small changes (perturbations) and evaluate schedules until there is no improvement in the objective function. Popular techniques that belong to this family include the tabu search (TS), simulated annealing (SA), genetic algorithm (GA), and so on. At each iteration, these procedures perform search within the neighbourhood and evaluate the various neighbouring solutions. The procedure accepts or rejects a solution as the next schedule based on a given acceptance-rejection criterion. This study will also adopt the neighbourhood based search to improve the solution quality.

Through this algorithm framework, the quantum chromosomes are converted into the job sequence and the algorithm can be designed for dealing with various types of permutation-based scheduling problem. The parts 2 and 4 can be easily replaced since we can use different updating strategies and neighbourhood based search for evolution. Also, the part 3 is problem-dependent and should be adjusted according to different scheduling problems. In the following section, we will propose a novel approach called QDEA by implementing each part of this algorithm framework for the PFSP and JSP both of which are the typical permutation-based production scheduling problem, and give the main procedure of proposed algorithm.

4. Proposed QDEA for permutative scheduling problem

In this section, based on the algorithm framework proposed above, we will implement each part shown in Figure 2 and give the description of proposed QDEA in detail.

4.1 The implement of the QDEA

4.1.1 The initialization of the population

According to the principles of the basic QEA, each Q-bit has two probability amplitudes, so the quantum chromosome consists of two strings of probability amplitude shown in equation 6. In this study, we suppose the quantum chromosome to be represented as:

$$q = [\theta_1 \quad \theta_2 \quad \ldots \quad \theta_n] \tag{10}$$

where θ_i is the quantum rotating angle with the range of $[0, \pi/2]$. We operate on the rotating angle and update it with the updating operation. In the decoding process, the solutions are converted according to the observation method proposed below.

Equation 6 has been adopted to solve the PFSP by Wang, *et al*. (2005a, 2005b), the stochastic FSP by Gu, *et al*. (2008) and hybrid FSP by Niu, *et al*. (2009). Although it is easy to understand and has been widely adopted for dealing with other problems, however, this kind of encoding scheme is not quite effective for the updating operation practiced by the quantum gate. Since the updating operation is the key of the QEA and the essence of the updating procedure is to influence the probability amplitude by changing the value of rotating angle, so we can directly practice on the quantum chromosomes represented in rotating angle. This provides a more effective way to deal with the Q-bits. Also, we can simplify the operations performed on the quantum chromosomes by using only one variable. Therefore, we consider that using the rotation angle to encode the quantum chromosome outperforms that of probability amplitude.

4.1.2 Differential evolution for updating

The differential evolution (DE) is a kind of population based stochastic optimization algorithm proposed by Storn and Price (1997, 1999), and also it's a kind of evolutionary algorithm which is similar to genetic algorithm. The DE adopts the real number encoding scheme, mutation, crossover and selection operation based on differential vectors and has excellent ability of overall search ability.

As with all other evolution based optimization algorithms, DE works with a population of solutions, not with a single solution for the optimization problem. Population x of generation g contains n solution vectors called individuals of the population and each vector represents potential solution for the problem. The population is often initialized by seeding it with random values within the given boundary constraints.

Suppose the population $Q_g = [q_{1,g}, q_{2,g}, \ldots, q_{n,g}]$ (n is the population scale, g is the current evolutionary generation), individual $q_{i,g} = [\theta_{i,1,g}, \theta_{i,2,g}, \ldots, \theta_{i,m,g}]$ (m is the dimension of the problem, $i \in [1, n]$). Suppose $v_{i,g+1}$ is the corresponding individual obtained by practicing the mutation operator on individual $q_{i,g}$, and one type of mutation operators works as:

$$v_{i,g+1} = q_{r1,g} + F(q_{r2,g} - q_{r3,g}) \tag{11}$$

where $r1, r2, r3 \in [1, n]$ and $r1 \neq r2 \neq r3 \neq i$; $q_{r1,g}$ is called father basic vector, $(q_{r2,g} - q_{r3,g})$ is called father differential vector; F is a real number and constant factor which controls the amplification of the differential variation.

In order to increase the diversity of the parameter vectors, we also use the $u_{i,j,g+1}$ ($j \in [1, m]$) vector which is obtained by practicing kinds of crossover operation between $q_{i,g}$ and mutative individual $v_{i,g+1}$ obtained by equation 11. The bin crossover we will use in this study is showed in equation 12:

$$u_{i,j,g+1} \begin{cases} v_{i,j,g+1} & for \quad rand < CR \quad or \quad j = Jrnd \\ \theta_{i,j,g} & otherwise \end{cases} \tag{12}$$

where CR is the crossover factor and $Jrnd$ is chosen randomly from the interval $[1, m]$.

We adopt the differential operation to perform the updating of the quantum chromosomes which is an innovation in this study. Since the quantum chromosomes are encoded in quantum angle, so the differential operations are directly practiced on the quantum angle and can provide the updating with excellent overall search ability and diversity.

4.1.3 The representation of permutation-based solution

For the decoding process of the quantum chromosome, since the solution to permutative scheduling problems is the order of all the elements (jobs for FSP, operations for JSP), therefore, we should convert the quantum chromosome encoded in rotating angle to job or operation sequences. In the decoding scheme adopted by Wang, *et al*. (2005a, 2005b) and Gu, *et al*. (2008), the representation needs several conversions (Q-bit chromosome → binary chromosome → decimal chromosome → job order) and the computation is complicated when the problem scale becomes larger. We put forward a simple but efficient strategy for conversion, which is also an innovation in this study.

- **Initialization**
  1. Obtain quantum chromosome $q_i = [\theta_{i,1}, \theta_{i,2},\ldots,\theta_{i,n}]$ from the Q-bit based population; Calculate $temp_i = [\cos\theta_{i,1}, \cos\theta_{i,2},\ldots,\cos\theta_{i,n}]$ and initiate two arrays *first*() and *last*().
- **Convertion**
  2. Generate a random number η between [0,1] and compare it with $\cos\theta_{i,e}$ where $e \in [1, n]$. If $\cos\theta_{i,e} > \eta$, put e into *first*() , else put e into *last*(). Repeat until all Q-bits in q_i are operated.
  3. Combine these two arrays *first*() and *last*() to one array *permutation*(), the *element* in *permutation*() is the permutative sequence for solution.
- **FSP representation**
  4. For PFSP, *permutation*() is the final solution.
- **JSP representation**

5. For JSP, get a element p from *permutation*() and perform the $code(i) = \text{mod}(p, m) + 1$, where the "mod" is an operator of calculating the reminder p being divided by m and m is the machine number. Repeat until all p in permutation() are operated. The *code*() is the solution for JSP.

Fig. 3. The procedure of converting mechanism for solution representation

For the solution reprentation of permutative problems, we define the rotating angles of element 1,2,…n are $[\theta_1, \theta_2,\ldots,\theta_n]$, that the probability amplitude of element i is $[\cos\theta_i, \sin\theta_i]$, then determine the permutative sequence according to the steps shown in Figure 3.

An example of JSP (suppose machine constrains to be [2-1; 1-2; 2-1] for 3 jobs) for converting mechanism is shown in Figure 4. For example, we have q_i = [0.87, 0.68, 0.15, 0.42, 1.38, 1.09], so the $temp_i$ = [0.64, 0.77, 0.98, 0.91, 0.18, 0.46]. Then generate η as 0.76 which is larger than $\cos\theta_{i,1}$, so we put '1' into *last*(). We continue generate η as 0.37 which is smaller than $\cos\theta_{i,2}$, so we put '2' into *first*(). After we operated on all 6 Q-bits, we have *first*() = [2 3 6] and *last*() = [1 4 5]. By combining these two arrays, we have *permutation*() = [2 3 6 1 4 5]. By applying the mod(p, m) + 1 operation, the JSP code becomes [3 1 1 2 3 2]. So each job number occurs m

times in the chromosome, and by reading the array *code*() from 1 to $n \times m$, the i-th occurrence of a job number refers to the i-th operation in the technological sequence of this job. Through the operation-based representation, any permutation can be decoded to a feasible schedule for JSP. In Figure 4, by scanning the elements in the job shop code, we can get the final schedule of $\{O_{312}, O_{112}, O_{121}, O_{211}, O_{321}, O_{221}\}$, where O_{ijk} means the j-th operation of job i is processed on the machine k.

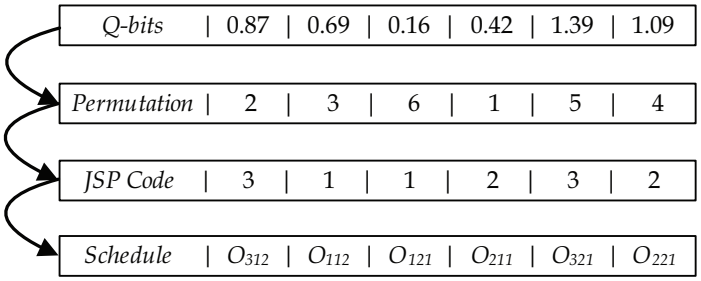


Fig. 4. The example of converting mechanism for JSP

Here, suppose we have a scheduling problem with the scale of 30 jobs and 10 machines and we can make a comparison like this way: for PFSP, since $2^4 < 30 < 2^5$, so the length of the quantum chromosome should be 30×5 = 150 at least by the method proposed by Wang, *et al*. (2005a, 2005b) and Gu, *et al*. (2008), and three conversions are needed to get the final solution. While, by our method, we just need a quantum chromosome with length of 30 and practice the conversion only once; for JSP, since the solution is the operation sequence, so the length of the quantum chromosome should be 30×10×5 = 1500 for these approaches, and five conversions are needed to get the operation. While, by our method, we just need a quantum chromosome with length of 300 and practice the conversion only three times. Thus, the representation of the solution we proposed simplifies the decoding procedure for the quantum chromsomes greatly and can provide a more effective way to deal with permutative scheduling problems.

4.1.4 Hybrid QDEA with local search scheme

By adopting the proposed converting mechanism, the Q-bits based population can be converted to permutative-based solution for scheduling effectively, so various types of neighborhood based search can be easily embedded to develop effective hybrid algorithms. For the permutative-based optimization problems, the *Insert*, *Interchange*, *Swap*, *2-opt* and *Or-opt* neighborhoods are often adopted. According to the analysis from Schiavinotto and Stützle (2007), for the permutative-based search landscape, using *Insert* at most n-1 times, one solution can transit to any other solution. Compared with other commonly used neighborhoods, the diameter of *Insert* is one of the shortest ones, so here we adopt the following local search scheme based on *Insert* neighborhood to perform thorough exploitation in the promising permutation-based solutions.

We obtain the global best chromosome *Best_g* and suppose n to be the element (job for PFSP and operation for JSP) number for a special problem. The pseudo code of the local search adopted is given as follows:

- **Initialization**
  1. Initialize counter t = 0, calculating makespan of Best_g as M(Best_g).
- **Perform the search**
  2. **Remove.** Generate i randomly and remove i-th job in *Best_g* and obtain a partial sequence *temp*.
  3. **Insert.** Insert the removed job into the best position $j(j \neq i)$ in *temp* and calculate $M(Best\_g)$.
- **Stopping condition check**
  4. If $M(temp) > M(Best\_g)$ and $t < n^{1/2}$, then $t \leftarrow t + 1$ and go to step 2.
- **Update**
  5. If M(temp) < M(Best_g), update Best_g = temp.

Fig. 5. The procedure of local search

Through the local search by using *Insert* neighborhood, we can obtain two job sequences stand for the one before the operation and the one after the operation, the better one is saved for the next generation iteration. It is should be noticed that the local search is directly applied on the job permutation, not on the quantum chromosomes. So after the whole local search completes, the corresponding Q-bit chromosome should be repaired since this quantum chromosome will be used to perform the next updating by DE and should match the permutation results obtained by the local search. The repair operation is simple since we just need to exchange the corresponding position of the Q-bits.

element	**1**	**2**	**3**	**4**	**5**	**6**
q_i	0.87	0.69	0.16	0.42	1.39	1.09
p_i	2	3	6	1	4	5
$q_{i'}$	0.87	**1.09**	0.16	0.42	1.39	**0.69**
$p_{i'}$	2	5	6	1	4	3

Table 1. An example of repair for local search

For example, before the local search, the Q-bit chromosome q_i and corresponding element permutation p_i obtained by conversion are shown in Table 1. After the local search, the element permutation $p_{i'}$ becomes [2 5 6 1 4 3] in which the '3' and '5' change the 2-th and 6-th position according to the *Insert* neighborhood, so we make the repair by changing the corresponding 2-th and 6-th Q-bits in q_i and obtain the $q_{i'}$= [0.87, 1.09, 0.16, 0.42, 1.39, 0.69]. The $q_{i'}$ will be adopted to perform the updating by DE strategy in the next iteration.

4.2 The main procedure of QDEA

To implement the common algorithm framework introduced in section 3.2, we adopt the DE strategy to perform the updating of quantum gate and introduce an effective converting mechanism for representing the permutative solution. In this way, we propose the QDEA for PFSP and JSP. Also, we develop the hybrid QDEA (HQDEA) by embedding the local search scheme to perform the neighborhood based search. The main procedure of QDEA (along with HQDEA) is as follows:

- **Initialize control parameters**
  1. Set the value of control parameters for DE and the lower and upper of angle for the QEA. Set maximum evolution generation *iter* and initialize iteration counter $t = 0$.
- **Initialize the population**
  2. Determine initial population $Pop_0 = [chrom_{0,1}, chrom_{0,2},..., chrom_{0,n}]$, where $chrom_{0,i} = [\theta_{0,1}, \theta_{0,2},..., \theta_{0,m}]$, n is the population scale, m is the number of jobs and $\theta_{0,j} \in [0, \pi / 2]$. In this step, the initial quantum chromosomes are generated randomly.
- **Make the solution**
  3. Adopt the converting mechanism to make the solution for permutation-based problem from the Q-bits based population.
- **Evaluate the population**
  4. Obtain objective values by evaluating Pop_0, store the best one into $Best_0$; store the best individual into $Best\_\theta$ and best element sequence into $Best\_g$. In this step, we perform the evaluation operation based on element sequence and get the objective values by calculating the permutative solution.
- **Perform the evolution**
  5. Update the Pop_{t-1} to Pop_t by using PSO strategy. In order to provide the search with excellent diversity and guarantees the normalization of Q-bits, we need to make sure that the individuals in Pop_t should be in the range between the lower(0) and upper(0.5π), so if any individual is out of this range, it should be revised by using θ_t = lower + rand × (upper − lower).
  6. Adopt the converting mechanism to make the solution for permutation-based problem from the Q-bits based population.
  7. Evaluate Pop_t and get objective values, compare with the corresponding solution in $Best_{t-1}$ and store the better one to $Best_t$. Update the $Best\_\theta$ and $Best\_g$.
  8. Practice the local search operations on $Best\_g$ by using the *Insert* operator and make the repair of the Q-bit chromosomes.
- **Stopping condition check**
  9. If the stopping condition $t >$ *iter* is met or the optimum is found, output the optimum; else $t \leftarrow t + 1$ and go to step 5**.**

Fig. 6. The main procedure of proposed QDEA

When perform the evolution, the updating of PSO is practiced on the quantum chromosome encoded in rotating angle and the local search is practiced on permutative element sequence, respectively. It should be noticed that for the updating performed by the PSO, we first need to save the $Best\_\theta$ as θ_gbest along with the v_t and θ_pbest_t for each Q-bit before we perform the updating next time.

5. Simulations and comparisons

The validation of proposed QDEA and HQDEA is conducted on the two demanding problems of PFSP and JSP which both are the typical permutative scheduling problems. Each experiment is conducted in two phases. The first phase is to introduce the benchmark and measure adopted in the simulations and to experimentally obtain the operating parameters for QDEA. The second phase is the comparison of the QDEA and HQDEA with other established approaches reported in the literature.

5.1 Simulations and comparisons on PFSP

5.1.1 Preparations for simulation

To test the performance of the proposed QDEA and HQDEA for PFSP, computational simulation is carried out with some well-studied benchmarks. In this study, four problem sets are selected. The first eight problems are called car1, car2 through car8 by Carlier (1978). The second 21 problems are called rec01, rec03 through rec41 by Reeves (1995). The third 110 problems are from Taillard (1993) and the last problem sets are called DMU problems from Demirkol, *et al*. (1998). Thus far these problems have been used as benchmarks for study with different methods by many researchers.

In order to make comparisons by using different methods, we adopt the following measures widely used in other literatures:

1. *RE*: the Relative Error of the average solution after we run the algorithm n times. The BRE and ARE stand for the best and average relative percentage error to the *Opt*, where *Opt* denotes the optimal solution value known thus far. The BRE and ARE can be calculated as equation 13 (the *S* is short for solution):

$$BRE(ARE) = \sum_{r=1}^{n}\left(\frac{S_{best,average} - Opt}{Opt} \times 100\right) \times \frac{1}{n}(\%) \tag{13}$$

2. *APRD*: the percentage relative deviation (PRD) of a best solution value P_{best} found by an algorithm from the optimal solution P_{opt} can be calculated as:

$$PRD = (P_{best} - P_{opt}) \,/\, P_{opt} \times 100\% \tag{14}$$

   and *APRD* is the average *PRD* values for a set of instances.

3. *ARPI:* the relative percentage increase (PRI) is defined as: suppose the best solution value found by an algorithm A denoted as P_A, and the best solution value found by n algorithms (include algorithm A) denoted as P_k, k = 1,2,...,n, then the PRI can be calculated as:

$$PRI = (P_A - \min(P_k, k = 1,2,...,n)) \,/\, \min(P_k, k = 1,2,...,n) \times 100\% \tag{15}$$

   and *APRI* is the average *PRI* values for a set of instances.

Parameter selection may influence the quality of the results. For the differential evolutionary strategy, two parameters should be set properly, one is the crossover factor *CR* and the other is the weight factor *F*. In this study, we will make a comparison between the 25 combination of $CR \in \{0.1,0.3,0.5,0.7,0.9\}$ and $F \in \{0.1,0.3,0.5,0.7,0.9\}$. In order to determine the appropriate values of parameters, a preliminary simulation is performed on 8 selected instances from the Rec benchmark problem set. For the 25 group data, we run 20 times for each group and calculate the average solution. We find that with the different group of parameters, the results vary dramatically and what we want to do is to choose a group of *F*/*CR* with the good and stable performance for all of 8 problems. The computational results indicate that the best solution is obtained at values $F = 0.1$ and $CR = 0.9$, so these values are adopted for all further experiments in this study.

At the same time, the different evolutionary strategy leads to different performance. For the PFSP, we make the simulation by comparing the performance of different evolutionary strategies proposed by Storn and Price (1999) in the form of *DE*/*x*/*y*/*z*, where x determine the vector to be operated is randomly generated or the best one, i.e. rand/best; y means the

number of the differential variables, i.e. 1/2 and here z stands for the *exp* or *bin* crossover. In this preliminary simulation, we want to make comparisons by using 10 evolutionary strategies: $DE1,2 \rightarrow DE/best/1/z$; $DE3,4 \rightarrow DE/rand/1/z$; $DE5,6 \rightarrow DE/rand-to-best/1/z$; $DE7,8 \rightarrow DE/rand/2/z$ and $DE9,10 \rightarrow DE/best/2/z$, where z has two values for crossover. After we made the simulation, we notice that for the F/CR = 0.1/0.9, the *DE7* strategy with the *bin* crossover provides the best performance for different benchmark problems, so we choose the *DE7* strategy to perform the updating of the quantum chromosomes.

In the following QDEA algorithm, we set F = 0.1 and CR = 0.9, differential evolutionary strategy by *DE7* and maximum iterative time I_{max} = 500. Based on the preliminary simulations, these values will give high probability to obtain better solutions. To test the performance of the proposed QDEA and HQDEA, computational simulation is carried out with the selected benchmark problems. The Visual C++ 6.0 is used to program the algorithm and all the computations are conducted on a Celeron 1.59 GHZ with 512 MB memory. We run the algorithm 20 times for each problem and use the statistical results for discussion.

Since many meta-heuristics and hybrid meta-heuristics have been adopted for solving the PFSP in literature, we want to make some comparisons to demonstrate the superiority of our QDEA. In this section, two types of recent and effective particle swarm optimization algorithm will be compared with QDEA firstly. Then we make a comparison between QDEA and quantum-inspired genetic algorithm (QGA) for both single objective and multi-objective PFSP to show the effectiveness of the proposed encoding scheme, converting mechanism and updating operation. And we also want to show our hybrid QDEA can obtain better optimization results than other hybrid algorithms.

5.1.2 Comparison of QDEA with CPSO and DPSO

Particle swarm optimization has two versions: continuous PSO (CPSO) and discrete PSO (DPSO). In this section, continuous PSO proposed by Tasgetiren, *et al*. (2004), discrete PSO proposed by Liao, *et al*. (2007) are compared with QDEA. In the CPSO proposed by Tasgetiren, *et al*., they proposed a SPV rule for solution representation and adopted the local search and mutation operation to avoid the premature convergence; in DPSO, the evolution is performed by defining the discrete particle and velocity trail, and the construction of a particle sequence is proposed for the PFSP. The experiments are conducted on the DMU problems from Demirkol, *et al*. (1998) (available from http://cobweb.ecn.purdue.edu/~uzsoy2/benchmark/fcmax.txt) in accordance with these two algorithms. In these 40 benchmark instances, eight combinations with number of machines m = 15, 20 and number of jobs n = 20, 30, 40, 50 are randomly generated and the best upper bounds for these instances are also provided by authors. We use the QDEA to minimize the makespan of jobs and the simulation results are given in Table 2.

In Table 2, the results are shown in the form of APRD. To be fair, the APRD of CPSO, DPSO and QDEA are calculated within the same running time for each problem sets (Liao, *et al*. 2007): [1.25s, 1.55s, 3.30, 3.95s, 6.40s, 7.60s, 11.00s, 12.90s] by setting the population size *Np* and number of iterations *iter*. From the results, we can see that DPSO is superior to CPSO for most of problem sets both in average (Avg) and minimum (Min) APRD, and our QDEA is much better than DPSO by comparing the APRD. Especially for Min APRD, the proposed QDEA has overwhelming superiority than DPSO which means our algorithm can obtain minimal makespan than DPSO. This simulation clearly demonstrates the excellent population diversity and unique optimization performance of Q-bits based search.

$n\times m$	CPSO by Tasgetiren, *et al*			DPSO by Liao, *et al*			Proposed QDEA		
	Np×*iter*	Avg	Min	*Np*×*iter*	Avg	Min	*Np*×*iter*	Avg	Min
20×15	40×2,500	−6.54	−7.68	100×1,000	−6.47	−7.93	20×2,000	−7.37	−11.65
20×20		−4.93	−6.20		−4.92	−6.20		−6.25	−8.11
30×15	60×2,500	−7.22	−8.75	150×1,000	−7.37	−9.05	30×2,000	−7.96	−11.06
30×20		−5.67	−7.44		−5.79	−7.56		−6.52	−11.18
40×15	80×2,500	−7.80	−9.31	200×1,000	−8.06	−9.74	40×1,000	−8.16	−11.26
40×20		−5.60	−7.39		−5.61	−6.87		−5.43	−7.97
50×15	100×2,500	−6.47	−7.70	250×1,000	−6.71	−7.92	50×1,500	−6.68	−8.95
50×20		−7.23	−8.81		−7.18	−8.43		−7.20	−9.43
Average		−6.40	−7.86		−6.44	−7.86		−6.94	−9.95

Table 2. Results of testing two PSOs and QDEA

5.1.3 Comparison of QDEA with QGA for single objective PFSP

To show the effectiveness of the coding scheme and the updating strategy proposed in this study, we want to compare the QDEA with the quantum-inspired genetic algorithm (QGA) developed by Wang, *et al*. (2005a) based on the Car and Rec benchmark problems (available from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt). The QGA adopts the random key to represent the solution and the lookup table to perform the updating which are all discussed in section 3.2. We firstly make the comparison on the single objective of minimizing the makespan and the results are shown in Table 3.

From Table 3, we can see the QDEA is overwhelming over the QGA for all the Car and Rec problems and even the AREs of QDEA are better than the BREs of QGA for most of instances. Since we do not perform the local search on the permutative solutions and the differences between QDEA and QGA are the coding scheme and the updating strategy only, so we can conclude the coding scheme and the updating strategy proposed in this study is more suitable for dealing with the permutation-based scheduling problems like the PFSP. The allowed running times (in second) of QDEA are also listed in the Table 3 for reference.

5.1.4 Comparison of QDEA with QGA for multi-objective PFSP

In the real world manufacturing environment, practical problems often involve multiple objectives that need to be considered concurrently, both from a process planning and a scheduling perspective. To apply the proposed QDEA to multi-objective PFSP and compare the performance with QGA (Wang, *et al*., 2005a), we conduct an experiment inspired from the research made by Sridhar and Rajendran (1996). They developed a genetic algorithm for the PFSP with triple objectives—makespan, total flow time, and total machine idle time. A DELTA operator is used to determine whether the parents should be replaced by the children and a single solution with equal weights for the three objectives is finally produced. Based on equation 1 to equation 3, the total flow time C_{sum} and total machine idle time I_{sum} can be obtained by calculating the $C_{sum} = \sum c(J_i, m)$ and $I_{sum} = \{ c(J_1, k-1) + \sum\{\max\{ c(J_i, k-1) - c(J_{i-1}, k) , 0\}\} | k = 2,...,m\}$ for i from 1 to n. Both evaluation operation and the updating operation of QDEA and QGA should be modified for dealing with the multi-objective problems. To establish a measure for these triple objectives, we include a modified version

P	N,M	C_{max*}	QGA		QDEA		
			BRE	ARE	BRE	ARE	Time
Car1	11,5	7,038	0.00	0.00	0.00	0.00	0.41
Car2	13,4	7,166	0.00	1.90	0.00	0.00	0.47
Car3	12,5	7,312	1.09	1.65	0.00	0.07	0.44
Car4	14,4	8,003	0.00	0.06	0.00	0.00	0.50
Car5	10,6	7,720	0.00	0.11	0.00	0.10	0.38
Car6	8,9	8,505	0.00	0.19	0.00	0.15	0.34
Car7	7,7	6,590	0.00	0.00	0.00	0.00	0.28
Car8	8,8	8,366	0.00	0.03	0.00	0.00	0.33
Rec01	20,5	1,247	2.81	6.79	0.00	0.47	0.87
Rec03	20,5	1,109	0.45	3.87	0.00	0.46	0.89
Rec05	20,5	1,242	2.25	3.00	0.24	0.45	0.88
Rec07	20,10	1,566	1.05	4.67	0.00	1.20	1.25
Rec09	20,10	1,537	4.03	6.40	0.65	2.70	1.23
Rec11	20,10	1,431	6.08	8.79	0.42	2.24	1.22
Rec13	20,15	1,930	5.08	7.98	1.64	3.17	1.55
Rec15	20,15	1,950	3.49	5.93	1.17	3.11	1.56
Rec17	20,15	1,902	6.51	9.10	2.35	4.20	1.55
Rec19	30,10	2,093	7.98	9.80	2.79	5.28	2.39
Rec21	30,10	2,017	6.94	10.05	2.01	4.19	2.38
Rec23	30,10	2,011	9.10	10.55	3.67	4.99	2.39
Rec25	30,15	2,513	7.16	10.06	3.18	5.10	3.13
Rec27	30,15	2,373	7.63	11.05	3.33	4.65	3.14
Rec29	30,15	2,287	12.42	14.06	2.61	6.01	3.15
Rec31	50,10	3,045	9.82	12.68	5.21	7.10	5.90
Rec33	50,10	3,114	6.20	9.54	1.10	3.48	5.81
Rec35	50,10	3,277	4.21	6.52	0.89	3.18	5.91
Rec37	75,20	4,951	15.54	17.49	8.15	9.10	7.80
Rec39	75,20	5,087	13.50	15.49	5.90	7.43	7.79
Rec41	75,20	4,960	16.92	18.84	8.10	9.02	7.79
AVE			5.18	7.12	1.84	3.03	2.47

Table 3. The comparisons between QDEA and QGA for minimizing makespan

of evaluation operation by assigning suitable weights to the three objectives in order to obtain a single solution. In this study, the weights are determined in accordance with Franminan, *et al*. (2002) to balance the effect of magnitude and the single solution is calculated as $1/3\times C_{max}\times n/2+1/3\times C_{sum}+1/3\times I_{max}\times n/10$. When to update the quantum

chromosome, the DELTA (Sridhar and Rajendran 1996) operator is used to make the comparison between the current solution (denoted as CS_t) and the previous best solution (denoted as PS_{t-1}) for t-th iteration of evolution. By evaluating CS_t and PS_{t-1}, the values of two makespans of $C_{t,max}$ and $C_{t-1,max}$, total flow times of $C_{t,sum}$ and $C_{t-1,sum}$, and total idle times of $I_{t,sum}$ and $I_{t-1,sum}$ can be obtained and the DELTA is defined as follows:

$$\text{DELTA} = w_1(C_{t,\max} - C_{t-1,\max}) / \min(C_{t,\max}, C_{t-1,\max}) + w_2(C_{t,\text{sum}} - C_{t-1,\text{sum}}) / \min(C_{t,\text{sum}}, C_{t-1,\text{sum}}) + w_3(I_{t,\text{sum}} - I_{t-1,\text{sum}}) / \min(I_{t,\text{sum}}, I_{t-1,\text{sum}}) \quad (16)$$

where $w_1 = w_2 = w_3 = 1/3$. So if $DELTA>0$, it indicates CS_t is better than PS_{t-1} and we update the corresponding Q-bits; otherwise, we keep PS_{t-1} as the best solution for this iteration.
We program both of the QDEA and QGA by using above evaluation and updating strategy. To have a fair comparison, we run these two algorithms in the same computer by using the same soft of Visual C++ 6.0 and within the same running time. The 90 Taillard's problems (Taillard, 1993) are adopted to perform the simulation and the PRI shown in equation 15 is used for the performance measure. The comparison results of the proposed QDEA and QGA along with the computation time for each problem set are summarized in Table 4.

$n\times m$	QDEA			QGA			computation time (s)
	C_{max}	C_{sum}	I_{sum}	C_{max}	C_{sum}	I_{sum}	
20×5	0.00	0.00	0.00	1.39	1.56	3.12	0.81
20×10	0.00	0.00	0.00	1.32	1.73	2.45	1.25
20×20	0.00	0.41	0.00	0.87	0.00	0.56	1.80
50×5	0.00	0.00	0.00	3.14	1.96	4.01	3.25
50×10	0.00	0.00	0.00	2.56	1.21	2.16	4.85
50×20	0.00	0.23	0.32	1.78	0.00	0.00	5.70
100×5	0.00	0.00	0.00	2.16	2.13	5.34	6.16
100×10	0.00	0.00	0.00	3.18	1.56	4.24	7.79
100×20	0.00	0.00	0.00	3.23	2.15	6.12	9.18
Average	0.00	0.07	0.04	2.18	1.37	3.11	4.53

Table 4. Results of testing two algorithms for multi-objective PFSP

We give the average value of PRI (APRI) for the three objectives. From the Table 4, it can be observed that for the multi-objective FSP, the proposed QDEA performes as well as the single one. Compared to the QGA, the QDEA can obtain the best value in general. Although for the criterion of total flowtime and total idle time, the QDEA is slightly inferior to QGA for several problem sets, the QDEA performs best among most of problems for these two criterions and all of problems for minimizing the makespan, which also demonstrates that the proposed QDEA has the prospects in the real world production scheduling applications.

5.1.5 Comparison of HQDEA with HGA, HQGA and HDE

To show the effectiveness of proposed hybrid QDEA embedded with the local search, we carry on comparisons with some popular hybrid algorithms. In this section, we make the comparisons between HQDEA and the hybrid genetic algorithm (HGA) proposed by Zheng

& Wang (2003), the hybrid quantum-inspired evolutionary algorithm (HQGA) proposed by Wang, *et al*. (2005b) and the hybrid differential evolution (HDE) algorithm proposed by Qian, *et al*. (2008) based on Car and Rec problems. HGA uses multi-crossover operators

P	N,M	C_{max}*	HGA		HQGA[a]		HDE		HQDEA	
			BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE
Car1	11,5	7,038	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Car2	13,4	7,166	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Car3	12,5	7,312	0.000	1.504	0.000	0.000	0.000	0.000	0.000	0.000
Car4	14,4	8,003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Car5	10,6	7,720	0.000	0.938	0.000	0.000	0.000	0.000	0.000	0.000
Car6	8,9	8,505	0.000	2.132	0.000	0.000	0.000	0.000	0.000	0.000
Car7	7,7	6,590	0.000	1.003	0.000	0.000	0.000	0.000	0.000	0.000
Car8	8,8	8,366	0.000	1.281	0.000	0.000	0.000	0.000	0.000	0.000
Rec01	20,5	1,247	0.160	0.192	0.000	0.140	0.000	0.144	0.000	0.112
Rec03	20,5	1,109	0.000	0.271	0.000	0.170	0.000	0.000	0.000	0.009
Rec05	20,5	1,242	0.242	0.628	0.240	0.340	0.242	0.242	0.242	0.242
Rec07	20,1	1,566	0.115	1.149	0.000	1.020	0.000	0.230	0.000	0.000
Rec09	20,1	1,537	0.605	1.627	0.000	0.640	0.000	0.000	0.000	0.000
Rec11	20,1	1,431	0.000	1.532	0.000	0.670	0.000	0.000	0.000	0.000
Rec13	20,1	1,930	0.415	1.974	0.160	1.070	0.104	0.301	0.104	0.225
Rec15	20,1	1,950	0.615	2.385	0.050	0.970	0.000	0.308	0.000	0.158
Rec17	20,1	1,902	1.840	2.482	0.630	1.680	0.000	1.178	0.000	0.126
Rec19	30,1	2,093	1.113	2.676	0.290	1.430	0.287	0.559	0.287	0.435
Rec21	30,1	2,017	1.522	1.636	1.440	1.630	0.198	1.413	0.149	1.041
Rec23	30,1	2,011	0.497	2.188	0.500	1.200	0.448	0.482	0.348	0.597
Rec25	30,1	2,513	1.922	2.706	0.770	1.870	0.478	1.492	0.119	0.995
Rec27	301,	2,373	1.551	2.318	0.970	1.830	0.843	1.285	0.253	0.954
Rec29	30,1	2,287	2.610	3.629	0.350	1.970	0.306	0.791	0.000	0.824
Rec31	50,1	3,045	1.156	2.759	1.050	2.500	0.296	0.824	0.263	0.565
Rec33	50,1	3,114	0.450	1.188	0.830	0.910	0.000	0.434	0.000	0.297
Rec35	50,1	3,277	0.000	0.131	0.000	0.150	0.000	0.000	0.000	0.000
Rec37	75,2	4,951	4.312	5.096	2.520	4.330	1.818	2.727	1.717	2.771
Rec39	75,2	5,087	2.597	3.205	1.630	2.710	0.983	1.541	0.845	1.485
Rec41	75,2	4,960	4.133	5.599	3.130	4.150	1.673	2.649	1.190	1.965
AVE			0.892	1.801	0.502	1.082	0.265	0.572	0.175	0.428

Table 5. Results of testing three hybrid algorithms and HQDEA

[a] In HQGA, the results are accurate to the second decimal place.

acting on the divided subpopulations and replaces the classical mutation by SA; in HQGA, the Q-bit representation is converted to random key representation which genetic operation are practiced on, and a permutation-based genetic algorithm is also applied after the solutions are constructed; as for the HDE, it not only applies the parallel evolution mechanism of DE to perform effective exploration, but also adopts problem-dependent local search to perform exploitation. The statistic performances of the four algorithms for the criterion of minimizing makespan are given in Table 5.

From the Table 5, we can see that for the Car problems with small scale, the HGA, HQGA, HDE and HQDEA all can find the optimum; for the Rec problems with relatively large scale, HQDEA also provide us with better performance which means the BREs and AREs are much smaller than that of the HGA and better than or equal to HQGA and HDE for all the problems. Also, we can notice that for the problem Rec01, Rec03, Rec07, Rec09, Rec11, Rec15, Rec17, Rec29, Rec33 and Rec35, the HQDEA has found the best solution known up to now. So the proposed HQDEA is a novel and effective approach for the PFSP.

5.2 Simulations and comparisons on JSP

5.2.1 Preparations for simulation

To test the performance of the proposed HQDEA for JSP, computational simulation is carried out with some well-studied benchmarks. In this study, 43 benchmarks (available from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html) are selected. The first 3 problems are called FT06, FT10 and FT20. The other 40 problems are called LA01, LA02 through LA40.

In order to evaluate the performance of different algorithms for JSP, the following four measures will be introduced:

1. Minimum makespan (MS): it is used for evaluating quality of solution. For the JSP, the minimum makespan a certain algorithm can achieve is usually adopted to prove the search ability of this algorithm.
2. Average convergence generation (CG): at each running of HQDEA, the optimal or the sub-optimal solution will be found after a number of generations. For several simulation replications, the number may be different, so this metric also reflects the average convergence speed of an algorithm.
3. Average compulation time (CT): the average computation time (in second) for an algorithm to find the optimal (or sub-optimal). Since different approaches run in different machines, the comparisons based on the CPU times might not seem to be meaningful. While, when make the comparison on the same PC, this metric can be used to show the effectiveness of an algorithm.
4. Relative error (RE): same to the definition of equation 13 in section 5.1.1

The parameters are set same to the preliminary simulation results given in section 5.1.1.

5.2.2 Comparison of QDEA with QGA

Firstly, we want to compare the proposed QDEA (without the local search operation) with the QGA developed by Wang, *et al*. (2005a) based on the three FT benchmark problems to show the effectiveness of the coding scheme and the updating strategy for JSP. We program both of these two algorithms with the same decoding procedure proposed in this study and run them on the same PC, we set the population size to be the number of job for each problem and the iteration time I_{max} to be 300 for both two algorithms, each instance runs 20 times and the results are shown in Table 6.

	FT06		FT10		FT20	
	QDEA	QGA	QDEA	QGA	QDEA	QGA
CG	187	204	192	178	217	254
MS	55	55	980	1125	1276	1465
RE	1.455	2.909	13.694	23.226	18.318	29.528
CT	0.480	1.208	4.518	19.888	4.478	19.795

Table 6. The results of QDEA and QGA on FT problems

From Table 6, we can see the QDEA is overwhelming over the QGA for all the three problems, especially for the FT10 and FT20, QDEA obtained much better makespan within about only 1/4 of the running time of QGA. Since we do not perform the local search on the permutative solutions and the differences between QDEA and QGA are the coding scheme and updating strategy only, so we can conclude that the coding scheme and updating strategy proposed in this study is also suitable for dealing with the JSP.

5.2.3 Comparison of HQDEA with HQGA

In order to check the effectiveness of proposed HQDEA with local search for JSP, we run the algorithm by combining the QDEA and local search and make the comparison with the HQGA proposed by Wang, *et al*. (2005b). The results are shown in Table 7. From Table 7, we notice the HQDEA can find all the optimums for three FT problems within 200 generations, while the HQGA can not achieve the optimum of FT10 and FT20 even spend more time and run more generations. The comparisons of relative error also shows the effectiveness of the propose HQDEA.

	FT06		FT10		FT20	
	HQDEA	HQGA	HQDEA	HQGA	HQDEA	HQGA
CG	1.1	9	132	221	148	276
MS	55	55	930	937	1165	1178
RE	0	0	1.785	4.430	0.961	3.867
CT	0.220	0.356	140.344	193.061	170.993	190.728

Table 7. The results of HQDEA and HQGA on FT problems

5.2.4 Comparisons between HQDEA with other approaches

To further show the effectiveness of HQDEA, we carry on some comparisons with other popular algorithms include the hybrid genetic algorithm (HGA) proposed by Goncalves, *et al*. (2005), memetic algorithm (MA) by Hasan, *et al*. (2009), tabu search (TSSB) by Pezzella & Merelli (2000), hybrid particle swarm optimization (HPSO) by Xia & Wu (2006) based on 40 LA benchmarks. We run the HQDEA using the same settings in section 5.2.2, and the results are shown in Table 8. In Table 8, the 'BKS' refers to the best solution found by now for each LA problems, 'average gap' is calculated as: $(MS-BKS)/BKS\times 100\%$ and 'No. of BKS obtained' means how many BKS can be found by an algorithm.

P	N,M	BKS	HGA	MA	TSSB	HPSO	HQDEA		
							MS	RE	CG
LA01	10,5	666	666	666	666	666	666	0.000	2
LA02	10,5	655	655	655	655	655	655	0.217	15
LA03	10,5	597	597	597	597	597	597	0.271	21
LA04	10,5	590	590	590	590	590	590	0.119	10
LA05	10,5	593	593	593	593	593	593	0.000	1
LA06	15,5	926	926	926	926	926	926	0.000	1
LA07	15,5	890	890	890	890	890	890	0.000	1
LA08	15,5	863	863	863	863	863	863	0.000	1
LA09	15,5	951	951	951	951	951	951	0.000	1
LA10	15,5	958	958	958	958	958	958	0.000	1
LA11	20,5	1222	1222	1222	1222	1222	1222	0.000	1
LA12	20,5	1039	1039	1039	1039	1039	1039	0.000	1
LA13	20,5	1150	1150	1150	1150	1150	1150	0.000	1
LA14	20,5	1292	1292	1292	1292	1292	1292	0.000	1
LA15	20,5	1207	1207	1207	1207	1207	1207	0.000	1
LA16	10,10	945	945	945	945	945	945	0.836	88
LA17	10,10	784	784	784	784	784	784	0.045	51
LA18	10,10	848	848	848	848	848	848	0.259	85
LA19	10,10	842	842	842	842	842	842	0.481	104
LA20	10,10	902	907	907	902	902	902	0.527	117
LA21	15,10	1046	1046	1079	1046	1047	1046	0.738	112
LA22	15,10	927	935	960	927	927	927	0.912	121
LA23	15,10	1032	1032	1032	1032	1032	1032	0.000	3
LA24	15,10	935	953	959	938	938	935	0.892	212
LA25	15,10	977	986	991	979	977	977	1.111	241
LA26	20,10	1218	1218	1218	1218	1218	1218	0.000	6
LA27	20,10	1235	1256	1286	1235	1236	1235	1.109	243
LA28	20,10	1216	1232	1286	1216	1216	1216	0.354	113
LA29	20,10	1157	1196	1221	1168	1164	1161	1.256	221
LA30	20,10	1355	1355	1355	1355	1355	1355	0.000	2
LA31	30,10	1784	1784	1784	1784	1784	1784	0.000	1
LA32	30,10	1850	1850	1850	1850	1850	1850	0.000	1
LA33	30,10	1719	1719	1719	1719	1719	1719	0.000	1
LA34	30,10	1721	1721	1721	1721	1721	1721	0.000	1
LA35	30,10	1888	1888	1888	1888	1888	1888	0.000	1
LA36	15,15	1268	1278	1307	1268	1269	1268	1.086	212
LA37	15,15	1397	1408	1442	1411	1401	1401	1.611	265
LA38	15,15	1196	1219	1266	1201	1208	1201	1.896	216
LA39	15,15	1233	1246	1252	1240	1240	1238	1.123	234
LA40	15,15	1222	1241	1252	1233	1226	1224	1.011	247
Average gap (%)			0.4190	1.0708	0.1091	0.0842	0.0404		
No. of BKS obtained			28	27	33	31	35		

Table 8. The comparisons between HQDEA and other algorithms

From Table 8, we can see for the LA01 to LA20, these 5 algorithms all can find optimum, especially for most problems, the HQDEA just needs to run the algorithm once for searching the optimization space. For the difficult problems with the middle and large scale, some problems remain to be unsolved in the provided evolution iteration. While, the HQDEA has obtained the 35 optimum out of 40 problems and achieved the minimum average gap of 0.0404 among these 5 algorithms. All these demonstrate that the HQDEA we proposed is a novel, effective and robust approach for JSP. The relative error s for HQDEA are also list in Table 8 for reference.

6. Conclusions and future research

In this study, we proposed an improved quantum-inspired evolutionary algorithm called quantum-inspired differential evolutionary algorithm (QDEA) for solving the flow shop scheduling and job shop scheduling problems with permutation-based solutions. Based on the QEA, we proposed a simple converting mechanism to determine permutative sequence based on quantum chromosome encoded in the form of rotating angle. Then we studied the applications of the QDEA by adopting the differential evolution strategy to perform the updating of quantum gate and local search to perform thorough exploitation in the promising permutative solutions. We adopt this novel QDEA to deal with the single objective and the multi-objective permutation FSP and to minimize the makespan of JSP. Compared to other algorithms, the simulation results demonstrated the effectiveness of our algorithm. For the PFSP, the QDEA performed better than two PSO based algorithms (Tasgetiren, *et al*., 2004; Liao, *et al*., 2007) and the QGA (Wang, *et al*., 2005a) for both of single objective and multi-objective problem; the proposed hybrid QDEA also provides better results than the hybrid algorithms include HGA (Zheng & Wang, 2003), HQGA (Wang, *et al*., 2005b) and HDE (Qian, *et al*., 2008); for the JSP, we also obtained satisfactory results by comparing QDEA with QGA and HQDEA with other state-of-the-art approaches (Goncalves, *et al*., 2005, Hasan, *et al*., 2009, Pezzella & Merelli, 2000, Xia & Wu, 2006). All these show the excellent diversity of the Q-bits based search and the effectiveness of the local search.

This study has made a step towards establishing an efficient heuristic for the permutation-based scheduling problems based on the quantum-inspired evolutionary algorithm. In this study, we propose a common algorithm framework for permutation-based scheduling problems, and the QDEA is in fact one implementation to this algorithm framework. As for the future research work, we can extend this study in the following ways. For the part 2 of algorithm framework, the parameter settings for the differential evolution strategy are worth examining in detail firstly. Then, developing new hybrid strategy by combining Q-bit based search and other evolution based methods to improve the performance also makes a great sense. For the part 4 of algorithm framework, we can use other strategies like variable neighbourhood search (VNS) to perform the neighbourhood based search and check the performance. At last, for the application of this research, the proposed QDEA approach can be extended to deal with flow and job shop scheduling problems with different constraints and performance criteria; also we can apply this new method to other permutation-based shop scheduling problems such as open shop scheduling problem (OSP) and make comparisons with other algorithms.

7. References

Bean J C. (1994) Genetics and random keys for sequencing and optimization. *ORSA Journal of computing*, 6(2), 154-160.

Bin Qian, Ling Wang, Rong Hu, Wan-Liang Wang, De-Xian Huang and Xiong Wang. (2008). A hybrid differential evolution method for permutation flow-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 38(5-6), 757-777.

Carlier J (1978) Ordonnancements a Contraintes Disjonctives. *Recherche Operationelle /Operations Research*, 12(4), 333-350.

Ching-Jong Liao,Chao-Tang Tseng and Pin Luarn. 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research*, 34(10), 3099-3111.

D. Zheng and L.Wang (2003). An Effective Hybrid Heuristic for Flow Shop Scheduling. The *International Journal of Advanced Manufacturing Technology*. 21(1), 38-44.

Demirkol E, Mehta S, Uzsoy R (1998) Benchmarks for shop scheduling problems. *Eur J Oper Res* 109:137–141.

Doyen A, Engin O, Ozkan C (2003). A new artificial immune system approach to solve permutation flow shop scheduling problems. *Tukish Symposium on Artificial Immune System and Neural Networks* TAINN'03.

Framinan JM, Leisten R, Ruiz-Usano R. (2002) Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. *European Journal of Operational Research*, 141, 559–69.

Garey M, Johnson D and Sethi R (1976) The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 24(1), 117-129.

Goncalves, J. F., Mendes, J. J. M., and Resende, M. G. C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1), 77–95.

Han K-H. (2000) Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem. In: *IEEE Proc. Of the 2000 Congress on Evolutionary Computation*，San Diego，USA IEEE Press, July 2000.

Han K-H，Kim J-H. Quantum-inspired Evolutionary Algorithm for a class of Combinatorial Optimization. *IEEE Trans on Evolutionary Computation*, 2002.

Han K-H，Kim J-H. Quantum-inspired Evolutionary Algorithms with a New Termination Criterion H,Gate and Two-Phase Scheme. *IEEE Trans on Evolutionary Computation* 2004.

Hisao Ishibuchi, Shinta Misaki and Hideo Tanaka (1995) Modified simulated annealing algorithms for the flow shop sequencing problem. *European Journal of Operational Research*. 81(2), 388-398.

Jinwei Gu, Xingsheng Gu, Bin Jiao. (2008). A Quantum Genetic Based Scheduling Algorithm for stochastic flow shop scheduling problem with random breakdown. *Proceedings of the 17th World Congress. The International Federation of Automatic Control* Seoul, Korea, July 6-11, 63-68.

Jun Zhang, Xiaomin Hu, X. Tan, J.H. Zhong and Q. Huang. (2006). Implementation of an Ant Colony Optimization technique for job shop scheduling problem. *Transactions of the Institute of Measurement and Control*, 28(1), 93-108.

Kuo-Ching Ying and Ching-Jong Liao (2004) An ant colony system for permutation flow-shop sequencing. *Computers & Operations Research*. 31(5), 791-801.

Nowicki E, Smutnicki C (1996) A fast tabu search algorithm for the permutation flow-shop problem. *Eur J Oper Res* 91,160–175.

Pezzella, F., & Merelli, E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120(2), 297–310.

QK Pan, MF Tasgetiren, YC Liang (2008) A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering*, 55(4), 795-816.

Qun Niu, Taijin Zhou, Shiwei Ma. (2009). A Quantum-Inspired Immune Algorithm for Hybrid Flow Shop with Makespan Criterion. *Journal of Universal Computer Science*, 15(4), 765-785.

Reeves, C R (1995) A genetic Algorithm for Flowshop Sequencing. *Computers and Operations Research*, 22(1), 5-13.

Reeves CR, Yamada T (1998) Genetic algorithms, path relinking and the flowshop sequencing problem. *Evol Comput* 6, 45–60.

Rubén Ruiz, and Thomas Stützle (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*. 177(3), 2033-2049.

S. M. Kamrul Hasan, Ruhul Sarker, Daryl Essam and David Cornforth. (2009). Memetic Algorithms for Solving Job-Shop Scheduling Problems. *Memetic Computing*, 1(1), 69-83.

Schiavinotto T, Stützle T. (2007). A review of metrics on permutations for search landscape analysis. *Computers Operations & Research*, 34(10), 3143–53.

Sridhar J, Rajendran C. (1996). Scheduling in flowshop and cellular manufacturing systems with multiple objectives—a genetic algorithmic approach. *Production Planning and Control*, 7, 374–82.

Storn R, Price K (1997) Differential evolution—a simple evolution strategy for fast optimization. *Dr. Dobb's Journal*, 78, 18–24.

Storn R, Price K. (1999). Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report*, TR-95-012, ICSI.

Stützle, T (1998) Applying iterated local search to the permutation flow shop problem. *Technical report*, AIDA-98-04, FG Intellektik, TU Darmstadt.

Taillard, E (1993) Benchmarks for basic scheduling problems. *European journal of operational research*, 64(2), 278-285.

Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G. (2004) Particle swarm optimization algorithm for makespan and maximum lateness minimization in permutation flowshop sequencing problem. In: *Proceedings of the fourth international symposium on intelligent manufacturing systems*, urkey: Sakarya; 431–41.

Xiao-dong Xu & Cong-xin Li. (2007). Research on immune genetic algorithm for solving the job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*. 34, 783–789.

Wang Ling, Wu Hao, and Zheng Da-Zhong (2005a) A quantum-inspired genetic algorithm for scheduling problems. *Lecture Notes in Computer Science*, v 3612, n PART III, Advances in Natural Computation: First International Conference, ICNC 2005. Proceedings, 417-423.

Wang L, Wu H, Tang F and Zheng DZ (2005b) A hybrid quantum-inspired genetic algorithm for flow shop scheduling. *Lecture Notes in Computer Science*, 3645, 636-644.

Wei-jun Xia, Zhi-ming Wu. (2006). A hybrid particle swarm optimization approach for the job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*. 29, 360–366.

8

Quantum-Inspired Particle Swarm Optimization for Feature Selection and Parameter Optimization in Evolving Spiking Neural Networks for Classification Tasks

Haza Nuzly Abdull Hamed[1,2], Nikola K. Kasabov[1] and Siti Mariyam Shamsuddin[2]
[1]*Auckland University of Technology*
[2]*Universiti Teknologi Malaysia*
[1]*New Zealand*
[2]*Malaysia*

1. Introduction

Particle Swarm Optimization (PSO) was introduced in 1995 by Russell Eberhart and James Kennedy (Eberhart & Kennedy, 1995). PSO is a biologically-inspired technique based around the study of collective behaviour in decentralized and self-organized animal society systems. The systems are typically made up from a population of candidates (particles) interacting with one another within their environment (swarm) to solve a given problem. Because of its efficiency and simplicity, PSO has been successfully applied as an optimizer in many applications such as function optimization, artificial neural network training, fuzzy system control. However, despite recent research and development, there is an opportunity to find the most effective methods for parameter optimization and feature selection tasks.
This chapter deals with the problem of feature (variable) and parameter optimization for neural network models, utilising a proposed Quantum–inspired PSO (QiPSO) method. In this method the features of the model are represented probabilistically as a quantum bit (qubit) vector and the model parameter values as real numbers. The principles of quantum superposition and quantum probability are used to accelerate the search for an optimal set of features, that combined through co-evolution with a set of optimised parameter values, will result in a more accurate computational neural network model. The method has been applied to the problem of feature and parameter optimization in Evolving Spiking Neural Network (ESNN) for classification. A swarm of particles is used to find the most accurate classification model for a given classification task. The QiPSO will be integrated within ESNN where features and parameters are simultaneously and more efficiently optimized. A hybrid particle structure is required for the qubit and real number data types. In addition, an improved search strategy has been introduced to find the most relevant and eliminate the irrelevant features on a synthetic dataset. The method is tested on a benchmark classification problem. The proposed method results in the design of faster and more accurate neural network classification models than the ones optimised through the use of standard evolutionary optimization algorithms.

This chapter is organized as follows. Section 2 introduces PSO with quantum information principles and an improved feature search strategy used later in the developed method. Section 3 is an overview of ESNN, while Section 4 gives details of the integrated structure and the experimental results. Finally, Section 5 concludes this chapter.

2. Particle swarm optimization

PSO is a population-based stochastic optimization technique. In common classifiers, PSO is a global optimization technique that is often used to seek a good set of weights. Similar to other evolutionary algorithms, PSO is initialized with a random population and searches for optimal solutions by updating the particles. Unlike Genetic Algorithms (GA), PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

To create a swarm of $i = 1,...,N$ particles, at all points in time, each particle *i* has

1. A current position X_i or $X_n = \left(x_{i1},...,x_{iD}\right)$,
2. A record of the direction it follows to get to that position V_i or $V_n = (v_{i1},...,v_{iD})$,
3. A record of its own best previous position $pbest = (pbest_1,...,pbest_D)$,
4. A record of the best previous position of any member in its group $gbest = (gbest_1,...,gbest_D)$.

Given the current position of each particle, as well as the other information, the problem is to determine the change in direction of the particles. As mentioned above, this is done by reference to each particle's own experience and its companions. Its own experience includes the direction it comes from V_i and its own best previous position. The experience of others is represented by the best previous position of any member in its group. This suggests that each particle might move in

1. The same direction that it comes from V_i,
2. The direction of its best previous position $pbest - X_i$,
3. The direction of the best previous position of any member in its group $gbest - X_i$.

The algorithm supposes that the actual direction of change for particle *i* will be a weighted combination of (Shi & Eberhart, 1998);

$$V_n = W_x V_n + C_1 * r_1 * (G_{best,n} - X_n) + C_2 * r_2 * (P_{best,n} - X_n) \quad (1)$$

where

r_1 and r_2 are uniform random numbers between $[0,1]$,
$C_1 \rangle 0$ and $C_2 \rangle 0$ are constants called the *cognitive* and *social* parameters, and
$w \rangle 0$ is a constant called the *inertia* parameter.

For successive index periods (generations), n and $n+1$, the direction of change, i.e., the new position of the particle will simply be:

$$X_n = X_n + V_n \quad (2)$$

Given the initial values of X_i, V_i, $pbest$ and $gbest$, Equation (1) and Equation (2) will determine the subsequent path that each particle in the swarm will follow. To avoid particles flying beyond the boundary, the velocities of each dimension are clamped to a maximum velocity, $V_{\max}$. If the sum of accelerations causes the velocity of that dimension to exceed $V_{\max}$, which is a pre-defined parameter, then the velocity is limited to $V_{\max}$.

Obviously, from the above procedure, it seems that PSO shares many common features with GA. Both algorithms start with a randomly generated population, and have a fitness function to evaluate the population. Both methods update the population and search for the optimum solutions with random techniques. However, PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity, and have memory as storage of history. In PSO, only *gbest* gives the information to others in the population, and it is a one -way information sharing mechanism. The evolution only looks for the best solution, and, in most cases, all the particles tend to converge to the best solution quickly.

2.1 Application in parameter optimization

In neural network models, an optimal combination of parameters can influence their performance. It is not feasible to manually adjust the parameters, particularly when dealing with different combinations for different datasets. Consequently, parameter optimization is vital and much research has been conducted on it (Bäck & Schwefel, 1993). Parameter optimization using PSO works when each particle in the swarm holds the parameter value. This value is considered a particle's position. Updating the particle's position means updating the parameter value. The process begins by initializing the population and all particles with random parameter values. Then, in every iteration, all particles' position are updated based on two factors; the *gbest* and *pbest* . Updating the parameter value based on these two values normally results in a better solution. This updating process is repeated in every iteration until stopping criteria is met, for example a desired fitness value or a maximum number of iterations.

2.2 Quantum inspired probability concept for feature selection

Feature optimization is considered as a crucial pre-processing phase in a classification task. This is because using a higher number of features does not necessarily translate into better classification accuracy. In some cases, having fewer significant features could help reduce the processing time and produce good classification results. Blum and Langley have classified the feature selection techniques into three basic approaches (Blum & Langley, 1997): Embedded approach adds or removes features in response to prediction error on new instances; Filter approach first selects features and then uses them in a classification model; Wrapper approach uses classification accuracy to evaluate features. However, the conventional PSO is inadequate for solving problems that require probability computation such as in the feature selection tasks. Therefore, the quantum information principle is embedded with principles of evolutionary computation in the PSO as a mechanism for feature probability calculation and consequent selection based on these probabilities.

Referring to (Narayanan, 1999; Kasabov, 2007), quantum computing principles have been seen as a source of inspiration for novel computational methods. Two famous quantum applications are factorisation problem (Shor, 1994) and Grover's database search algorithm (Grover, 1996).

According to the normal or classical computing concept, information is represented in bits where each bit must hold a value of either 0 or 1. However, in quantum computing, information is instead represented by a qubit in which a value of a single qubit could be 0, 1, or a superposition of both. Superposition allows the possible states to represent both 0 and 1 simultaneously based on its probability. The quantum state is modelled by the Hilbert space of wave functions and is defined as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3}$$

where α and β are complex numbers defining probabilities at which the corresponding state is likely to appear when a qubit collapses, for instance, when reading or measuring. Probability fundamentals stated that $|\alpha|^2 + |\beta|^2 = 1$, where $|\alpha|^2$ gives the probability that a qubit is in the OFF (0) state and $|\beta|^2$ gives the probability that a qubit is in the ON (1) state. The probability of $\begin{bmatrix}\alpha\\ \beta\end{bmatrix}$ can be represented as quantum angle θ, where $\begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix}$ satisfies the probability fundamental of $|\sin(\theta)|^2 + |\cos(\theta)|^2 = 1$. The θ parameter is normally been used in quantum inspired Evolutionary Algorithms (EA) to calculate and update probability. There are several studies where quantum computation acts as probability computation in EA. The Quantum Evolutionary Algorithm (QEA) was popularized by Han and Kim (Han & Kim, 2000). Since then, a lot of attention has been given by researchers around the world to this technique. Descended from the basic EA concept, QEA is a population-based search method which simulates a biological evolutionary process and mechanism, such as selection, recombination, mutation and reproduction. Each individual in a population is a possible solution candidate and is evaluated by a fitness function to solve a given task. However, instead of using normal real value values, information in QEA is represented in qubits. This probability presentation has a better characteristic of diversity than classical approaches. QEA have been reported to successfully solve complex benchmark problems such as numerical (da Cruz et al., 2006), multiobjective optimization (Talbi et al., 2006) and real world problems (Jang et al., 2004).

The quantum computation also has been extended to PSO and this is known as Quantum-inspired Particle Swarm Optimization (QiPSO) (Sun et al., 2004). The main idea of QiPSO is to update the particle position represented as a quantum angle θ. The common velocity update equation in conventional PSO is modified to get a new quantum angle which is translated to the new probability of the qubit by using the following formula:

$$\Delta\theta_n = w * \Delta\theta_{n_{t-1}} + c_1 * rand() * (\theta_{gbest_n} - \theta_n) + c_2 * rand() * (\theta_{pbest_n} - \theta_n) \tag{4}$$

Based on the new θ velocity, the new probability of α and β is calculated using a rotation gate as follows:

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \begin{bmatrix}\cos(\Delta\theta) & -\sin(\Delta\theta)\\ \sin(\Delta\theta) & \cos(\Delta\theta)\end{bmatrix}\begin{bmatrix}\alpha_{t-1}\\ \beta_{t-1}\end{bmatrix} \tag{5}$$

In a feature selection task, each qubit denoted as quantum angle θ, represents one feature. In this case, the collapse qubit value 1 represents features selected while value 0 represents those not selected.

2.3 Enhancement for parameter optimization and feature selection

There are some problems when using QiPSO algorithms for parameter optimization and feature selection. Therefore, this chapter proposes an improved QiPSO algorithm called Dynamic Quantum-inspired Particle Swarm Optimization (DQiPSO). The problems include

a possibility of missing the optimal parameter value when using only binary QiPSO. As the information is represented in a binary structure, the conversion from binary to real value will lead to such problems, especially if the selected number of qubits representing the parameter value is insufficient. To overcome this problem, a combination of QiPSO and conventional PSO is proposed. The DQiPSO particle is divided into two parts: the first part uses quantum probability computation for feature selection and another part holds the real value for parameters as shown in Figure 1. This method not only effectively solves this problem, but also eliminates one parameter which holds number of qubits representing the parameter value.

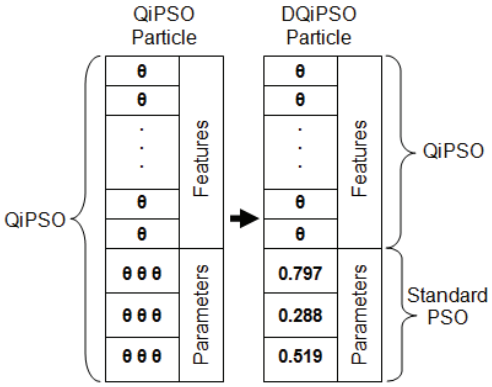


Fig. 1. The proposed hybrid particle structure in DQiPSO

In addition, the search strategy of QiPSO is based on random selection at the beginning of the process. Each particle will update itself based on the best solution subsequently found. A major problem with this approach is the possibility of not selecting the relevant features at the beginning; other particles in the entire process are thus affected. This is due to each particle updating its information without relevant features. Therefore, a new strategy is proposed in which five types of particles in the DQiPSO are considered. Apart from the normal particle, referred to as the Update Particle, which renews itself based on *pbest* and *gbest* information, four new types of particles are added to the swarm. The first type is the Random Particle, which will randomly generate new sets of features and parameters in every iteration to increase the robustness of the search. The second type is the Filter Particle, which selects one feature at a time and feeds it to the network and calculates the fitness value. This process is repeated for each feature. Any features with above average fitness will be considered as relevant. This method is targeted at linear separation problems. The third particle type is the Embed In Particle in which input features are added to the network one by one. If a newly added feature improves fitness, it will be considered a relevant feature. Otherwise, the feature will be removed. The final particle type is the Embed Out Particle which starts the identification process with all features fed to the network to get the initial fitness value. These features are gradually removed one by one. If removing a feature causes decrement of the fitness value, then this feature will be considered relevant and hence will be kept. Otherwise, the feature will be considered irrelevant and removed.

The main idea behind Filter, Embed In and Embed Out particles is to identify the relevance of each feature and to reduce the number of candidates until a small subset remains. For

subsequent iterations, features considered relevant will be selected randomly to find the best combination of significant features. This strategy helps to solve unevaluated relevant features, while reducing the search space and facilitating the optimizer in finding relevant features faster. Similar to the standard PSO in updating the particles, if a new particle is found to be the best solution, then it will be stored as a *gbest* . In this scenario, some improvements have also been proposed for the update strategy. This includes replacing the *gbest* particle with a new particle if the fitness value is higher or equivalent, but with a lower number of selected features. Due to the robust search space provided by DQiPSO, fewer particles are needed to perform the optimization tasks; hence, less processing time can be achieved. The structure of this strategy is illustrated in Figure 2.

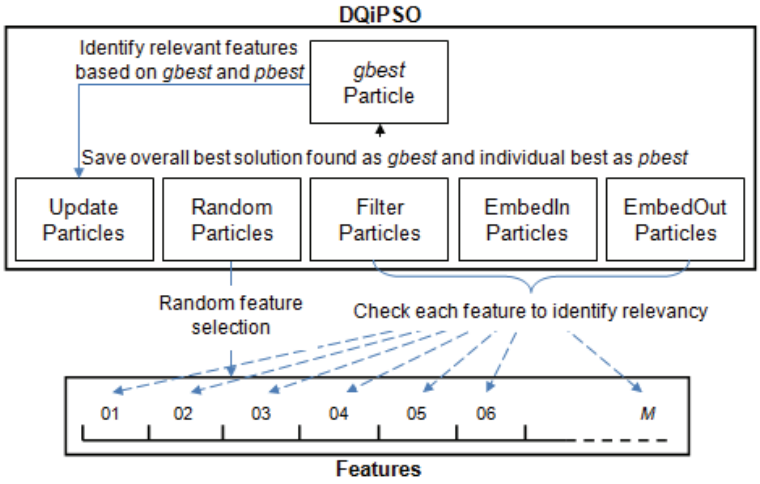


Fig. 2. DQiPSO feature selection strategy

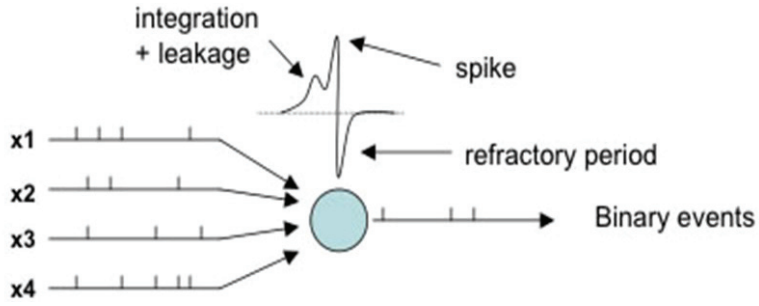
3. Evolving spiking Neural Networks

Many successful Artificial Neural Network (ANN) models have been developed and applied for learning from data and for generalization to new data (Arbib, 2003). Applications include: classification, time series prediction, associative storage and retrieval of information, robot and process control, medical and business decision support, and many others (Arbib, 2003). Most of these ANN use simple and deterministic models of artificial neurons, such as the McCulloch and Pitts model (McCulloch & Pitts, 1943). They also use rate coded information representation, where average activity of a neuron or an ANN is represented as a scalar value. Despite the large structural diversity of existing ANN, the limited functionality of the neurons and connections between them has constrained the scope of applications of ANN and their efficiency when modelling large scale, noisy, dynamic and stochastic processes such as ecological, environmental, physical, biological, cognitive, and others.

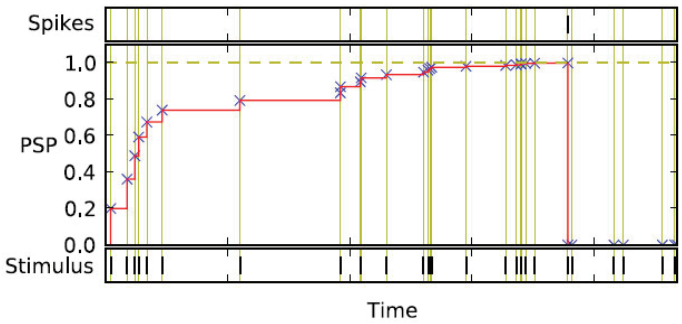
Recently new knowledge about neuronal, genetic and quantum levels of information processing in biological neural networks has been discovered. For example, whether a neuron spikes or not at any given time could depend not only on input signals but also on

gene and protein expression (Kojima & Katsumata, 2009), physical properties of connections (Huguenard, 2000), probabilities of spikes being received at the synapses, emitted neurotransmitters, open ion channels and others. Many of these properties have been mathematically modelled and used to study biological neurons (Gerstner & Kistler, 2002), but have not been properly utilised for more efficient ANN for complex AI problems.

Spiking Neural Networks (SNN) models are made up of artificial neurons that use trains of spikes to represent and process pulse coded information. In biological neural networks, neurons are connected at synapses and electrical signals (spikes) pass information from one neuron to another. SNN are biologically plausible and offer some means for representing time, frequency, phase and other features of the information being processed. A simplified diagram of a spiking neuron model is shown in Figure 3a. Figure 3b shows the mode of operation of a spiking neuron, which emits an output spike when the total spiking input – Post Synaptic Potential (PSP), is larger than a spiking threshold.



(a)



(b)

Fig. 3. (a) A simplified diagram of a spiking neuron model. (b) A spiking neuron emits an output spike when the total spiking input – Post Synaptic Potential (PSP), is larger than a spiking threshold.

Based on the SNN, Evolving SNN (ESNN) was introduced (Wysoski et al., 2006) where SNN evolve their structure through fast one-pass learning from data and have the potential for solving complex problems. ESNN have been applied for classification tasks, such as face recognition, person authentication based on audiovisual information, taste recognition. They achieved better results than previously published models. The ESNN architecture consists of an encoding method for real value data to spike time, neuron model and learning method.

3.1 Information encoding methods

The information in ESNN is represented as spikes; therefore, input information must be encoded in spike pulses. The well-known encoding technique for ESNN is the Population Encoding (Bohte et al., 2002). Population Encoding distributes a single input value to multiple pre-synaptic neurons. Each pre-synaptic neuron generates a spike at firing time.The firing time is calculated using the intersection of Gaussian function. The centre of the Gaussian function is calculated using Equation (6) and the width is computed using Equation (7) with the variable interval of $\left[I_{\min}, I_{\max}\right]$. The parameter β controls the width of each Gaussian receptive field.

$$\mu = I_{\min} + (2 * i - 3) / 2 * (I_{\max} - I_{\min}) / (M - 2) \tag{6}$$

$$\sigma = 1 / \beta (I_{\max} - I_{\min}) / (M - 2) \quad where \quad 1 \le \beta \le 2 \tag{7}$$

The illustration of this encoding process is shown in following figure.

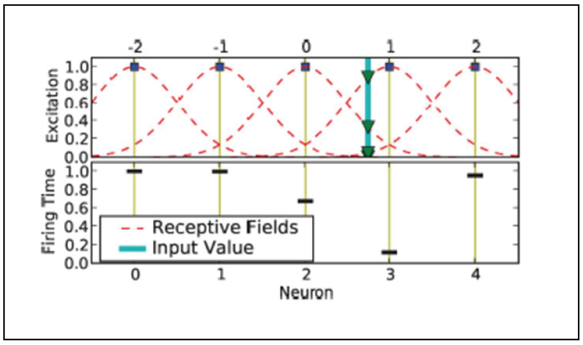


Fig. 4. Population Encoding Method.

3.2 Neuron models

Most of the SNN models have been well explained by Gerstner and Kistler (Gerstner & Kistler, 2002). For the ESNN, Thorpe's neuron model (Thorpe, 1997) has been selected because of its effectiveness and simplicity. The fundamental concept of this model is that the earlier spikes received by a neuron have a stronger weight compared with later spikes. Once the neuron reaches a certain amount of spikes and the Post-Synaptic Potential (PSP) exceeds the threshold value, it fires and becomes disabled. The neuron in this model can only fire once. The computation of the PSP of neuron i is presented in Equation (8).

$$PSP_i = \begin{cases} 0 & if \quad fired \\ \sum w_{ji} * Mod_i^{order(j)} & else \end{cases} \tag{8}$$

where w_{ji} being the weight of pre-synaptic neuron j . Mod_i being a parameter called modulation factor with an interval of [0,1]and $order^{(j)}$ representing the rank of the spike emitted by the neuron. The $order^{(j)}$ starts with 0 if it spikes first amongst all pre-synaptic neurons and increases according to firing time.

3.3 Learning method

Learning in ESNN is a complex process since information is represented in spikes, which is time dependence. Spike Time Dependent Plasticity (STDP) is a form of Hebbian Learning where spike time and transmission are used in order to calculate the change in the synaptic weight of a neuron. If a pre-synaptic spike arrives at the synapse before the postsynaptic action potential, the synapse is potentiated; if the timing is reversed, the synapse is depressed (Markram et al., 1997).

The One-Pass Algorithm is the learning algorithm for ESNN which follows both the SDTP learning rule and the time-to-first spike learning rule (Thorpe,1997). In this algorithm, each training sample creates a new output neuron. The trained threshold values and the weight pattern for that particular sample are stored in the neuron repository. However, if the weight pattern of the trained neuron greatly resembles a neuron in the repository, it will merge into the most similar one. The merging process involves modifying the weight pattern and the threshold of the merged neurons to the average value. Otherwise, it will be added to the repository as a newly trained neuron. The major advantage of this learning algorithm is the ability of the trained network to learn incrementally new samples without retraining.

3.4 ESNN structure

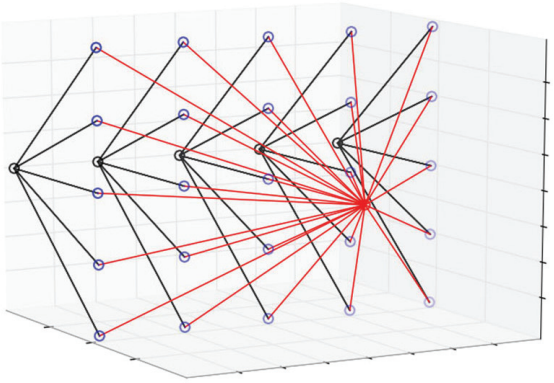


Fig. 5. A simplified ESNN structure

In general, each input neuron in the ESNN (black neuron in Figure 5) is connected to multiple pre-synaptic neurons (blue neurons). This process will transform the input values into a highly dimensional structure where each pre-synaptic neuron generates a certain spike at a firing time. The firing time is calculated using the intersection of the Gaussian function with the input value. Based on the firing time, a weight for each connection to the output neuron (red neuron) is generated. In the training process, the output neuron stores the computed weight of all pre-synaptic neurons, a threshold value to determine when the output neuron will spike and the class label the input sample belongs to. In the testing process, similar to the training process, each testing sample is encoded to spikes by the multiple pre-synaptic neurons. Then, the Post-Synaptic Potential (PSP) of the output class neurons is calculated. Once the neuron reaches a certain amount of spikes and the PSP exceeds the threshold value, it fires an output spike and becomes disabled. The testing

sample belongs to an output class of the output neuron which fires first among all output neurons. A detailed ESNN training algorithm follows.

Algorithm 1 ESNN Training Algorithm

1: Initialize neuron repository $R = \{\ \}$

2: Set ESNN parameters $Mod = [0,1]$, $C = [0,1]$ and $Sim = [0,1]$

3: **for every** input sample i that belongs to the same output class **do**

4: Encode input samples into firing time of pre-synaptic neurons j
Create a new output neuron for this class and calculate the connection weights as follows: $w_j = (Mod)^{order(j)}$

5: Calculate $PSP_{\max(i)} = \sum w_j * Mod^{order(j)}$

6: Get PSP threshold value $\chi_i = PSP_{\max(i)} * C$

7: **if** the new neuron weight vector <= Sim of trained output neuron weight vector in R **then**

8: Merge the weights and the threshold of the new neuron with the most similar neuron in the same class output group

9: $$w = \frac{w_{new} + w * N}{N + 1}$$

10: $$\chi = \frac{\chi_{new} + \chi * N}{N + 1}$$

11: where N is the number of all previous merges of the merged neuron

12: **Else**

13: Add the new neuron to the output neuron repository R

14: **end if**

15: **end for** (Repeat to all input samples for other output classes)

4. Integrated structure for parameter optimization and feature selection

An integrated structure is presented in which the features and parameters are optimized simultaneously, and this leads to better optimization. This experiment further tests the efficiency of DQiPSO in selecting the most relevant features and also optimizing the ESNN parameters. Here the DQiPSO optimizes the ESNN parameters: Modulation Factor (Mod), Proportion Factor (C) and Similarity (Sim), as well as identifies the relevant features. All particles are initialized with random value and subsequently interact with each other based on classification accuracy. Since there are two components to be optimized, each particle is divided into two parts. The first part of each hybrid particle holds the feature mask where information is stored in a string of qubits. Another part holds parameters of ESNN. The proposed integrated framework is shown in Figure 6.

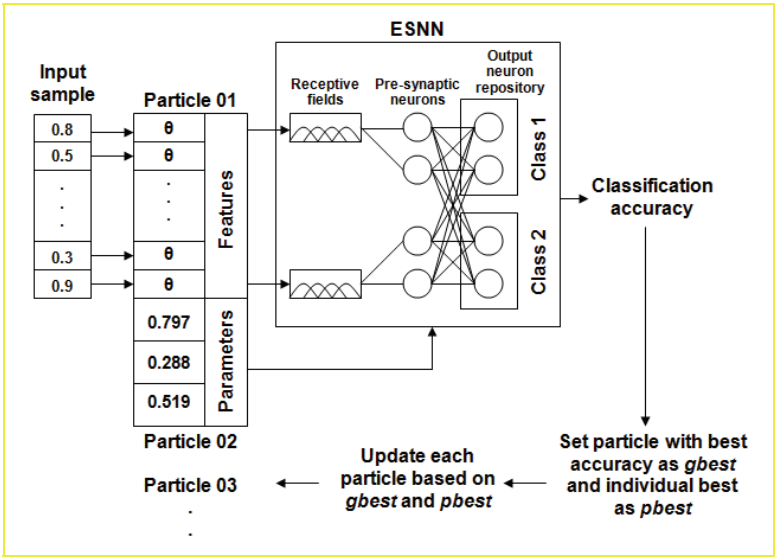


Fig. 6. An integrated ESNN-DQiPSO framework for feature selection and parameter optimization.

4.1 Setup

The proposed integrated ESNN-DQiPSO method was tested on a Uniform Hypercube dataset (Estavest et al., 2009). Thirty features were created where only 10 are the relevant features, where a sample belongs to class 1 when $r_i < \gamma^{i-1} * \alpha$ for i = 1 till 10. Parameters chosen were $\gamma = 0.8$ and $\alpha = 0.5$. The features which are not relevant to determining the output class consist of 10 random features with the random value of $[0,1]$, and 10 redundant features copied from relevant features with an addition of Gaussian noise of 0.3. The features were arranged randomly to simulate the real world problem where relevant features are scattered in the dataset as follows:

Features	Arrangement
Relevant	02, 04, 09, 10, 11, 15, 19, 20, 26, 30
Redundant	03, 07, 12, 14, 17, 18, 21, 25, 27, 28
Random	01, 05, 06, 08, 13, 16, 22, 23, 24, 29

Table 1. Feature arrangement

The problem consists of 500 samples, equally distributed into two classes. It was applied to the proposed framework and compared with the QiPSO method and ESNN with standard PSO. However, because standard PSO is inadequate for feature selection, it only optimizes the ESNN parameters. Based on the preliminary experiment, 20 ESNN's pre-synaptic neurons were chosen. For the DQiPSO, 18 particles were used, consisting of six Update, three Filter, three Random, three Embed In and three Embed Out. For the QiPSO, 20

particles were used. C_1 and C_2 were set to 0.05 to balance the exploration between *gbest* and *pbest* with the inertia weight w = 2.0. Ten-fold cross validations were used and the average result was computed in 500 iterations.

4.2 Results

Figure 7 illustrates the comparison of selected features from DQiPSO and QiPSO during the learning process. The lighter colour means more frequent corresponding features are

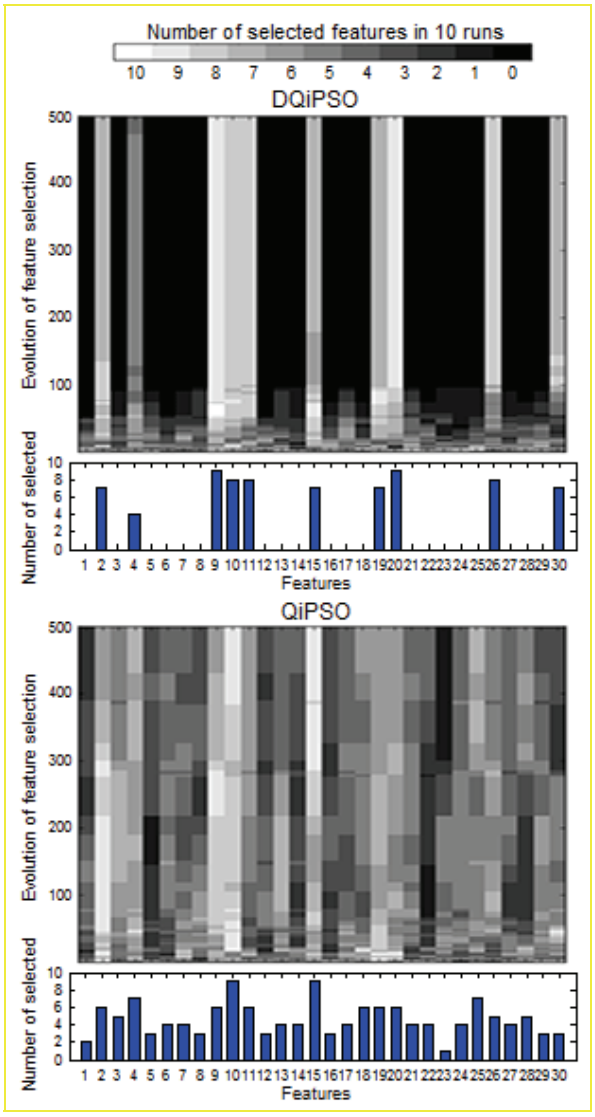


Fig. 7. Evolution of feature selection

selected and darker means otherwise. All the features have been ranked based on the number of selected features from 10 runs to determine their relevancy. From the figure, 10 relevant features which contained the most information can be clearly identified and are constantly being selected by DQiPSO. In contrast, the redundant and random features are completely rejected during the optimization process. DQiPSO takes less than 100 iterations to identify the relevant and irrelevant features. Based on the feature ranking, the most relevant features found are: Feature 9 and Feature 20, followed by Feature 10, Feature 11, Feature 26, Feature 2, Feature 15, Feature 19, Feature 30 and Feature 4. In contrast, the ability of the QiPSO to reject the irrelevant features is unsatisfactory. Most of the irrelevant features are still being selected, which contributes to the low classification accuracy and increased computation time. The most relevant features found by QiPSO are Feature 10 and 15, followed by Feature 4, Feature 25, Feature 2, Feature 9, Feature 11, Feature 18, Feature 19 and Feature 20. Other features are occasionally selected and can be considered as irrelevant features by QiPSO. Some relevant features are also being regarded as irrelevant due to the number of selected is low, while some irrelevant features which contain no information are considered as relevant by QiPSO. This situation has affected the results and overall classification performance of the ESNN-QiPSO.

Figure 8 shows the results of parameter optimization. All parameters evolve steadily towards a certain optimal value, where the correct combination together with the selected relevant features leads to a better classification accuracy. In terms of the classification result, the average accuracy for ESNN-DQiPSO is 99.25% with the result of every single run consistently above 98%. For the ESNN-QiPSO algorithm, the average accuracy is 96.57%. The proposed DQiPSO and QiPSO methods are able to select relevant features with few or occasionally no irrelevant features, while simultaneously providing nearly optimal parameter combinations in the early stage of learning. This situation leads to acceptably

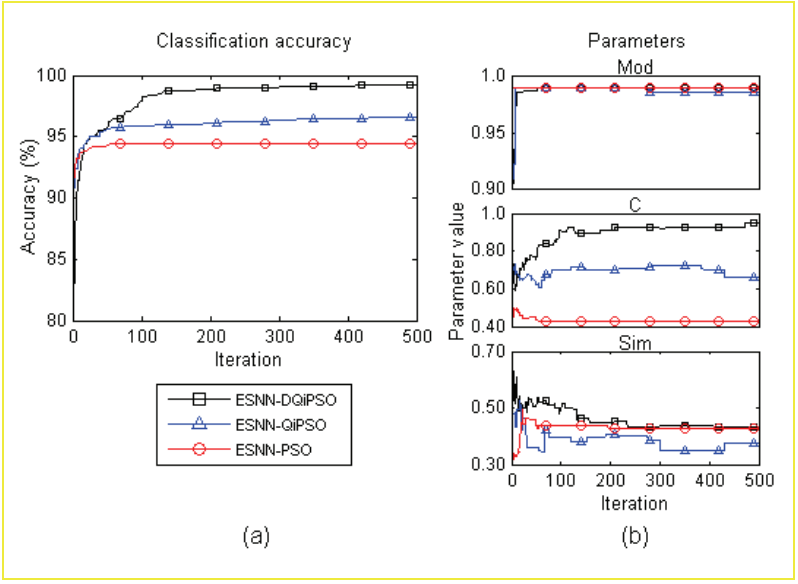


Fig. 8. a) Classification accuracy and b) Parameter optimization result

high average accuracy at the beginning of the learning process. For the ESNN-PSO algorithm, although the classification accuracy is 94.43%, this algorithm is entirely dependent on the parameter optimization which has affected the results, giving the lowest accuracy. The testing results for ESNN-DQiPSO, ESNN-QiPSO and ESNN-PSO are 95.99%, 91.58% and 83.93% respectively.

5. Conclusion and future research

This chapter has introduced a new PSO model and has shown how this optimizer can be implemented for parameter optimization and feature selection. Since feature selection is a unique task involving probability, quantum computation has been embedded into PSO and has been applied to an ESNN for classification. The new method results in a more efficient classification ESNN model with optimal features selected and parameters optimised.
Future work is planned to improve the proposed optimization method and to apply it to the Probabilistic Spiking Neural Networks (PSNN) (Kasabov, 2010). In this PSNN, not only features will be represented by a quantum bit vector, but also all connections between the neurons. A neuronal connection is either existent (1) or nonexistent (0), or in another interpretation – either propagating a spike or not propagating it. A quantum bit vector would be a suitable representation of all connections that can be optimized using the modified PSO. Each particle will be divided into three parts; the first two parts use quantum probability computation for feature and connection selection and the last part holds the real value for the parameters. It is to be believed that the proposed method will be able to optimize the given problem in a far more efficient way.

6. References

Arbib, M. (2003). *The Handbook of Brain Theory and Neural Networks*, MIT Press, ISBN 978-0-262-01148-8, Cambridge, MA, USA.

Bäck, T. & Schwefel, H. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, Vol. 1, No. 1, (March 1993), pp. 1-23, ISSN 1063-6560.

Blum, L.A. & Langley, P. (1997). Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, Vol. 97, No. 1-2, (December 1997), pp. 245–271, ISSN 0004-3702.

Bohte, S.M.; Kok, J.N. & Poutre, H.L. (2002). Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons. *Neurocomputing*, Vol. 48, No. 1-4, (October 2002), pp. 17-37, ISSN 0925-2312.

da Cruz, A.V.A.; Vellasco, M.M.B. & Pacheco, M.A.C. (2006). Quantum Inspired Evolutionary Algorithm for Numerical Optimisation, *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2630-2637, ISBN 0-7803-9487-9, Vancouver, Canada, July 16-21, 2006.

Eberhart, R. & Kennedy, J. (1995). A New Optimizer Using Particle Swarm Theory, *Proceedings of Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43, ISBN 0-7803-2676-8, Nagoya, Japan, October 4-6, 1995.

Estavest, P.; Tesmer, M.; Perez, C.; & Zurada, J. (2009). Normalized Mutual Information Feature Selection. *IEEE Transactions on Neural Networks*, Vol. 20, No. 2, (February 2009), pp. 189-201, ISSN 1045-9227.

Gerstner, W. & Kistler. W. (2002). *Spiking Neuron Models: An Introduction*, Cambridge University Press, ISBN 0521890799, New York, NY, USA.

Grover, L. K. (1996). A Fast Quantum Mechanical Algorithm for Database Search, *Proceedings of 28th annual ACM symposium on Theory of computing*, pp. 212–219, ISBN 0-89791-785-5, Philadelphia, USA, May 22-24, 1996.

Han, K.H. & Kim, J.H. (2000). Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem, *Proceedings of 2000 Congress on Evolutionary Computation*, pp. 1354-1360, ISBN 0-7803-6375-2, La Jolla, CA , USA, July 16-19, 2000.

Huguenard, J.R. (2000). Reliability of Axonal Propagation: The Spike Doesn't Stop Here. *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 97, No. 17, (August 2000), pp. 9349-9350, ISSN 1091-6490.

Jang, J.S.; Han, K.H. & Kim, J.H. (2004). Face Detection Using Quantum-Inspired Evolutionary Algorithm, *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2100-2106, ISBN 0-7803-8515-2, Portland, OR, USA, June 19-23, 2004.

Kasabov, N. (2007). *Evolving Connectionist Systems* (2nd ed.), Springer-Verlag, ISBN 978-1-84628-345-1, Secaucus, NJ, USA.

Kasabov, N. (2010). To Spike or Not To Spike: A Probabilistic Spiking Neural Model. *Neural Networks*, Vol. 23, No. 1, (January 2010), pp. 16-19, ISSN 0893-6080.

Kojima, H. & Katsumata, S. (2008). Analysis of Synaptic Transmission and Its Plasticity by Glutamate Receptor Channel Kinetics Models and 2-Photon Laser Photolysis, *Proceedings of International Conference on Neural Information Processing*, pp. 88-94, ISBN 3-642-02489-0, Auckland, New Zealand, November 25-28, 2008.

Markram, H.; Lubke, J.; Frotscher, M. & Sakmann, B. (1997). Regulation of Synaptic Efficacy by Coincidence of Postsynaptic Aps and Epsps. *Science*, Vol. 275, No. 5297, (January 1997), pp. 213-215, ISSN 0193-4511.

McCulloch, W.S. & Pitts, W. A. (1943). Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology*, Vol. 5, No. 4. (December 1943), pp. 115-133, ISSN 00074985.

Narayanan, A. (1999). Quantum Computing for Beginners, *Proceedings of Congress on Evolutionary Computation*, pp. 2231-2238, ISBN 0-7803-5536-9, Washington, DC , USA, July 6-9, 1999.

Shi, Y. & Eberhart, R. (1998). A Modified Particle Swarm Optimizer, *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 69–73, ISBN 0-7803-4869-9, Anchorage, USA, May 4-9, 1998.

Shor, P.W. (1994). Algorithms for Quantum Computation: Discrete Log and Factoring, *Proceedings of IEEE Symposium on Foundations of Computer Science*, pp. 124-134, ISBN 0-8186-6580-7, Santa Fe, NM, USA, November 20-22, 1994.

Sun, J.; Feng, B. & Xu, W. (2004). Particle Swarm Optimization with Particles Having Quantum Behavior, *Proceedings of Congress on Evolutionary Computation*, pp. 325-331, ISBN 0-7803-8515-2, Portland, Oregon, USA, June 20-23, 2004.

Talbi, H.; Draa, A. & Batouche, M. (2006). A Novel Quantum Inspired Evaluation Algorithm for Multisource Affine Image Registration. *The International Arab Journal of Information Technology*, Vol. 3, No. 1, (January 2006), pp. 9-15, ISSN 1683-3198.

Thorpe, S.J. (1997). How Can the Human Visual System Process a Natural Scene in Under 150ms? Experiments and Neural Network Models, In: *Proceedings of European Symposium on Artificial Neural Networks*, Verleysen, M. (ed.), D-Facto public, ISBN 2-9600049-7-3, Bruges, Belgium.

Wysoski, S. G.; Benuskova, L. & Kasabov, N. (2006). On-Line Learning with Structural Adaptation in a Network of Spiking Neurons for Visual Pattern Recognition, In: *Proceedings of 2006 International Conference on Artificial Neural Networks, Lecture Notes in Computer Science, Vol. 4131*, Kollias, S. et al. (Eds), pp. 61-70, Springer-Verlag, ISBN 978-3-540-38625-4, Berlin Heidelberg.

9

Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures

Ivan Zelinka[1], Donald Davendra[2], Roman Senkerik[3], Roman Jasek[4] and Zuzana Oplatkova[5]

[1,2,3,4,5]*Tomas Bata University in Zlin, Faculty of Applied Informatics, Nad Stranemi 4511, Zlin, 76001*

[1]*Department of Computer Science, Faculty of Electrical Engineering and Computer Science VSB-TUO, 17. listopadu 15, 708 33 Ostrava-Poruba Czech Republic*

1. Introduction

This chapter discusses an alternative approach for symbolic structures and solutions synthesis and demonstrates a comparison with other methods, for example Genetic Programming (GP) or Grammatical Evolution (GE). Generally, there are two well known methods, which can be used for symbolic structures synthesis by means of computers. The first one is called GP and the other is GE. Another interesting research was carried out by Artificial Immune Systems (AIS) or/and systems, which do not use tree structures like linear GP and other similar algorithm like Multi Expression Programming (MEP), etc. In this chapter, a different method called Analytic Programming (AP), is presented. AP is a grammar free algorithmic superstructure, which can be used by any programming language and also by any arbitrary Evolutionary Algorithm (EA) or another class of numerical optimization method. This chapter describes not only theoretical principles of AP, but also its comparative study with selected well known case examples from GP as well as applications on synthesis of: controller, systems of deterministic chaos, electronics circuits, etc. For simulation purposes, AP has been co-joined with EA's like Differential Evolution (DE), Self-Organising Migrating Algorithm (SOMA), Genetic Algorithms (GA) and Simulated Annealing (SA). All case studies has been carefully prepared and repeated in order to get valid statistical data for proper conclusions.

The term *symbolic regression* represents a process during which measured data sets are fitted, thereby a corresponding mathematical formula is obtained in an analytical way. An output of the symbolic expression could be, for example, $\sqrt[N]{x^2 + \frac{y^3}{k}}$, and the like. For a long time, symbolic regression was a domain of human calculations but in the last few decades it involves computers for symbolic computation as well.

The initial idea of symbolic regression by means of a computer program was proposed in GP (Koza, 1990; 1998). The other approach of GE was developed in Ryan et al. (1998) and AP in Zelinka et al. (2005a). Another interesting investigation using symbolic regression were carried out in Johnson (2004) on AIS and Probabilistic Incremental Program Evolution (PIPE), which generates functional programs from an adaptive probability distribution over

all possible programs. Yet another new technique is the so called *Transplant Evolution*, see Weisser & Osmera (2010b), Weisser & Osmera (2010a) and Weisser et al. (2010) which is closely associated with the conceptual paradigm of AP, and modified for GE. GE was also extended to include DE by O'Neill & Brabazon (2006). Symbolic regression is schematically depicted in Figure 1. Generally speaking, it is a process which combines, evaluates and creates more complex structures based on some elementary and noncomplex objects, in an evolutionary way. Such elementary objects are usually simple mathematical operators $(+, -, \times, ...)$, simple functions (*sin*, *cos*, *And*, *Not*, ...), user-defined functions (simple commands for robots – MoveLeft, TurnRight, ...), etc. An output of symbolic regression is a more complex "object" (formula, function, command,...), solving a given problem like data fitting of the so-called Sextic and Quintic problem described by Equation (1) (Koza et al., 1999; Zelinka & Oplatkova, 2003a), randomly synthesized function by Equation (2) (Zelinka & Oplatkova, 2003a), Boolean problems of parity and symmetry solution (basically logical circuits synthesis) by Equation (3) (Koza et al., 2003; Zelinka et al., 2005a), or synthesis of quite complex robot control command by Equation (4) (Koza, 1998; Oplatkova & Zelinka, 2006). Equations (1)–(4) mentioned here are just a few samples from numerous repeated experiments done by AP, which are used to demonstrate how complex structures can be produced by symbolic regression in general for different problems.

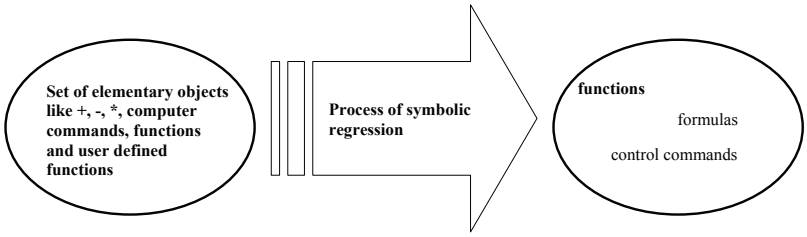


Fig. 1. Symbolic regression - schematic view

$$x\left(K_1 + \frac{\left(x^2 K_3\right)}{K_4\left(K_5 + K_6\right)}\right) * \left(-1 + K_2 + 2x\left(-x - K_7\right)\right) \tag{1}$$

$$\sqrt{t}\left(\frac{1}{\log\left(t\right)}\right)^{\sec^{-1}(1.28)} \log^{\sec^{-1}(1.28)}\left(\sinh\left(\sec\left(\cos\left(1\right)\right)\right)\right) \tag{2}$$

$$\begin{array}{l} Nor[(Nand[Nand[B||B, B\&\&A], B])\&\&C\&\&A\&\&B, \\ Nor[(!C\&\&B\&\&A||!A\&\&C\&\&B||!C\&\&!B\&\&!A)\&\& \\ (!C\&\&B\&\&A||!A\&\&C\&\&B||!C\&\&!B\&\&!A)|| \\ A\&\&(!C\&\&B\&\&A||!A\&\&C\&\&B||!C\&\&!B\&\&!A), \\ (C||!C\&\&B\&\&A||!A\&\&C\&\&B||!C\&\&!B\&\&!A)\&\&A]] \end{array} \tag{3}$$

$$\begin{array}{l} \text{Prog2[Prog3[Move, Right, IfFoodAhead[Left, Right]],} \\ \text{IfFoodAhead[IfFoodAhead[Left, Right], Prog2[IfFoodAhead[} \\ \text{IfFoodAhead[IfFoodAhead[Left, Right], Right], Right],} \\ \text{IfFoodAhead[Prog2[Move, Move], Right]]]]} \end{array} \tag{4}$$

1.1 Genetic programming

GP was the first tool for symbolic regression carried out by means of computers instead of humans. The main idea comes from GA, which was used in GP (Koza, 1990; 1998). Its ability to solve very difficult problems is well proven; for example, GP performs so well that it can be applied to synthesize highly sophisticated electronic circuits (Koza et al., 2003).

The main principle of GP is based on GA, which is working with populations of individuals represented in the LISP programming language. Individuals in a canonical form of GP are not binary strings, different from GA, but consist of LISP symbolic objects (commands, functions, ...), etc. These objects come from LISP, or they are simply user-defined functions. Symbolic objects are usually divided into two classes: functions and terminals. Functions were previously explained and terminals represent a set of independent variables like x, y, and constants like π , 3.56, etc.

The main principle of GP is usually demonstrated by means of the so-called trees (basically graphs with nodes and edges, as shown in Figure 2 and Figure 3, representing individuals in LISP symbolic syntax). Individuals in the shape of a tree, or formula like $0.234Z + X - 0.789$, are called programs. Because GP is based on GA, evolutionary steps (mutation, crossover, ...) in GP are in principle the same as GA. As an example, GP can serve two artificial parents – trees on Figure 2 and Figure 3, representing programs $0.234Z + X - 0.789$ and $ZY(Y + 0.314Z)$. When crossover is applied, for example, subsets of trees are exchanged. Resulting offsprings of this example are shown on Figure 3.

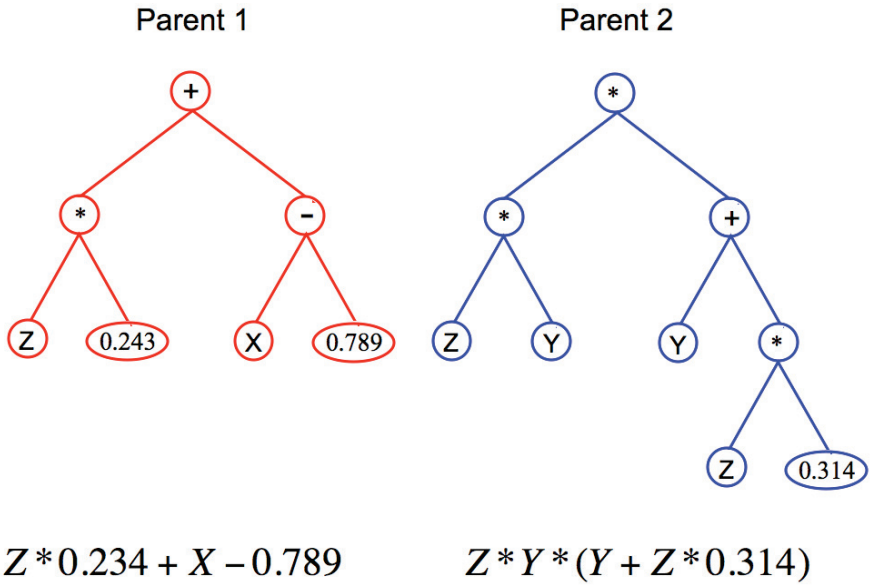


Fig. 2. Parental trees

Subsequently, the offspring fitness is calculated, such that the behavior of the just-synthesized and evaluated individual-tree should be as similar as possible to the desired behavior. The desired behavior can be regarded as a measured data set from some process (a program that

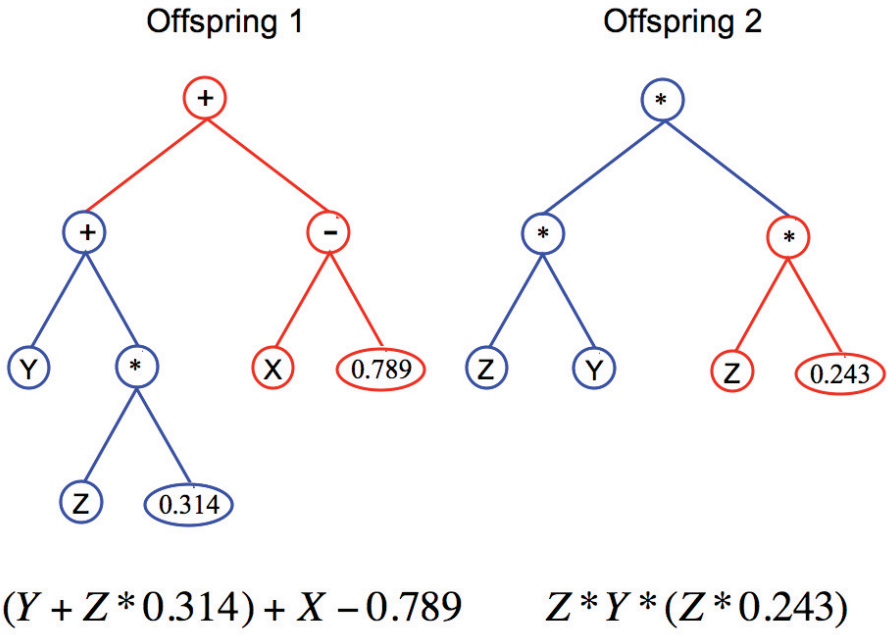


Fig. 3. Offsprings

should fit them as well as possible) or like an optimal robot trajectory, i.e., when the program is evaluating a sequence of robot commands (TurnLeft, Stop, MoveForward,...) leading as close as possible to the final position. This is basically the same for GE.
For detailed description of GP, see Koza (1998), Koza et al. (1999).

1.2 Grammatical evolution

GE is another program developed in O'Neill & Ryan (2003), which performs a similar task to that of GP. GE has one advantage over GP, and this is the ability to use any arbitrary programming language, not only LISP as in the case of the canonical version of GP. In contrast to other EA's, GE was used only with a few search strategies, and with a binary representation of the populations (O'Neill & Ryan, 2003). The last successful experiment with DE applied on GE was reported in O'Neill & Brabazon (2006). GE in its canonical form is based on GA, thanks to a few important changes it has in comparison with GP. The main difference is in the individual coding.
While GP manipulates in LISP symbolic expressions, GE uses individuals based on binary strings. These are transformed into integer sequences and then mapped into a final program in the Backus-Naur Form (BNF) (O'Neill & Ryan, 2003), as explained by the following artificial example. Let $T = \{+, -, \times, /, x, y\}$ be a set of operators and terminals and let F = {epr, op, var} be the so-called nonterminals. In this case, the grammar used for final program synthesis is given in Table 1. The rule used for individuals transforming into a program is based on Equation (5) below. GE is based on binary chromosome with a variable length, divided into the so-called codons (range of integer values, 0-255), which is then transformed into an integer domain according to Table 2.

Nonterminals		Unfolding	Index
expr	::=	op expr expr	0
		var	1
op	::=	+	0′
		-	1′
		*	2′
		/	3′
var	::	X	0″
		Y	(1″)

Table 1. Grammatical evolution - rules

Chromozone	Binary	Integer	BNF index
Codon 1	101000	40	0
Codon 2	11000011	162	2′
Codon 3	1100	67	1
Codon 4	10100010	12	0″
Codon 5	1111101	125	1
Codon 6	11100111	231	1″
Codon 7	10010010	146	Unused
Codon 8	10001011	139	Unused

Table 2. Grammatical evolution - codon

$$\begin{aligned} &\text{unfolding} = \text{codon mod rules} \\ &\text{where rules is number of rules for given nonterminal} \end{aligned} \tag{5}$$

Synthesis of an actual program can be described by the following. Start with a nonterminal object expr. Because the integer value of Codon 1 (see Table 2) is 40, according to Equation (5), one has an unfolding of *expr = op expr expr* (40 mod 2, 2 rules for *expr*, i.e., 0 and 1). Consequently, Codon 2 is used for the unfolding of *op* by * (162 mod 4), which is the terminal and thus the unfolding for this part of program is closed. Then, it continues in unfolding of the remaining nonterminals (*expr expr*) till the final program is fully closed by terminals. If the program is closed before the end of the chromosome is reached, then the remaining codons are ignored; otherwise, it continues again from the beginning of the chromosome. The final program based on the just-described example is in this case $x \cdot y$ (see Figure 4). For a fully detailed description of GE principles, see O'Neill & Ryan (2003) or consult [http://www.grammaticalevolution.org/].

1.3 Analytic programming

The final method described here and used for experiments in this chapter is called AP, which has been compared to GP with very good results (see, for example, Zelinka & Oplatkova (2003a), Oplatkova (2005), Zelinka et al. (2005a), Oplatkova & Zelinka (2006), Zelinka et al. (2008)) or visit website *www.ivanzelinka.eu*.

The basic principles of AP were developed in 2001 and first published in Zelinka (2001) and Zelinka (2002a). AP is also based on the set of functions, operators and terminals, which are usually constants or independent variables alike, for example:

- **functions**: *sin*, *tan*, *tanh*, *And*, *Or*,...

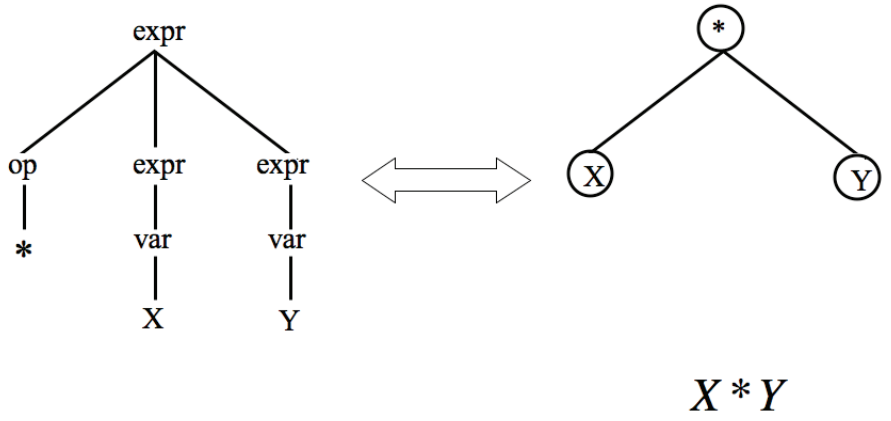


Fig. 4. Final program by GE

- **operators**: +, -, ×, /, dt,...
- **terminals**: 2.73, 3.14, t,...

All these objects create a set, from which AP tries to synthesize an appropriate solution. Because of the variability of the content of this set, it is called a general functional set (GFS). The structure of GFS is nested, i.e., it is created by subsets of functions according to the number of their arguments (Figure 5). The content of GFS is dependent only on the user. Various functions and terminals can be mixed together. For example, GFS_{all} is a set of all functions, operators and terminals, GFS_{3arg} is a subset containing functions with maximally three arguments, GFS_{0arg} represents only terminals, etc. (Figure 5).
AP, as further described later, is a mapping from a set of individuals into a set of possible programs. Individuals in population and used by AP consist of non-numerical expressions (operators, functions,...), as described above, which are in the evolutionary process represented by their integer position indexes (Figure 6, Figure 7, see also Chapter 2). This index then serves as a pointer into the set of expressions and AP uses it to synthesize the resulting function-program for cost function evaluation.
Figure 7 demonstrates an artificial example as to how a final function is created from an integer individual via Discrete Set Handling (DSH). Number 1 in the position of the first parameter means that the operator + from GFS_{all} is used (the end of the individual is far enough). Because the operator + must have at least two arguments, the next two index pointers 6 (*sin* from GFS) and 7 (*cos* from GFS) are dedicated to this operator as its arguments. The two functions, *sin* and *cos*, are one-argument functions, so the next unused pointers 8 (*tan* from GFS) and 9 (*t* from GFS) are dedicated to the *sin* and *cos* functions. As an argument of *cos*, the variable *t* is used, so this part of the resulting function is closed (*t* is zero-argument) in its AP development. The one-argument function *tan* remains, and because there is one unused pointer 9, *tan* is mapped on *t* which is on the 9th position in GFS.
To avoid synthesis of pathological functions, a few security *tricks* are used in AP. The first one is that GFS consists of subsets containing functions with the same or a smaller number of arguments. The nested structure (see also Figure 5) is used in the special security subroutine, which measures how far the end of an individual is and, according to this, mathematical

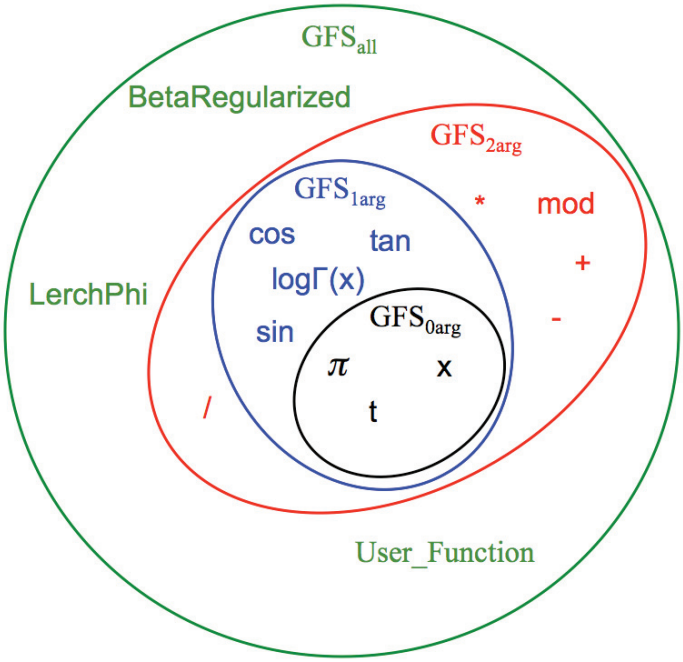


Fig. 5. Hierarchy in GFS

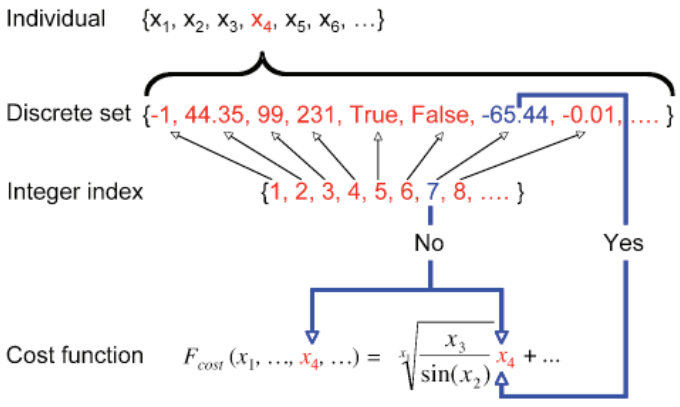


Fig. 6. DSH-Integer index

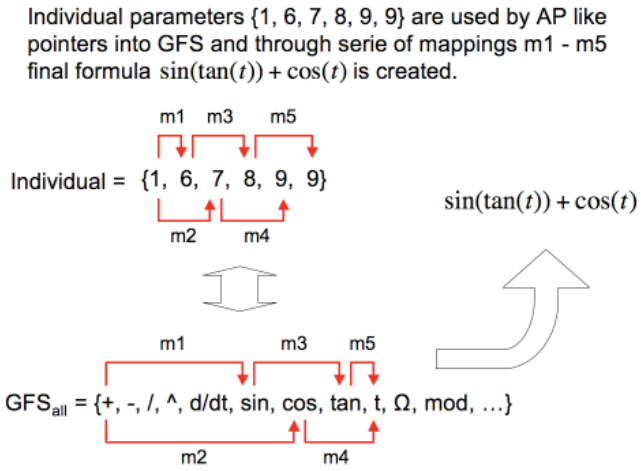


Fig. 7. Principle of mapping from GFS to programs

elements from different subsets are selected to avoid pathological functions synthesis. More precisely, if more arguments are desired then a possible function (the end of the individual is near) will be replaced by another function with the same index pointer from the subset with a smaller number of arguments. For example, it may happen that the last argument for one function will not be a terminal (zero-argument function). If the pointer is longer than the length of subset, e.g., a pointer is 5 and is used GFS_0, then the element is selected according to the rule: element = pointer_value mod number_of_elements_in_GFS_0. In this example, the selected element would be the variable t (see GFS_0 in Figure 5).
GFS need not be constructed only from clear mathematical functions as demonstrated above, but may also be constructed from other user-defined functions, e.g., logical functions, functions which represent elements of electrical circuits or robot movement commands, linguistic terms, etc.

1.3.1 Versions

AP was evaluated in three versions. All three versions utilize the same set of functions for program synthesis, terminals, etc., as in GP (Koza, 1998; Koza et al., 1999)). The second version labelled as AP_{meta} (the first version, AP_{basic}) is modified in the sense of constant estimation. For example, the so-called sextic problem was used in Koza (1998) to randomly generate constants, whereas AP uses only one, called K, which is inserted into the formula (6) below at various places by the evolutionary process. When a program is synthesized, all K's are indexed as K_1, K_2, ..., K_n to obtain (7) the formula, and then all K_n are estimated by using a second EA, the result of which can be, for example, (8). Because EA (slave) "works under" EA (master), i.e., $EA_{master} \rightarrow$ program $\rightarrow$ K indexing $\rightarrow$ $EA_{slave} \rightarrow$ estimation of K_n, this version is called AP with metaevolution, denoted as AP_{meta}.

$$\frac{x^2 + K}{\pi^K} \tag{6}$$

$$\frac{x^2 + K_1}{\pi^{K_2}} \tag{7}$$

$$\frac{x^2 + 3.56}{\pi^{-229}} \tag{8}$$

Due to this version being quite time-consuming, AP_{meta} was further modified to the third version, which differs from the second one in the estimation of K. This is accomplished by using a suitable method for nonlinear fitting (denoted AP_{nf}). This method has shown the most promising performance when unknown constants are present. Results of some comparative simulations can be found in Zelinka & Oplatkova (2003b), Zelinka et al. (2005b) and Oplatkova et al. (2008). AP_{nf} was the method chosen for the simulations described in this chapter.

1.3.2 Data set structure and mapping method

The subset structure presence in GFS is vitally important for AP. It is used to avoid synthesis of pathological programs, i.e. programs containing functions without arguments, etc. Performance of AP is, of course, improved if functions of GFS are expertly chosen based on experiences with solved problem.

An important part of AP is a sequence of mathematical operations which are used for program synthesis. These operations are used to transform an individual of a population into a suitable program. Mathematically stated, it is mapping from an individual domain into a program domain. This mapping consists of two main parts. The first part is DHS and the second one are security procedures which do not allow synthesizing of pathological programs. DHS proposed in Lampinen & Zelinka (1999), Zelinka (2004) is used to create an integer index, which is used in the evolutionary process like an alternate individual handled in EA by the method of integer handling. The method of DSH, when used, allows to handle arbitrary objects including nonnumerical objects like linguistic terms hot, cold, dark, ..., logic terms (True, False) or other user defined functions. In the AP, DSH is used to map an individual into GFS and together with Security Procedures (SP) creates the above mentioned mapping, which transforms the arbitrary individual into a program. Individuals in the population consist of integer parameters, i.e. an individual is an integer index pointing into GFS.

AP is basically a series of function mapping. In Figure 7, an artificial example is given as to how a final function is created from an integer individual. Number 1 in the position of the first parameter means that the operator + from GFS_{all} is used (the end of the individual is far enough). Because the operator + has to have at least two arguments, the next two index pointers 6 (*sin* from GFS) and 7 (*cos* from GFS) are dedicated to this operator as its arguments. Both functions, sin and cos, are one-argument functions so the next unused pointers 8 (*tan* from GFS) and 9 (t from GFS) are assigned to *sin* and *cos* function. Because *cos* has used variable t as an argument, this part of resulting function is closed (t is zero-argument) in its AP development. Only one-argument function remains and since there is one unused pointer 9 *tan*, it is mapped on t which is on the 9th position in GFS.

To avoid synthesis of pathological functions, a few security tricks are used in AP. The first one is that GFS consists subsets containing functions with the same number of arguments. Existence of this nested structure is used in the special security subroutine, which is measuring how far the end of individual is and according to this objects from different subsets are selected to avoid pathological function synthesis. Precisely, if more arguments are desired than the possible (the end of the individual is near) function will be replaced by another

function with the same index pointer from subset with lower number of arguments. For example, it may happen if the last argument for one argument function will not be a terminal (zero-argument function) as demonstrated in Figure 7.
GFS doesn't need to be constructed from only clear mathematical functions as is demonstrated, but also from other user-defined functions, which can be used, e.g. logical functions, functions which represent elements of electrical circuits or robot movement commands.

1.3.3 Crossover, mutations and other evolutionary operations

During evolution of a population, a number of different operators are used, such as crossover and mutation. In comparison with GP or GE, evolutionary operators like mutation, crossover, tournament, selection are fully in the competence of used EA. AP does not contain them in any point of view of its internal structure. AP is created like a superstructure of EAs for symbolic regression independent on their algorithmical structure. Operations used in EA's are not influenced by AP and vice versa. For example if DE is used for symbolic regression in AP then all evolutionary operations are done according to the DE rules. AP just transforms individuals into formulas.

1.3.4 Reinforced evolution

During evolution, more or less appropriate individuals are synthesized. Some of these individuals are used to reinforce the evolution towards a better solution synthesis. The main idea of reinforcement is based on the addition of the just-synthesized and partly successful program into an initial set of terminals. Reinforcement is based on a user-defined criterion used in decision as to which individual will be used as an addition into the initial set of terminals. A criterion for the decision is in fact a threshold, i.e., by a user-defined cost value, under which conditions are synthesized solutions being added into GFS.
For example, if the threshold is set to 5, and if the fitness of all individuals (programs in the population) is bigger than 5, then evolution is running on the basic, i.e., initially defined, GFS. When the best individual in the actual population is less than 5, then it is entirely added into the initial GFS and is marked as a terminal. Since this moment, evolution is running on the enriched GFS containing a partially successful program. Thanks to this advantage, evolution is able to synthesize the final solutions much faster than the AP without reinforcement. This fact has been repeatedly verified by simulations on different problems. When the program is added into GFS, the threshold is also set to its fitness. If furthermore, an individual with better fitness than the just-reset threshold is synthesized, then the old one is rewritten by the better one, and the threshold is rewritten by a new fitness value.
It is quite similar to Automatically Defined Functions (ADF) for GP; however, the set of functions and terminals in GP can contain more than one ADF, (which of course at least theoretically increases the complexity of the search space to the order of $n!$), including properly defined arguments of these ADF and critical situation checking (selfcalling,...). This is not a problem of AP reinforcement, because adding a program into the initial GFS is regarded as a terminal (or a terminal structure), i.e., no function, no arguments, no selfcalling, etc., and the cardinality of the initial GFS set increases only by one.

```
Start of simulation No. 5    Time: 22:7 :54.351481

Appended suboptimal solution with CV = 4    Time: 22:7 :54.547317

Appended suboptimal solution with CV = 3    Time: 22:7 :54.577788

Appended suboptimal solution with CV = 2    Time: 22:8 :41.529214

Appended suboptimal solution with CV = 1    Time: 22:10 :14.346728

Number of cost function evaluations: 36270
Cost value: 0
The best individual:
 {8, 5, 8, 3, 1, 9, 4, 2, 3, 9, 7, 4, 4, 10, 1, 10, 5, 8, 5, 7, 10, 6, 5, 10, 9, 9, 7, 2, 6, 3}
Solution: (((B⋁(((A⋀C⋀¬B)⋁(B⋀C⋀¬A)⋁(¬A⋀¬B⋀¬C))⋀C))⩔(B⩔(C⋀A)))⋀
    A⋀((A⩒(((A⋀C⋀¬B)⋁(B⋀C⋀¬A)⋁(¬A⋀¬B⋀¬C)))⩔
      ((A⋀C⋀¬B)⋁(B⋀C⋀¬A)⋁(¬A⋀¬B⋀¬C))))⩒
  (A⩒((A⩒((A⋀C⋀¬B)⋁(B⋀C⋀¬A)⋁(¬A⋀¬B⋀¬C)))⩒
      (((A⋀C⋀¬B)⋁(B⋀C⋀¬A)⋁(¬A⋀¬B⋀¬C))⩒((A⋀C⋀¬B)⋁(B⋀C⋀¬A)⋁(¬A⋀¬B⋀¬C)))))

    {2005, 11, 28, 22, 14, 45.450393}
```

Fig. 8. Effect of reinforced evolution

1.3.5 Security procedures

Security Procedures (SP) are in AP as well as in GP, used to avoid various critical situations. In the case of AP, security procedures were not developed for AP purposes after all, but they are mostly an integrated part of AP. However, sometimes they have to be defined as a part of cost function, based on specific situations (for example situation 2, 3 and 4, shown below). Critical situations are like:

1. pathological function (without arguments, self-looped...)
2. functions with imaginary or real part (if not expected))
3. infinity in functions (dividing by 0, ...)
4. *frozen* functions (an extremely long time to get a cost value - hrs...)
5. etc.

Put simply, an SP can be regarded as a mapping from an integer individual to the program, which is checked as to how far the end of the individual is and based on this information, a sequence of mapping is redirected into a subset with a lower number of arguments. This satisfies the constraint that no pathological function will be generated. Other activities of SP are integrated as part of the cost function to satisfy items 2-4, etc.

1.3.6 Similarities and differences

Similarities and Differences Because AP was partly inspired by GP, then between AP, GP and GE some differences as well as some similarities logically exist. A few of these are:

1. Synthesized programs (similarity): AP as well as GP and GE is able to do symbolic regression in a general point of view. It means that the output of AP is according to simulations (Zelinka, 2002b;c; Zelinka & Oplatkova, 2004; Zelinka et al., 2004), similar to programs from GP and GE.
2. Functional set (similarity): AP_{basic} operates in principle on the same set of terminals and functions as GP or GE, while AP_{meta} or AP_{nf} use a universal constant K (difference), which is indexed after program synthesis.

Sin ∘ Cos ∘ Tan ∘ t **Sin ∘ Cos ∘ Tan ∘ t**
Sin ∘ Cos ∘ t ∘ Tan ***Sin ∘ Cos ∘ t*** ∘ Tan
Sin ∘ Tan ∘ Cos ∘ t **Sin ∘ Tan ∘ Cos ∘ t**
Sin ∘ Tan ∘ t ∘ Cos ***Sin ∘ Tan ∘ t*** ∘ Cos
Sin ∘ t ∘ Cos ∘ Tan ***Sin ∘ t*** ∘ Cos ∘ Tan
Sin ∘ t ∘ Tan ∘ Cos ***Sin ∘ t*** ∘ Tan ∘ Cos
Cos ∘ Sin ∘ Tan ∘ t **Cos ∘ Sin ∘ Tan ∘ t**
Cos ∘ Sin ∘ t ∘ Tan ***Cos ∘ Sin ∘ t*** ∘ Tan
Cos ∘ Tan ∘ Sin ∘ t **Cos ∘ Tan ∘ Sin ∘ t**
Cos ∘ Tan ∘ t ∘ Sin ***Cos ∘ Tan ∘ t*** ∘ Sin
Cos ∘ t ∘ Sin ∘ Tan ***Cos ∘ t*** ∘ Sin ∘ Tan
Cos ∘ t ∘ Tan ∘ Sin ***Cos ∘ t*** ∘ Tan ∘ Sin
Tan ∘ Sin ∘ Cos ∘ t **Tan ∘ Sin ∘ Cos ∘ t**
Tan ∘ Sin ∘ t ∘ Cos ***Tan ∘ Sin ∘ t*** ∘ Cos
Tan ∘ Cos ∘ Sin ∘ t **Tan ∘ Cos ∘ Sin ∘ t**
Tan ∘ Cos ∘ t ∘ Sin ***Tan ∘ Cos ∘ t*** ∘ Sin
Tan ∘ t ∘ Sin ∘ Cos ***Tan ∘ t*** ∘ Sin ∘ Cos
Tan ∘ t ∘ Cos ∘ Sin ***Tan ∘ t*** ∘ Cos ∘ Sin
t ∘ Sin ∘ Cos ∘ Tan **t ∘ Sin ∘ Cos ∘ Tan**
t ∘ Sin ∘ Tan ∘ Cos **t ∘ Sin ∘ Tan ∘ Cos**
t ∘ Cos ∘ Sin ∘ Tan **t ∘ Cos ∘ Sin ∘ Tan**
t ∘ Cos ∘ Tan ∘ Sin **t ∘ Cos ∘ Tan ∘ Sin**
t ∘ Tan ∘ Sin ∘ Cos **t ∘ Tan ∘ Sin ∘ Cos**
t ∘ Tan ∘ Cos ∘ Sin **t ∘ Tan ∘ Cos ∘ Sin**

(a) Subfigure 1 (b) Subfigure 2

Fig. 9. Security

3. Individual coding (difference): coding of an individual is different. AP uses an integer index instead of direct representation as in canonical GP. GE uses the binary representation of an individual, which is consequently converted into integers for mapping into programs by means of BNF (O'Neill & Ryan, 2003).
4. Individual mapping (difference): AP uses DSH, while GP in its canonical form uses direct representation in LISP (Koza, 1998) and GE uses BNF.
5. Constant handling (difference): GP uses a randomly generated subset of numbers - constants (Koza, 1998), GE utilizes user determined constants and AP uses only one constant K for AP_{meta} and AP_{nf}, which is estimated by another EA or by nonlinear fitting.
6. Security procedures (difference): to guarantee synthesis of non-pathological functions, procedures are used in AP which redirect the flow of mapping into subsets of a whole set of functions and terminals according to the distance to the end of the individual. If pathological function is synthesized in GP, then synthesis is repeated. In the case of GE, when the end of an individual is reached, the mapping continues from the individual beginning, which is not the case in AP. It is designed so that a non-pathological program is synthesized before the end of the individual is reached (maximally when the end is reached).

2. Selected applications

This section briefly describes some selected applications of AP, which has been conducted during the past few years and cover a comparative study with GP techniques published by J. R. Koza as well as other different applications. In each subsection, the main idea of the AP application is described, results alongside references to publications, cumulating in the full report of proposed application.

2.1 Randomly generated solutions

This part discuss the first and very simple experiment (see Zelinka (2002b)) which has been done with AP. It was focused on the verification as to whether AP as it was programmed, is able to produce reasonable structures (programs, formulas, etc...) which are complete, i.e. there are not missing arguments, division by 0 etc. In this simulation, randomly selected functions from GFS were selected, i.e. randomly generated individuals has been transformed to programs by AP - no evolution has been taken into consideration in this experiment. The terminal set contains only one variable t and a few constants. The nonterminal set consisted of various and just randomly (by user) collected mathematical functions. Final solutions were synthesized from both sets. A few examples of synthesized formulas are represented by Equations (9) - (13). Selected formulas were also visualized to show interesting behavior of the synthesized programs. They are depicted in Figure 10 - 15. It is important to remember that there was no another deeper mathematical reason to synthesize such, on the first look wild, functions. There was only one aim - to check whether AP is able to synthesize structurally acceptable solutions. Instead of used mathematical functions, user defined functions can also be used. There was 1000 randomly generated individuals, converted into programs (formulas) and verified. No pathology in their structures has been observed.

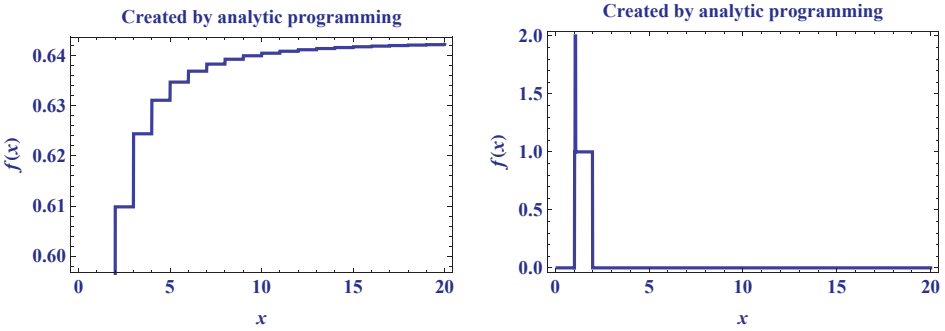
$$\cos^{-1}\left(\sec^{-1}\left(e^{-i\coth(1-it)}\right)^{\operatorname{sech}^{-1}\left(\left(\frac{\cosh^{-1}(K)}{\phi}\right)^{K}\right)}\right) \tag{9}$$

$$\cos^{-1}\left(\sec^{-1}\left(\frac{\log\left(\frac{\phi}{t}\right)}{\log(\cot(\gamma))}\right)^{\operatorname{sech}^{-1}\left(\operatorname{csch}^{-1}(t)^{-K}\right)}\right) \tag{10}$$

$$\tanh\left(\cot^{-1}\left(\frac{\phi}{\gamma}\right)^{\sinh^{-1}(e^{\phi-t})}\right) \tag{11}$$

$$\frac{\sqrt{1-\pi^2\sin^2\left(\cos\left(e^{\operatorname{sech}(\cosh(\cosh(\sin(t))))}\right)\right)}\csc\left(\cos\left(e^{\operatorname{sech}(\cosh(\cosh(\sin(t))))}\right)\right)}{\pi} \tag{12}$$

$$\sqrt{\operatorname{sech}^{-1}\left(\operatorname{sech}^{-1}\left(\exp\left(\sinh^{-1}\left(\frac{t^{\cos^{-1}(\gamma)}\phi^{\cos^{-1}(\gamma)}}{\cot^{-1}(A)}\right)\right)\right)\right)} \tag{13}$$



(a) (b)

Fig. 10. Visualization of randomly synthesized individuals - functions

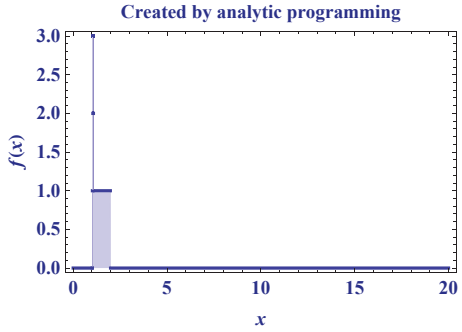
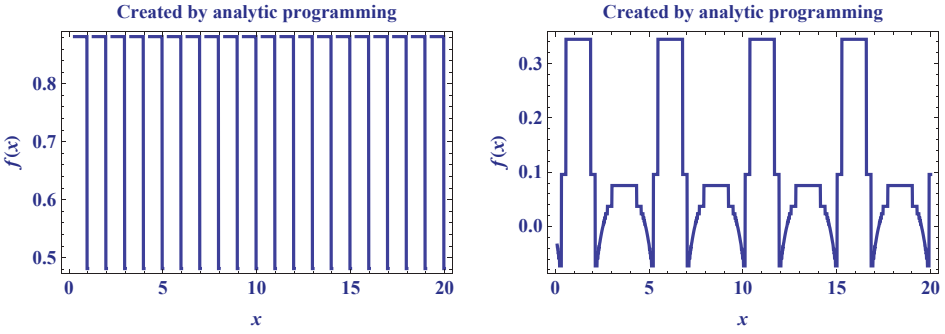
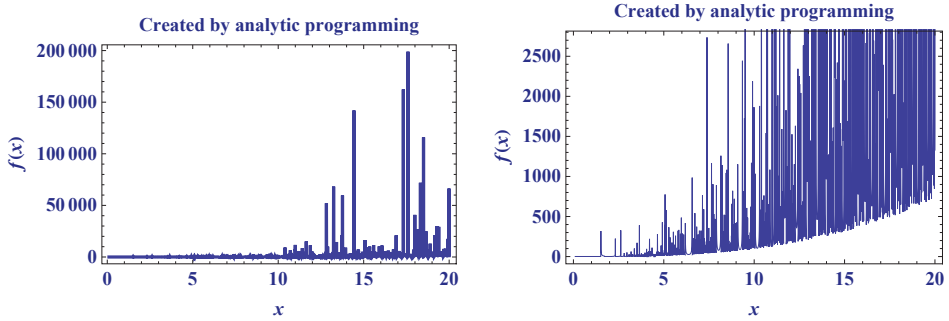


Fig. 11. Another view of the function from Figure 10b



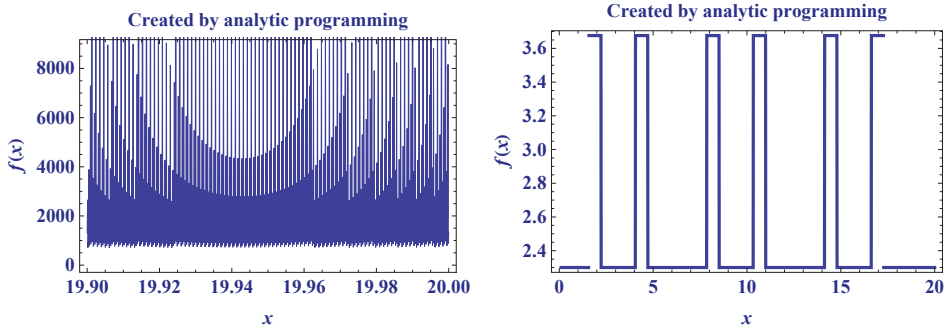
(a) (b)

Fig. 12. Visualization of randomly synthesized individuals - functions



(a) (b)

Fig. 13. Visualization of randomly synthesized individuals - functions, b) detailed view



(a) (b)

Fig. 14. a) More detailed view of Figure 13b and b) behavior of different function

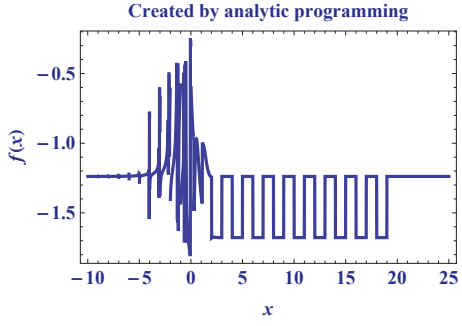


Fig. 15. Another interesting function

2.2 Comparison with selected GP examples

To verify more properly the functionality of AP, a set of comparative simulations based on selected examples from Koza's GP, have been done. Two algorithms were used for comparison of AP with GP - DE and SOMA. Simulations were focused on selected examples from Koza (1998) and Koza et al. (1999), especially:

- Sextic problem - $x^6 - x^4 + x^2$
- Quintic problem - $x^5 - 2x^3 + x$
- Boolean even-k-parity problem - synthesis of logical function in a few versions containing 3, 4, 5 and 6 input variables
- Boolean symmetry problem - synthesis of logical function in a few versions containing 3, 4, 5 and 6 input variables

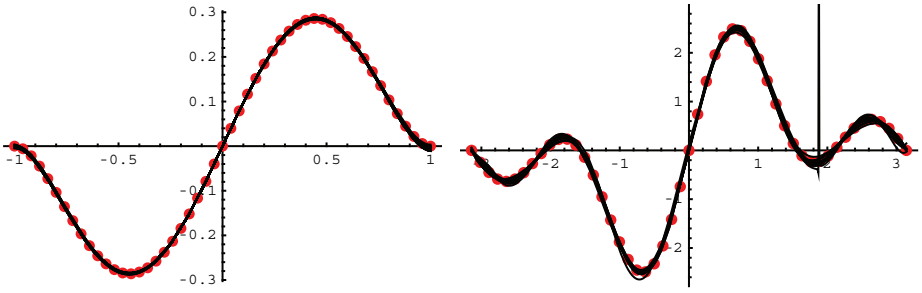
Based on the studies in Koza (1998) andKoza et al. (1999), the above mentioned problems have been selected for comparative study. The first two are focused on data fitting. Data are generated by means of polynomials $x^6 - x^4 + x^2$ and $x^5 - 2x^3 + x$. Equations 14 - 17 are typical example of synthesized solutions, especially Equations 14 and 16 are solutions with general constants K and Equations 15 and 17 are their fitted versions. In Figure 16, fitted data-dots and fitting by synthesized programs (black lines) is depicted. Another study - Booleans even-3-parity and symmetry problems were selected for comparative study and are fully reported in Zelinka et al. (2004) and Zelinka & Oplatkova (2004). In general, Boolean even-k-parity problems means that if the number of logical inputs of value True is even, then the output is True. If number of logical inputs of value True is not even, then the output is False. Number of all possible inputs (combinations) from $2^3 = 8$ for 3-parity problem to $2^6 = 64$ for a 6-parity problem. Truth table for 3-parity problem is given in Table 3. Symmetry problem has been investigated in the same way. Output of this logical function is True whenever True and False values are symmetrically distributed on inputs, see Koza (1998) andKoza et al. (1999). A typical example of synthesized solution is Equation 18. The full report of this comparative study is in Zelinka et al. (2004) and Zelinka & Oplatkova (2004).

$$\frac{xK[[5]]K[[6]](x(-K[[18]])-K[[19]]+2x)}{\frac{K[[23]](K[[24]]+x)}{x^2}+\frac{x(x-K[[20]])}{K[[21]]+K[[22]]}}+\left(\frac{K[[9]](xK[[10]]+x)(K[[25]]+x)}{K[[7]]K[[8]]}+K[[2]]-x\right)$$
$$(xK[[11]]-K[[12]]+x)-\frac{xK[[4]]\left(-K[[13]]+K[[14]]+K[[17]]+\frac{K[[15]]}{K[[16]]}-x\right)}{-K[[3]]-x}-K[[1]] \tag{14}$$

$$\frac{0.00621529x(0.793939-1.x)}{-1.x-0.934705}+(0.465773(x+2.82445)x-1.x-7.45208)(0.181218-0.749217x)+$$
$$\frac{2.9596(0.432881x-3.70673)x}{\frac{0.21213(x+13.054)}{x^2}+0.456758(x-0.562963)x}+1.27265 \tag{15}$$

$$x(x^2(x(K[[7]]+x)-K[[2]])+x(K[[4]]-K[[5]])+xK[[6]]+K[[1]]-K[[3]]-1) \tag{16}$$

$$x\left(1.-2.193908007555499^{-16}x+x^2(-2.+x(4.66960974116765^{-16}+x))\right) \tag{17}$$



(a) (b)

Fig. 16. Quintic (a) and Sinus 3 (b), all 50 successful simulations (black lines) very well fit the measured data (red dots)

Input 1	Input 2	Input 3	Output
True	True	True	False
True	True	False	True
True	False	True	True
False	True	True	True
True	False	False	False
False	True	False	False
False	False	True	False
False	False	False	True

Table 3. Truth table for Boolean even-3-parity problem according to Koza (1998)

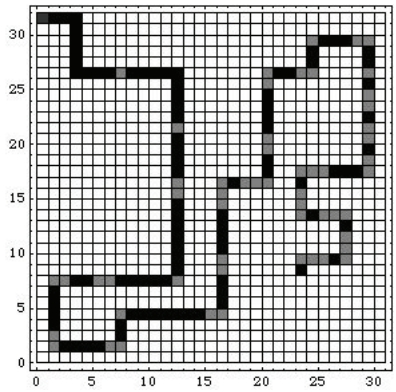
$$
\begin{aligned}
&((A \wedge (((((B \wedge A) \vee (C \wedge A) \vee (\neg C \wedge B) \vee (\neg C \wedge \neg A)) \,\overline{\wedge}\, ((B \wedge A) \vee (C \wedge A)\\
&\vee(\neg C \wedge B) \vee (\neg C \wedge \neg A))) \vee (B \,\overline{\vee}\, A)) \,\overline{\wedge}\, ((A \vee (B \wedge A) \vee (C \wedge A)\\
&\vee(\neg C \wedge B) \vee (\neg C \wedge \neg A)) \wedge B \wedge ((B \wedge A) \vee (C \wedge A) \vee (\neg C \wedge B) \vee (\neg C \wedge \neg A))))) \qquad (18)\\
&\overline{\wedge} C) \,\overline{\wedge}\, (C \vee (C \,\overline{\vee}\, (A \,\overline{\wedge}\, (C \,\overline{\wedge}\, ((B \wedge A) \vee (C \wedge A) \vee (\neg C \wedge B)\\
&\vee(\neg C \wedge \neg A))))))
\end{aligned}
$$

2.3 Santa Fe trail and Artificial ant

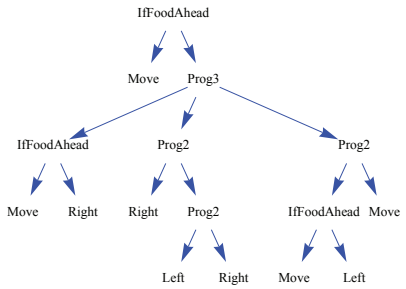
Another test of AP has been done on the setting of an optimal trajectory of an artificial ant on the so-called SantaFe Trail. In this experiment, SOMA (Zelinka, 2004) and DE (Price, 1999) were selected as the two EA's for simulation. In space and other comparative industries, the number of robots used for specific tasks are increasing daily. Subsequently, precise tasks such as optimal trajectory setting of the robot is a very desirable attribute. The problem description for this preliminary study is taken from Koza (1998). The aim of this experiment is that a robot, in this case, an artificial ant, should go through a defined trail and eat all food which is there. One of the possible ways for the ant to transverse is the so-called SantaFe trail, which is demonstrated in Figure 19. The SantaFe trail is defined as a 32 x 31 field, where food is set out. In Figure 19, a black field is food for the ant. The grey one is basically the same as a white field but for clarity, the grey color was used. The grey fields represent obstacles (fields without food on the road) for the ant. If obstacles do not exist, the ant would have a clear run. Ideally,

it would be the best to precede the ant and ascertain if any food is present in the current path. If food is present, then the ant would follow and eat it. If no food is found, the ant would change trajectory. This cycle would repeat till all the food is devoured.
However, in the real world, robots face obstacles during their movement. Therefore, this analogy is also applied in this research. The first obstacle which the ant has to overcome is the simple hole (position [8,27] in Figure 19. The second obstacle are the two holes in the line (positions [13,16] and [13,17]), or three holes ([17,15], [17,16], [17,17]). Next obstacle are the holes in the corners - one (position [13,8]], two ([1,8],[2,8]) and three holes ([17,15], [17,16], [17,17]). In the case of AP's application, all simulations (i.e. DE as well as SOMA) have shown very good performances and results were fully comparable with techniques like GP. For full report see Oplatkova & Zelinka (2006).

$$IfFoodAhead[Move, Prog3[IfFoodAhead[Move, Right], Prog2[Right, Prog2[Left, Right]], Prog2[IfFoodAhead[Move, Left], Move]]] \quad (19)$$



(a)



(b) Tree representation of the solution from the Equation 19

Fig. 17. SantaFe trail (a) and its solution (b)

2.4 Deterministic chaos synthesis

Zelinka et al. (2008) introduces the notion of chaos synthesis by means of EA's and develops a new method for chaotic systems synthesis. This method is similar to GP and GE and is applied with three EA's: DE, SOMA, and GA. The aim of this research is to synthesize new and "simple" chaotic systems based on some elements contained in a pre-chosen existing chaotic system and a properly defined cost function. The research consists of 11 case studies: the aforementioned three EA's in 11 versions. For all algorithms, 100 simulations of chaos synthesis were repeated and then averaged to guarantee the reliability and robustness of the proposed method. The most significant results are carefully selected, visualized and reported in this section.
Methods used in generating new chaotic systems from physical systems or from "manipulations" (e.g., control and parameter estimation (Grebogi & Lai, 1999; Hu et al., 1999) are based on deterministic mathematical analysis. Along with these classical methods, there

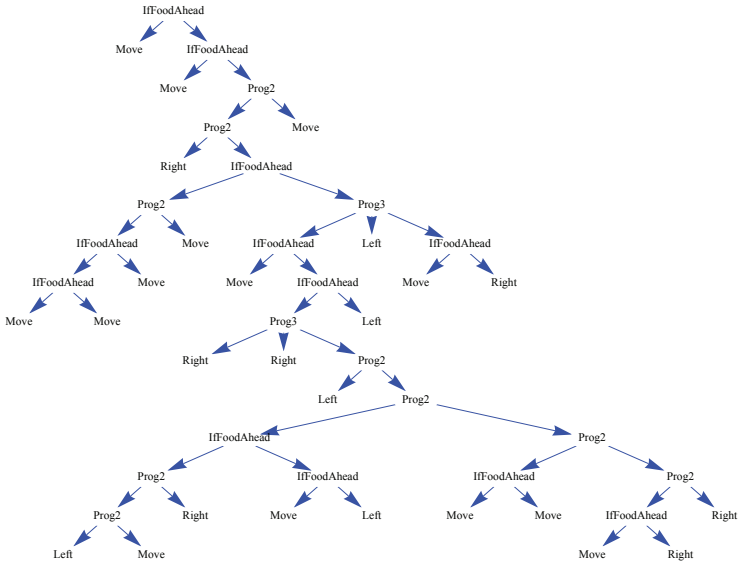


Fig. 18. Another more complex solution of the SanteFe Trail

are also numerical methods based partly on deterministic and partly on stochastic methods, called EA's (Back et al., 1997). EA's were used in searching solutions in many computationally hard problems including classes of P and NP problems (Garey & Johnson, 1979). In chaos studies, they were also used for chaos control as in Zelinka (2005), Richter (2002) or Zelinka (2006), amongst others.

The aim of the this research is to show that EA-based on GP-like techniques is capable of synthesizing chaotic behavior in the sense that the mathematical descriptions of chaotic systems are synthesized symbolically by means of EA's. The ability of EAs to successfully solve this kind of black-box problems has a proven track record (see, for example, Zelinka & Nolle (2005)), and is reinforced in this chapter.

Based on statistically robust simulations, a lot of interesting chaotic systems has been synthesized. In Zelinka et al. (2008), a number of chaotic systems in mathematical description as well as its bifurcation diagrams were reported. As an example Equation21 and Figure 19 can be used. The extended case study (for more EAs and continuous systems) is reported in the book chapter (Zelinka et al., 2010) where the synthesis of the continuous systems are taken into consideration.

$$\frac{Ax - A - x}{\frac{A}{2x} - \frac{A(A-x)}{Ax+2A-x+1} + A + x} \tag{20}$$

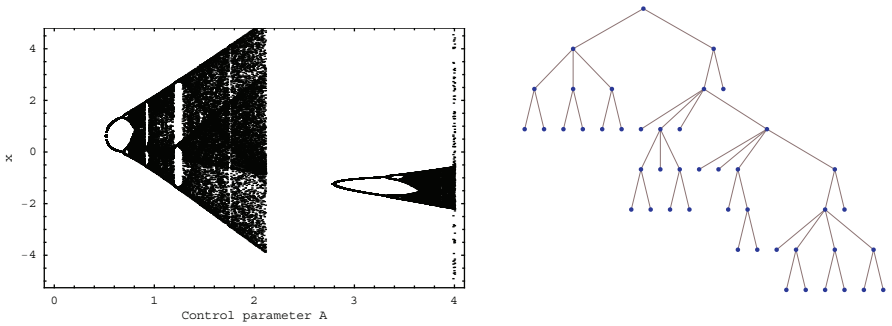


Fig. 19. Bifurcation diagram of the synthesized system (a) and solution (see Equation 21) in the tree representation (b)

2.5 Control law synthesis

Another application of AP (Oplatkova, Senkerik, Belaskova & Zelinka, 2010; Oplatkova, Senkerik, Zelinka & Holoska, 2010a;b) is focused on the synthesis of control law for discrete chaotic system. The interest in the control of chaotic systems has been an active area of research during the past decade. One of the first and important initial studies, of EA for control use was reported in Zelinka, Senkerik & Navratil (2009), Zelinka et al. (2006b) and Zelinka et al. (2006a), where the control law was based on the Pyragas method: Extended delay feedback control - ETDAS (Pyragas, 1995). Those papers were focused on the tuning of several parameters inside the control technique for a chaotic system. Compared to that, a presented paper Oplatkova, Senkerik, Belaskova & Zelinka (2010); Oplatkova, Senkerik, Zelinka & Holoska (2010a;b) shows a possibility as to how to generate the whole control law (not only to optimize several parameters) for the purpose of stabilization of a chaotic system. The synthesis of control is inspired by the PyragasâĂŹs delayed feedback control technique (Just, 1999; Pyragas, 1990). Unlike the original OGY control method (Ott et al., 1990), it can be simply considered as a targeting and stabilizing algorithm together in one package. Another big advantage of the Pyragas method is the amount of accessible control parameters. Instead of EA utilization, AP was used. Control law from the proposed system can be viewed as a symbolic structure, which can be created according the requirements for the stabilization of chaotic system. The advantage is that it is not necessary to have some “preliminary” control law and only to estimate its parameters. This system will generate the structure of the law also with suitable parameter values. The articles Oplatkova, Senkerik, Zelinka & Holoska (2010a), Oplatkova, Senkerik, Belaskova & Zelinka (2010) and Oplatkova, Senkerik, Zelinka & Holoska (2010b) contain 12 simulations with selected EAs applied with AP in order to synthesize suitable control law. Interested readers are recommended to read these articles.

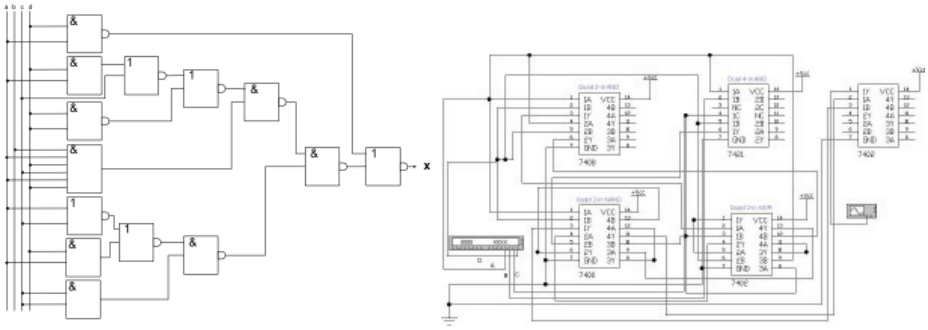
2.6 Algorithm synthesis

Our personal experiences have led to the hypothesis that a new algorithm (in this case evolutionary) can be created by AP. The main idea was that subroutines (operators) of selected EAs has been taken into consideration as symbolic objects (functions, ...), i.e. like members of the nonterminal set. Terminal set consisted of individuals (i.e. integer vector). The aim

of this simulation was to use EA's in such a way that with AP and a defined nonterminal and terminal sets, other versions of EA's were created. In Oplatkova (2009), the progress from the first study of the synthesis of a new EA to the simulations with more operators and higher dimensional systems is described. At the onset, DE was taken and its operators were separated into modules which were able to work independently. These operators were set up as simple functions for successful evaluations of AP. During the repeated simulations of AP, successful solutions as well as the original DE and other successful solutions (DE synthesis) were found. The next step continued with more operators from other evolutionary and stochastic algorithms such as SOMA, Hill Climbing and SA (Oplatkova, 2009). In this case, a new design of the cost function was utilized. With respect to the order of obtained cost values, the measurement was changed to minimize the difference between found extreme and the global. Penalization concerned to cost function evaluations was also applied. Simulations were performed in 2 dimensional space. This led to the third step, to use high dimensional benchmark functions as criterion in AP. The obtained results from higher dimensional test functions were then applied on 16 benchmark function in 2, 20 and 100 dimensional space for 4 found algorithms. Altogether, 192 simulations were carried out in 100 times repetition, equating to nearly 4×10^9 cost function evaluations. Results are depicted in tables and graphs in the Appendix of Oplatkova (2009). From results obtained, it can be stated that AP synthesized algorithms are able to optimize multimodal functions. Future research is open to add more operators, to tune parameters of found algorithms or to try to synthesize a new evolutionary operator itself.

2.7 Electronic circuits synthesis

In the diploma thesis of Strakos (2005), three electronic circuits were experimentally synthesized. The main point of this AP application was to confirm, that EA's with AP are possible to successfully design electronics circuits. In the first part of Strakos (2005) the general theory (GP, GE and AP) is outlined, while in the experimental part the synthesis of three electronic circuits (traffic light control, heat control and train station control) is described. All three control systems has been successfully designed by AP. Each winning solution was visualized as a circuit and hardware implementation (see for example Figure 20). In all three experiments (50 times repeated) AP had been observed to be capable of electronic circuit synthesis.



(a) Logical circuit designed by AP (b) HW reprezentation

Fig. 20. Circuit designed by AP (a) and its hardware implementation (b)

2.8 Nonlinear dynamical system identification

Synthesis, identification and control of complex dynamical systems are usually extremely complicated. When classics methods are used, some simplifications are required, which tends to lead to idealized solutions that are far from reality. In contrast, the class of methods based on evolutionary principles is successfully used to solve this kind of problems with a high level of precision. In this section, an alternative method of EA's, which has been successfully proven in many experiments like chaotic systems synthesis, neural network synthesis or electrical circuit synthesis. Zelinka, Senkerik, Oplatkova & Davendra (2009) discusses the possibility of using EA's for the identification of chaotic systems. The main aim of this work is to show that EA's are capable of the identification of chaotic systems without any partial knowledge of it's internal structure, i.e. based only on measured data. Two different EA's are presented and tested in a total of 10 versions. Systems selected for numerical experiments is the well-known logistic equation. For each algorithm and its version, repeated simulations were done, amounting to 50 simulations. Typical example of evolutionary identification is in Equation (21), (22) and visualization in Figure 21. According to obtained results, it can be stated that evolutionary identification is an alternative and promising way as to how to identify chaotic systems. Extended case study is also reported in Zelinka et al. (2010).

$$x\left(A+\frac{\left(-1-A+x-Ax+x^2-\frac{-A+x}{A}\right)\left(A+A(x+Ax)\right)}{2A^2}\right) \tag{21}$$

$$(1-x)x^2(3A+x-3Ax+Ax^2) \tag{22}$$

2.9 Neural network synthesis

Based on the case studies in Zelinka (2002c), Zelinka & Oplatkova (2003b), Zelinka & Oplatkova (2004) and Zelinka & Volna (2005), synthesis of Neural Networks (NN) was used for this study. Two problems, solved by AP were chosen : linearly and nonlinearly separable (XOR) problems. Concerning to previous simulations (Maniezzo, 1993), in this case of NN synthesis, simple elementary objects like $+, -, /, *, Exp, x_1, x_2, K$ were used. During evolution, more complicated structures from these simple objects of NN nature were created. As a learning algorithm, evolutionary fitting of weights was used, because the use of algorithms like back-propagation would be complicated on final neural structures. Figure 22 can be used as an example, where two examples of AP synthesized Artificial NN (ANN) is depicted. AP has been successfully used for different kind of problems. Positive results has shoved that AP can be used in this way. In the future more complex study on NN synthesis are going to be done by means of other EA's.

2.10 PDE solution synthesis

Zelinka (2001) outlines the use of AP on mathematical problems of civil engineering and problems of mathematical physics. In this paper, a few evolutionary simulations with AP has been done in order to get solutions of selected problems. This set of simulations was focused on ODE solving (see Figure 23 and Equation (23)). In Rektorys (1999), it is solved by the means of Ritz and Galerkin method on apriori selected functional base, which was orthogonal. Here it was solved with AP without any apriori demand.

This simulation was focused on finding the solution of a quite complicated ODE problem originating from mechanical engineering. Original solution obtained by means of Ritz

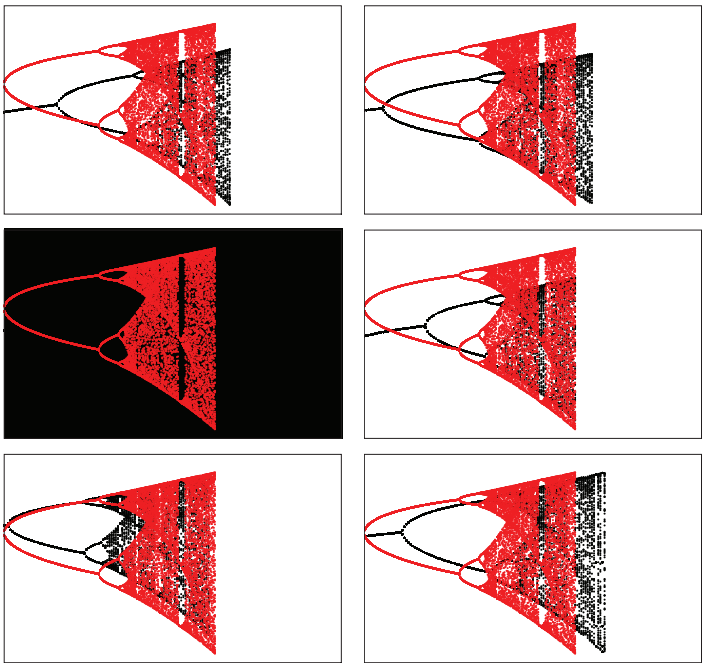
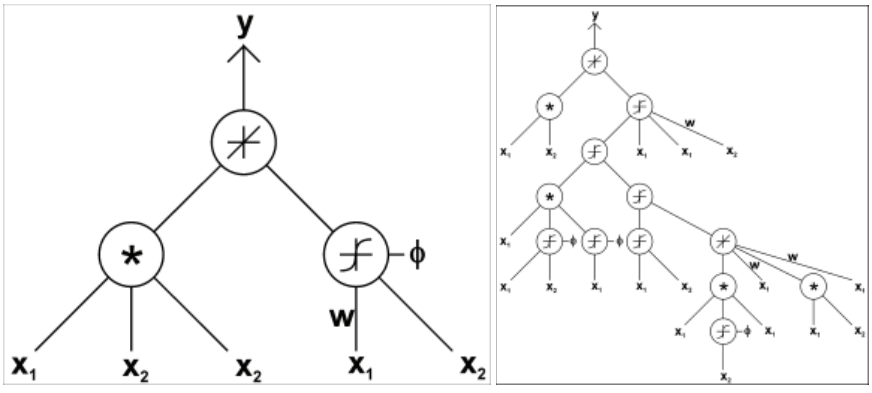


Fig. 21. Original (black fat points) and identified (red thin points) behavior



(a) ANN for linear separable problem (b) ANN for XOR problem

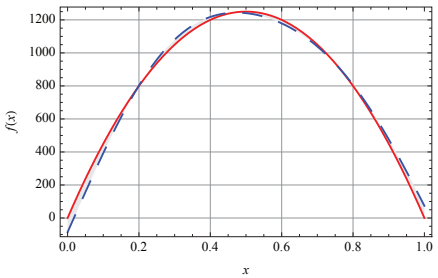
Fig. 22. Synthesized ANN by AP a) for linearly separable problems, b) for XOR problem

and Galerkin method (Rektorys, 1999) is depicted in Figure 23. Base in Hilbert space consisted of sinus functions and final solution of unknown u(x) was $u(x) = 1.243\sin(\pi x) + 0.0116\sin(2\pi x) + 0.00154\sin(3\pi x)$.
The original solution obtained by means of Ritz and Galerkin method and AP embedded with SOMA was used in two ways. In the first one, SOMA was used to estimate only parameters a, b, c of founded $u(x)$, i.e. $u(x) = a\sin(\pi x) + b\sin(2\pi x) + c\sin(3\pi x)$.. In the original solution all three coefficients were calculated by means of quite complicated Ritz and Galerkin methods. SOMA was able to find all three coefficients as is depicted in Figure 23. This problem was basically a classical optimization because functions were a priori known. This simulation was 100 times repeated and in all cases has lead to the same results. The second use of SOMA here was not focused only on parameter estimation. SOMA was used with AP on a complex set of functions ($sin, cos, ...$) operators : $+, -, /, Power, \times$ and constants a, b, c to find their best combination i.e. to build up function fitting function $5000(x - x^2)$ as closely as possible, see Equations 24 and 25.

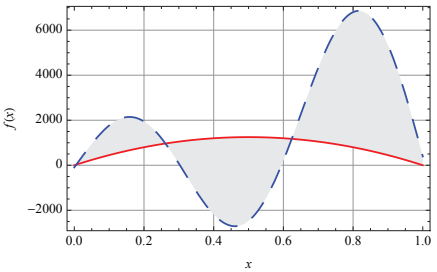
$$((4 + x)u(x)'')'' = 5000\left(x - x^2\right) \tag{23}$$

$$u = 1.243\sin(\pi x) + 0.0116\sin(2\pi x) + 0.00154\sin(3\pi x) \tag{24}$$

$$u = 1.243\sin(\pi x) - .3\sin(2\pi x) + .1\sin(3\pi x) \tag{25}$$



(a) The best solution (Equation 24) of the Equation 23



(b) Bad solution, see (Equation 25)

Fig. 23. Synthesized PDE solutions by AP a) the best solution, b) typical but not suitable solution observable at the beginning of the evolution. Original solution given by numerical Galerkin/Ritz method in the Hilbert space is represented by solid red line, while evolutionary synthesized solution is in dashed blue line.

3. Conclusion

Based on various applications of AP, it can be stated that AP seems to be powerful algorithmical equivalent of such methods like GP or GE. The AP method has been carefully tested during the last 9 years on various examples including selected examples from GP for comparative study. Results from all of experiments, partially reported here, confirm the fact that AP is possible to use for the same class of problems like another algorithms (GP, GE, etc.).

4. Acknowledgement

The following two grants are acknowledged for the financial support provided for this research: Grant Agency of the Czech Republic - GACR102/09/1680 and Grant of the Czech Ministry of Education - MSM7088352101.

5. References

Back, T., Fogel, D. & Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*, Institute of Physics.

Garey, M. & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York.

Grebogi, C. & Lai, Y. (1999). Controlling chaos, *in* H. Schuster (ed.), *Handbook of Chaos Control*, Wiley-VCH, New York.

Hu, G., Xie, F., Xiao, J., Yang, J. & Qu, Z. (1999). Control of patterns and spatiotemporal chaos and its application, *in* H. Schuster (ed.), *Handbook of Chaos Control*, Wiley-VCH, New York.

Johnson, C. (2004). Artificial immune systems programming for symbolic regression, *in* C. Ryan, T. Soule, M. Keijzer, E. Tsang & R. P. E. Costa (eds), *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 345–353.

Just, W. (1999). Principles of time delayed feedback control, *in* H. Schuster (ed.), *Handbook of Chaos Control*, Wiley-VCH, New York.

Koza, J. (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems., *Stanford University, Computer Science Department, Techincal Report* STAN-CS-90-1314.

Koza, J. (1998). Genetic programming, *MIT Press* .

Koza, J., Bennet, F., Andre, D. & Keane, M. (1999). *Genetic Programming III*, Morgan Kaufnamm, New York.

Koza, J., Keane, M. & Streeter, M. (2003). Evolving inventions, *Scientific American* pp. 40–47.

Lampinen, J. & Zelinka, I. (1999). Mechanical engineering design optimisation by differential evolution, *in* D. Corne, M. Dorigo & F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, London, UK, pp. 127–146.

Maniezzo, V. (1993). Searching among search spaces: Hastening the genetic evolution of feedforward neural networks, *in* C. R. R. R. F. Albrecht & N. C. Steele (eds), *Artificial Neural Nets and Genetic Algorithms*, Springer-Verlag.

O'Neill, M. & Brabazon, A. (2006). Grammatical differential evolution, *Proceedings of International Conference on Artificial Intelligence*, CSEA Press, pp. 231–236.

O'Neill, M. & Ryan, C. (2003). *Grammatical Evolution, Evolutionary Automatic Programming in an Arbitrary Language*, Springer-Verlag, New York.

Oplatkova, Z. (2005). Optimal trajectory of robots using symbolic regression, *Proceedings of 56th International Astronautics Congress*, Fukuoka, Japan.

Oplatkova, Z. (2009). *Metaevolution: Synthesis of Optimization Algorithms by means of Symbolic Regression and Evolutionary Algorithms*, Lambert Academic Publishing, New York.

Oplatkova, Z., Senkerik, R., Belaskova, S. & Zelinka, I. (2010). Synthesis of control rule for synthesized chaotic system by means of evolutionary techniques, *MEndel 2010*, Technical universitu of Brno, Brno, Czech Republic, pp. 91–98.

Oplatkova, Z., Senkerik, R., Zelinka, I. & Holoska, J. (2010a). Synthesis of control law for chaotic henon system - preliminary study, *ECMS 2010*, ECMS, Kuala Lumpur, Malaysia, pp. 277–282.

Oplatkova, Z., Senkerik, R., Zelinka, I. & Holoska, J. (2010b). Synthesis of control law for chaotic logistic equation - preliminary study, *AMS 2010*, ASM, Kota Kinabalu, Borneo, Malaysia.

Oplatkova, Z. & Zelinka, I. (2006). Investigation on artificial ant using analytic programming, *Proceedings of Genetic and Evolutionary Computation Conference*, Seattle, WA, pp. 949–950.

Oplatkova, Z., Zelinka, I. & Senkerik, R. (2008). Santa fe trail for artificial ant by means of analytic programming and evolutionary computation, *International Journal of Simulation, Systems, Science and Technology* Vol.9(No.3): 20–33.

Ott, E., Grebogi, C. & Yorke, J. (1990). Controlling chaos, *Phys. Rev. Lett.* Vol. 64: 1196 – 1199.

Price, K. (1999). An introduction to differential evolution, *in* D. Corne, M. Dorigo & F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, London, UK, pp. 79–108.

Pyragas, K. (1990). Continuous control of chaos by self-controlling feedback, *Phys. Lett. A,* Vol. 170: 421–428.

Pyragas, K. (1995). Control of chaos via extended delay feedback, *Physics Letters A* 206(206): 323–330.

Rektorys, K. (1999). *Variational methods in Engineering Problems and Problems of Mathematical Physics (Czech Ed.)*, Academia.

Richter, H. (2002). An evolutionary algorithm for controlling chaos: The use of multi-objective fitness functions, *in* M. Guervós, J. Panagiotis, A. Beyer, F. Villacanas & H. Schwefel (eds), *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 308–317.

Ryan, C., Collins, J. & O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language, *Lecture Notes in Computer Science*, First European Workshop on Genetic Programming.

Strakos, R. (2005). Design electronics circuits by means of evolution method, *Diploma thesis*, UTB Zlín, Zlin, Czech Republic.

Weisser, R. & Osmera, P. (2010a). Two-level tranpslant evolution, *17th Zittau Fuzzy Colloquium*, Zittau, Germany.

Weisser, R. & Osmera, P. (2010b). Two-level transplant evolution for optimization of general controllers, *New Trends in Technologies*, Sciyo.

Weisser, R., Osmera, P. & Matousek, R. (2010). Transplant evolution with modified schema of differential evolution : Optimization structure of controllers, *International Conference on Soft Computing MENDEL.*, Brno, Czech Republic.

Zelinka, I. (2001). Analytic programming by means of new evolutionary algorithms, *Proceedings of 1st International Conference on New Trends in Physics'01*, Brno, Czech Republic, pp. 210–214.

Zelinka, I. (2002a). Analytic programming by means of soma algorithm, *Proceedings of First International Conference on Intelligent Computing and Information Systems*, Cairo, Egypt, pp. 148–154.

Zelinka, I. (2002b). Analytic programming by means of soma algorithm, *Proc. 8th International Conference on Soft Computing*, VUT Brno, Mendel'02 Czech Republic, pp. 93–101.

Zelinka, I. (2002c). Analytic programming by means of soma algorithm, *First International Conference on Intelligent Computing and Information Systems*, Ain Shams University, ICICIS'02 Egypt.

Zelinka, I. (2004). Soma - self organizing migrating algorithm, *in* B. Babu & G. Onwubolu (eds), *New Optimization Techniques in Engineering*, Springer-Verlag, New York, pp. 167–218.

Zelinka, I. (2005). Investigation on evolutionary deterministic chaos control, *Proceedings of IFAC*, Prague, Czech Republic.

Zelinka, I. (2006). Investigation on realtime deterministic chaos control by means of evolutionary algorithms, *Proceedings of First IFAC Conference on Analysis and Control of Chaotic Systems*, Reims, France, pp. 211–217.

Zelinka, I., Chen, G. & Celikovsky, S. (2010). Chaos synthesis by evolutionary algorithms, *in* G. C. I. Zelinka, H. Richter & S. Celikovsky (eds), *Evolutionary Algorithms and Chaotic Systems*, Springer, Germany, pp. 357–398.

Zelinka, I., Guanrong, C. & Celikovsky, S. (2008). Chaos synthesis by means of evolutionary algorithms, *International Journal of Bifurcation and Chaos* Vol.18(No.4): 911–942.

Zelinka, I. & Nolle, L. (2005). Plasma reactor optimizing using differential evolution, *in* K. Price, J. Lampinen & R. Storn (eds), *Differential Evolution: A Practical Approach to Global Optimization*, Springer-Verlag, New York, pp. 499–512.

Zelinka, I. & Oplatkova, Z. (2003a). Analytic programming – comparative study, *Proceedings of Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems*, Singapore.

Zelinka, I. & Oplatkova, Z. (2003b). Analytic programming - comparative study, *Proceedings of The second International Conference on Computational Intelligence, Robotics, and Autonomous Systems*, CIRAS'03 Singapore, p. paper ID PS040204.

Zelinka, I. & Oplatkova, Z. (2004). Boolean parity function synthesis by means of arbitrarry evolutionary algorithms - comparative study, *8th World Multiconference on Systemics, Cybernetics and Informatics*, SCI 2004 Orlando, USA.

Zelinka, I., Oplatkova, Z. & Nolle, L. (2004). Boolean symmetry function synthesis by means of arbitrarry evolutionary algorithms - comparative study, *18th European Simulation Multiconference*, Gruner Druck, GmbH, ESM 2004 Magdeburg, Germany, pp. 143–148.

Zelinka, I., Oplatkova, Z. & Nolle, L. (2005a). Analytic programming – symbolic regression by means of arbitrary evolutionary algorithms, *Int. J. of Simulation, Systems, Science and Technology* Vol. 6(No. 9): 44–56.

Zelinka, I., Oplatkova, Z. & Nolle, L. (2005b). Analytic programming - symbolic regression by means of arbitrary evolutionary algorithms, *In: Special Issue on Inteligent Systems of International Journal of Simulation, Systems, Science and Technology* Vol.6(No.9): 44–55.

Zelinka, I., Senkerik, R. & Navratil, E. (2006a). Investigation on real time deterministic chaos control by means of evolutionary algorithms, *1st IFAC Conference on analysis and control of chaotic systems*, Université de Reims, CHAOS'06 Reims France, pp. 211–217.

Zelinka, I., Senkerik, R. & Navratil, E. (2006b). Optimization of feedback control of chaos by evolutionary algorithms, *1st IFAC Conference on analysis and control of chaotic systems*, Université de Reims, CHAOS'06 Reims France, pp. 97–102.

Zelinka, I., Senkerik, R. & Navratil, E. (2009). Investigation on evolutionary optimitazion of chaos control, *CHAOS, SOLITONS and FRACTALS* 40(1): 111–129. doi:10.1016/j.chaos.2007.07.045.

Zelinka, I., Senkerik, R., Oplatkova, Z. & Davendra, D. (2009). Evolutionary identification of chaotic system, *2nd IFAC Conference on analysis and control of chaotic systems*, Queen Mary, University of London, CHAOS'09 London UK.

Zelinka, I. & Volna, E. (2005). Neural network synthesis by means of analytic programming - preliminary results, *Proc. 11th International Conference on Soft Computing Mendel'05*, Technical universoty of Brno, Mendel'05 Brno Czech Republic, p. paper 504.

10

PPCea: A Domain-Specific Language for Programmable Parameter Control in Evolutionary Algorithms

Shih-Hsi Liu[1], Marjan Mernik[2], Mohammed Zubair[1],
Matej Črepinšek[2] and Barrett R. Bryant[3]
[1]*California State University, Fresno,*
[2]*University of Maribor,*
[3]*University of Alabama at Birmingham,*
[1,3]*USA*
[2]*Slovenia*

1. Introduction

An Evolutionary Algorithm (EA) is a meta-heuristic and stochastic optimization search process that mimics Darwinian evolution theory and Mendel's Genetics. Each process facilitates (a) population(s) evolve into fittest and/or convergence by setting parameters of selection, mutation, crossover, population resizing, and/or many other variant operators. However, due to two primary identified factors, EAs are still a challenging research topic: (1) Value choices/ranges for parameters (i.e., parameter settings) will greatly influence the evolution performance of a search process in terms of fittest and/or convergence; and (2) Parameter settings that are good for one fitness function do not guarantee the same evolution performance of another fitness function. Namely, parameter settings are function-specific. Different functions may have various characteristics that request specific attention. In order to better organize and overcome the parameter setting problem, Eiben et al. have classified parameter settings into parameter tuning and parameter control (Eiben et al., 1999): Parameter tuning determines parameter values before a search process begins while parameter control changes parameter values during a search process. More specifically, parameter control adjusts parameters on-the-fly using three different approaches: (1) Deterministic approach alters parameters based on certain pre-determined rules or formulae; (2) Adaptive approach strategically adjusts parameter values based on the feedbacks of a search process. Such feedbacks could be fitness, diversity, distance, among others; and (3) Self-adaptive approach encodes parameters to be adapted and evolves them along with a search process. Yet, even with such a classification, to our best knowledge there is no existing tool to assist researchers with conducting experiments of parameter settings with ease. Namely, researchers need to find out appropriate places out of thousand lines of EA source code to introduce and update specific parameters (including feedbacks) as well as formulae and adaptive strategies. Additionally, a number of revisions for EA source code will be also required for different kinds of experiments. To EA experimenters, such endeavor is time consuming and error prone. To EA developers, complex and tangling source code, resulted from different

parameter and strategy introductions, may also cause inflexibility for further extension and inevitability of faulty EA source code. In order to solve the aforementioned problems, a programmable approach, called PPCea (Programmable Parameter Control for Evolutionary Algorithms) (Liu et al., 2004), is presented in this book chapter.

PPCea is a Domain-Specific Language (DSL) (Mernik et al., 2005) for EAs. It uplifts the abstraction layer to a higher (i.e., domain-specific) level and introduces domain-specific notations (e.g., parameters and statements) as well as common linguistic elements. Namely, the implementation details of Genetic Algorithms (GAs) and Evolution Strategies (ESs) are encapsulated and hidden so that EA experimenters are able to experiment with evolutionary algorithms and obtain statistical results by programming a few PPCea statements. Additionally, the flexible programming fashion also enhances the possibility of reproducing existing EA experiments in a simpler PPCea source code and likely introducing new experiments to facilitate even better optimization search or faster convergence.

For EA experimenters, the first part of this book chapter introduces PPCea with examples to demonstrate PPCea's capabilities and usability. Famous existing parameter tuning and parameter control examples are reproduced using PPCea (e.g., Fogarty's formula (Fogarty, 1989), PROFIGA (Eiben et al., 2004), and 1/5 success rule (Bäck & Schwefel, 1995)). Additionally, new examples are also demonstrated to show the flexibility of PPCea. For example, introducing new metrics as feedbacks for parameter control and adaptively switching among different operators during an evolutionary process can be done with ease.

For EA developers, design and implementation of PPCea are covered in the second part of the book chapter. In this part, DSL patterns and design patterns are utilized. Coding and/or UML examples are presented and discussed to show how such patterns lessen the extension problems during development and maintenance phases. Software metrics are also measured to prove the effectiveness of design patterns for modularization and extension. In summary, PPCea is a domain-specific tool that is "win-win" to both EA experimenters and developers: For EA experimenters, the programmable fashion and high level abstraction allow EA users to conduct EA experiments in a productive manner. For EA developers, the design and implementation of PPCea allow evolutionary algorithms, operators, algorithms of operators (i.e., strategies), and parameters to be introduced or revised painlessly.

The book chapter is organized as follows. By using grammars, code snippets, and UML diagrams, Sections 2 and 3 respectively introduce PPCea from the perspectives of experimenters and developers. Section 4 discusses related work on parameter settings in Evolutionary Algorithms. PPCea's capabilities, limitations, and future directions are concluded in Section 5.

2. PPCea: A Painless Problem Curer for EA users

Because of the meta-heuristic and stochastic characteristics towards searching optimization, experimenters or users of EAs are inevitably requested to perform a sufficient number of experiments. Needless to say, there are numerous combinations and scopes of domain-specific parameters (e.g., mutation rate, crossover rate, and selection pressure) need to be tuned or controlled so that fittest and/or convergence can be discovered. A primary objective of PPCea is to become a problem curer for EA users/experimenters to conduct experiments painlessly. We first introduce PPCea through a number of examples categorized by Eiben et al.'s classification suggestions. The grammar of PPCea is appended at the end of the chapter for interested readers.

2.1 PPCea for parameter tuning

Parameter tuning is an approach for EA experiments classified in (Eiben et al., 1999). Such a kind of experiments determines the parameters of an evolutionary process before it runs and will not change the parameter values during the process. Many of the existing EAs are classified into this category. Per their endeavors, common guidelines for setting mutation and crossover rates in GAs are as follows: mutation rate (*pm*) ≒ 1/(the bit length of an individual in genetic algorithms) and crossover rate (*pc*) ≒ 0.75~0.95. PPCea can reproduce such experiments easily. Figure 1 shows that twenty experiments of Ackley's function from (Yao et al., 1999) with different parameter tuning settings are defined using Grefenstette's guideline (Grefenstette, 1986). Also, if one does not want to reset different values for *pm, pc,* or any domain-specific parameters for each experiment, formulae may be defined to adjust the values of such parameters as seen in the italic part of Figure 1, where the *if*-statement within the *while*-statement adjusts *pm* every 5 experiments. The experimental results of three reproduced parameter tuning-based experiments are available at (Liu, 2010).

```
genetic
    readfile weightF10.txt; //load coeff. of Ackley's function from Yao et al., 1999
    Function := 10; //load Ackley's function from Yao et al., 1999
    Round := 20; //number of experiments
    Maxgen := 1000; //maximum generation of an evolutionary process
    pm := 0.001; //set mutation rate
    pc := 0.95; //set crossover rate
    r := 0;
    while ( r < Round ) do
        init; //initialize population
        callGA; //invoke an evolutionary process of GA
        if (( r % 5) == 0) then
            pm := pm + 0.001 //change pm every 5 experiments
        fi;
        r := r + 1
    end;
    writeresult //output the experimental results to text and Excel files
end genetic
```

Fig. 1. Parameter tuning using PPCea

2.2 PPCea for deterministic parameter control

```
genetic
//skip initializing Round, Maxgen, Popsize, Epoch, pm, alpha, beta, gamma, length, r, g
    while ( r < Round ) do
        init; //initialize population
        while ( g < Maxgen ) do
            callGA; //invoke an evolutionary process of GA
            pm := sqrt(alpha / beta) * exp((0 - gamma)*g/2) / (Popsize / length);
        // the above formula is from Hessen & Manner
        // pm := 1 / (2+(( length-2 )/Maxgen)* g )
        // the above formula is from Bäck & Schütz
            g := g + Epoch // Generation stride for parameter control adaptation
        end;
        r := r + 1
    end;
    writeresult //output the experimental results to text and Excel files
end genetic
```

Fig. 2. Deterministic parameter control using PPCea

An important advantage of PPCea over other EA frameworks or software is its capability of performing parameter changes on-the-fly through a programmable fashion. Deterministic parameter control is an approach that defines how to change parameters during an evolutionary process using formulae. Fogarty (Fogarty, 1989) proposed one of the earliest deterministic approaches that adjusts mutation rate to a smaller value along with generations in order to tend from exploration towards exploitation. Liu et al. has published the experimental results of five unimodal and seven multimodal functions using Fogarty's mutation rate formula in (Liu et al., 2009). Figure 2 reproduces (Hesser & Männer, 1991)'s and (Bäck & Schütz, 1996)'s mutation formulae to show that PPCea is capable of representing more sophisticated cases.

2.3 PPCea for adaptive parameter control

Different from deterministic parameter control that does not interact with the evolutionary process that it controls, adaptive parameter control utilizes the analysis results from the evolutionary process and then determines which directions the evolutionary process may move forward by changing the parameters of associated operators. PPCea has reproduced 1/5 success rule (Bäck & Schwefel, 1995) and population resizing (Smith & Smuda, 1995) and introduced an entropy-driven approach (Liu et al., 2009) to adapt an evolutionary process. Figure 3 shows that PROFIGA (Eiben et al., 2004) is reproduced by PPCea.

PROFIGA is a GA that utilizes population resizing to balance between exploration and exploitation. As seen in the first *if*-statement in the figure, if the best fitness is improved, then population size will be increased proportionally so that more exploration can be promoted. Similarly, if the evolutionary process is not improved every *kgen* generations, the population size will be proportionally increased using the same factor (*growFactorX*). The second *if*-statement performs such an objective. Note that *growFactorX* is a negative value so that the formula within the second *resize* uses subtract operator. Lastly, the last *if*-statement shows that if neither the first nor the second conditions hold, the evolutionary process will tend to exploitation by shrinking the population size.

Of course, PPCea is not almighty. For example, GAVaPS (Arabas et al., 1994) and APGA (Bäck et al., 2000) perform population resizing based on aging concept. Such algorithms cannot be reproduced by current PPCea due to absence of age in individuals. Yet, once age is introduced along with associated operators, PPCea is capable of performing GAVaPS and APGA without a doubt. Similarly, parameter-less GA (Harik & Lubo, 1999) introduces a number of populations with different sizes to compete with each other. Because PPCea currently does not introduce multi-populations, reproducing parameter-less GA is also questionable. Because the design and implementation of PPCea facilitate extension and evolution, new algorithms like GAVaPS, APGA, parameter-less GA, and other EAs may be introduced with ease. More discussions on how to utilize such design and implementation advantages to introduce new algorithms will be covered in Section 3.

2.4 PPCea for adaptive operator control

Adapting parameters on-the-fly is not new in EAs. What about adapting operators on-the-fly? Adapting operators may be classified into three categories:

1. Operator adaptation is delegated to parameter control. Such an adaptation is done by adjusting parameters associated to specific operators. For example, 1/5 success rule utilizes mutation success rate to determine if mutation rate needs to be tuned up or

```
genetic
// initialize all needed parameters. bestImproved and noImprovedForLong are false
    init;
    initBest := Best; // best fitness from initial population
    nextBest := initBest;
    while (g < Maxgen) do
        currBest := nextBest; // best fitness from the current population
        callGA; // invoke an evolutionary process
        nextBest := Best;  // best fitness from the population of next generation
        growFactorX := factor * (Maxgen - g )* Popsize * (nextBest - currBest) /
        initBest;
        if ((nextBest - currBest) > 0) then //best fitness improved
            resize(Popsize * (1 + growFactorX));
            bestImproved := true
        fi;
        if ((nextBest - currBest) < 0 ) then //best fitness not improved for kgen
            i := i + 1;
            if ( i == kgen ) then
                i := 0
            fi;
            resize(Popsize * (1 - growFactorX)); //Popsize increase
            noImprovedForLong := true
        fi;
        if ((bestImproved != true) && (noImprovedForLong != true)) then
            resize(Popsize * (1 - 0.05))
        fi;
        g := g + Epoch;
    end
end genetic
```

Fig. 3. PROFIGA reproduction using PPCea

down. Similarly, selection pressure assists in adjusting the performance of selection operator in terms of fitting offspring. Usually adaptation in this category is classified as parameter control (Eiben et al., 1999);

2. Instead of focusing on the effectiveness of a specific operator using parameter control, an evolutionary process may switch among different operators based on certain real-time feedbacks. For example, Ursem (Ursem, 2002) introduced Diversity-Guided Evolutionary Algorithm (DGEA) that splits an evolutionary process into exploration and exploitation modes based on diversity. Under exploitation mode, recombination and selection are active. Otherwise, mutation is in charge;
3. Adaptation can be also done by switching among different variants of the same type of operators (Herrera & Lozano, 1996). For example, switching from one-point mutation to N-point mutation may result in more exploration during an evolutionary process, and switching from linear selection to non-linear one may change the influence weight of certain portion of individuals.

For (1), it has been discussed in the previous subsection. This subsection first reproduces DGEA falling into category (2) and then proposes how PPCea expresses experiments in category (3).

DGEA introduces a new diversity metric that computes the distance of all individuals to the average point of an N-dimensional search space. Exploration mode is identified if the diversity metric is lower than a predefined lower bound, and exploitation mode is recognized as the metric is higher than a predefined higher bound. Selection and crossover are applied to explore search space while mutation is treated as exploitation operator. Figure 4 shows the reproduction of DGEA, where *dLow* and *dHigh* are user-defined parameters and *DistanceToAvgPt* is computed by PPCea. *changeStrategy* is a PPCea statement

that switches between operators. For example, within exploration mode, tournament selection and 1-point crossover are active and mutation is halted. Conversely, mutation is the only active operator while selection and crossover are halted by the two keywords specified within the second *changeStrategy*.

```
genetic
// initialize all needed parameters including dLow, dHigh
    init;
    while ( g < Maxgen ) do
         callGA;
         if ( DistanceToAvgPt < dLow ) then // exploration mode
             changeStrategy(TOURNAMENT_SELECTION, GA_HALT_MUTATION,
         ONE_PT_CROSSOVER)
         fi;
         if ( DistanceToAvgPt > dHigh ) then // exploitation mode
             changeStrategy(GA_HALT_SELECTION, ONE_PT_MUTATION, GA_HALT_CROSSOVER)
         fi;
         g := g + 1
    end
```

Fig. 4. DGEA reproduction using PPCea

The previous example shows that PPCea can swap between different operators when needed. Halting an operator under a specific condition is also feasible by setting *GA_HALT_SELECTION*, *GA_HALT_MUTATION*, and *GA_HALT_CROSSOVER*, among others. When PPCea interpreter identifies such keywords, the operators will not be executed until they are reactivated by next *changeStrategy* statement. More details about how these are implemented will be covered in Section 3.

As mentioned before, DGEA is within category (2). A PPCea example classified within category (3) is introduced in Figure 5. Initially, *context*, a PPCea statement, defines specific operators will be executed by the evolutionary process. Line 1 shows that linear selection, 1-point mutation, and n-point crossover are picked to perform optimization search at the beginning. Lines 5 to 10 shows that two operator pairs (*TOURNAMENT_SELECTION*, *N_PT_MUTATION*) and (*RANK_SELECTION*, *ONE_PT_MUTATION*) will be swapped every 10 generations until 95th generation. Mutation will be stopped at the last 5 generations. Because *N_PT_CROSSOVER* never appears in the pairs of *changeStrategy*, this operator will remain active during the entire evolutionary process. How PPCea interpreter executes such operator adaptation will be also covered in Section 3.

2.5 Summary

As can be seen in the previous examples, the programming fashion of PPCea facilitates introducing a number of experiments with same or different settings by writing a few lines of code. Each evolutionary process run by PPCea can also be controlled deterministically or adaptively through parameter and/or operator adaptation. For space consideration, the experimental results of the examples, acting as a proof of feasibility of PPCea, are available at (Liu, 2010). Note that the previous examples also show some EAs cannot be reproduced easily derived from lacking needed attributes, multi-populations, parameters analyzed from an evolutionary process or operators. Section 3 attempts to address such problems from the perspective of EA developers. Lastly, categories of adaptive representation and adaptive fitness are also introduced in (Herrera & Lozano, 1996). They could be also potentially addressed by PPCea. Due to time constraint, they are left as one of our future work.

```
1  context(LINEAR_SELECTION, ONE_PT_MUTATION, N_PT_CROSSOVER);
2  init;
3  while (g <= Maxgen ) do //assume t = 1 initially and Maxgen = 100
4      callGA;
5      if (( g % 10) == 0) then
6          changeStrategy(TOURNAMENT_SELECTION, N_PT_MUTATION)
       //swap to tournament selection every 10 generations starting at g = 10
7      fi;
8      if (( g % 20) == 0) then
9          changeStrategy(RANK_SELECTION, ONE_PT_MUTATION)
       //swap to rank selection every 10 generations starting at g = 20
10     fi;
11     if (g > 95) then
12         changeStrategy(GA_HALT_MUTATION)
       //swap to temporarily stop mutation between generations 95 and 100
13     fi;
14     g = g + 1
15 end;
```

Fig. 5. Operator adaptation using PPCea

3. PPCea: A Portable Pattern-driven Contrivance for EA developers

Conducting parameter control experiments is always a time consuming task due to a variety of possible parameter combinations that may affect the convergence and optimization of an evolution process in different magnitudes. One may concentrate on a limited set of parameters to "de-scope" the problem (Harik & Lobo, 1999). Even so, a sufficient number of experiments are still needed due to heuristic nature of EAs. Per Aristotle, the aforementioned problems are essential difficulties (Brooks, 1987) inherent in EAs. Conversely, code snippets for computing metrics and programming logics for adapting an evolution process on-the-fly based on such metrics still scatter and tangle with other EA source code. Such inflexibility for further extension and inevitability of faulty EA source code are accidental difficulties (Brooks, 1987) that may be solved by the approaches hiding such difficulties.

A DSL is a modeling/programming language that shields accidental difficulties by introducing a higher level abstraction. It has been proved that DSLs may facilitate productivity (up to 10 times improvement), reliability, maintenability, and portability to domain users (Mernik et al., 2005). However, DSLs that are implemented by compiler or interpreter approaches may result in extension and evolution difficulties (Gray et al., 2008). For example, if a new mutation operator is introduced to PPCea, not only new syntax and semantics need to be introduced, but exsiting source code may be also affected due to inappropriate modularization in many compiler/interpreter-based DSLs including PPCea. Moreover, as mentioned in (Harik & Lobo, 1999), Holland would have never thought of a plentiful number of parameters are presented – Parameters are good for assisting in getting insight of an evolution process or helping control the process. Yet, EA computation may become overwhelmly slow resulted from parameter explosion. In summary, an objective of this section is to remedy the obstacles dervied from the introduction, extension, or evolution of parameters and operators in PPCea.

3.1 Design of PPCea

In order to design and implement PPCea in a manageable and systematic way, DSL patterns (Mernik et al., 2005) and design patterns (Gamma et al., 1995) are followed. Table 1 summarizes the DSL patterns that PPCea applies.

Workflow	Pattern	Description
Decision	Task automation System front-end	When and why to have PPCea
Analysis	FODA (Kang et al., 1990)	Find the common and variable features of EAs
Design	Denotational Semantics Design Patterns (Gamma et al., 1995)	Formally define syntax and semantics of PPCea PPCea applies composite, visitor, strategy, decorator and singleton patterns to address introduction, extension, and evolution problems.
Implementation	Interpreter	Introduce PPCea interpreter to conduct EA experiments

Table 1. The DSL patterns applied in PPCea

Decision patterns specify when and why a new DSL is essential. In order to provide an adaptable mechanism to solve such parameter control/setting problems, task automation and system front-end decision patterns are chosen. Task automation decision pattern hides the implementation details of EAs. Without browsing and understanding lengthy source code encapsulated in EAs, users omit the complex implementation but concentrate on the parameters and operators that lead to the optimization and/or convergence of EAs. Secondly, PPCea follows system front-end decision pattern that primarily handles configurations. Time-consuming and error-prone overhead can be reduced or avoided. As for analysis, PPCea utilizes Feature-Oriented Domain Analysis (Kang et al., 1990) to perform formal domain analysis so that common and variable features of EAs can be systematically identified. With such, PPCea can be formally defined using denotational semantics (Aho et al., 2007) and designed using design patterns (Gamma et al., 1995). Lastly, interpreter pattern is utilized to implement PPCea. An overview of PPCea interpreter is shown in Figure 6.

The interpreter is constructed with the assists of JFlex (Klein, 2010) and Construction of Useful Parsers (CUP) (Hudson, 2010). JFlex is a fast scanner generator for Java whose purpose is to generate a lexer that performs tokenization process for PPCea programs. CUP is a parser generator that introduces a bottom-up parser that performs syntax analysis. Such a parser may be integrated with user-defined semantics written in Java, accompanying with options to introduce syntax trees and symbol tables. The linguistic elements include commonly-seen constructs such as if-else, loop, and assignment statements as well as expressions and operators to perform necessary parameter adjustments. Additionally, domain-specific elements to describe an EA are presented: *init* statement initializes a population, *callGA* statement performs a GA, *callES* statement performs an ES, *resize* statement allows population resizing, *changeStrategy* statement offers the potentials to switch between different operators on-the-fly, *context* statement determines the operators that constitutes an EA, and *require* statement determines which domain-specific parameters to be computed. Such parameters are either the results from an evolutionary process that can be also acted as metrics or feedbacks to assist parameter control. There are also miscellaneous statements for various purposes (e.g., *IOStatement*). Interested readers may find more information on the PPCea web page (Liu, 2010). All the above linguistic elements

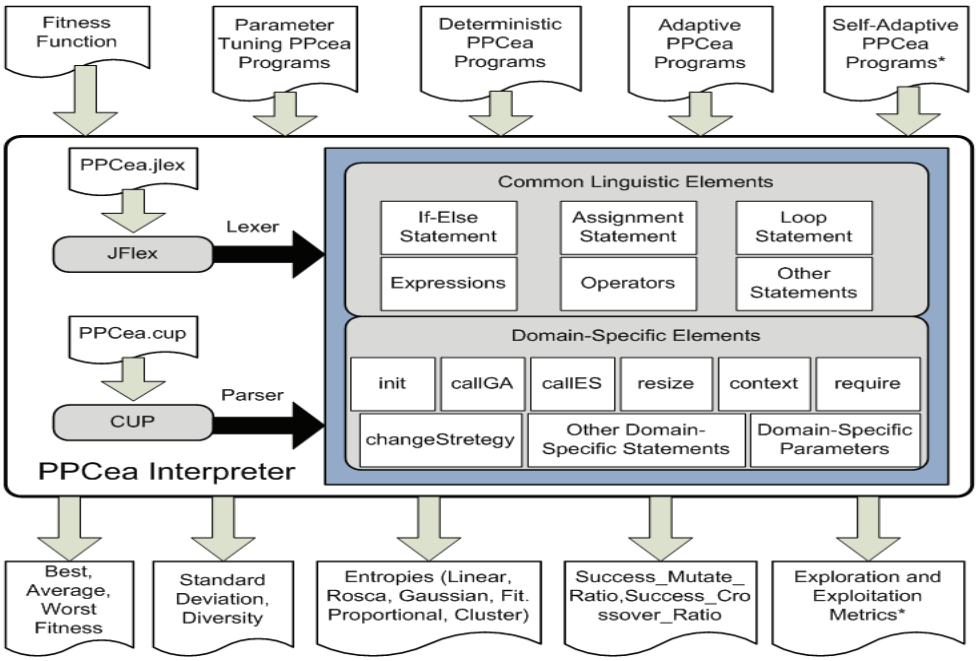


Fig. 6. An overview of PPCea interpreter (* means ongoing/future tasks)

are represented as Java classes embedded with associated semantics. The interpreter currently accepts parameter tuning, deterministic, and adaptive PPCea programs as inputs, as seen at the top of the figure. The outputs, at the bottom of the figure generated by the interpreter, comprise best, average, worst fitness, standard deviation and Euclidean distance (i.e., diversity), entropy (Liu et al., 2009), and the success rates of crossover and mutation, among others. More domain-specific constructs and parameters can be introduced, extended and evolved following the design patterns introduced in the subsequent subsections.

3.2 Evolutionary Algorithm and operator introductions

Since Evolutionary Algorithms (EAs) were coined, there have been a variety of algorithms, operators, and parameters proposed in order to apply to a various number of applications and experiments as well as further improve optimal results and/or convergence rate of such algorithms. For example, different from the general sketches of GAs in (Michalewicz, 1996), Bi-population GA (Tsutsui et al., 1997), aGA (Ghosh et al., 1996), and PROFIGA (Eiben et al., 2004), among others are variations that respectively introduces new algorithmic strategies (e.g., splitting populations into exploration and exploitation modes), new attributes (e.g., ages for individuals) or new operators (population resizing) to facilitate optimization and/or convergence. Additionally, DGEA, Evolution Strategies using Cauchy Distribution (Yao et al., 1999), Particle Swamp Optimization (Kennedy and Eberhart, 2001), and Differential Evolution (Storn & Price, 1997), to name a few, are also categorized in EAs that solve optimization problems from other perspectives. In addition to algorithm

introductions, many variations of existing operators and domain-specific parameters are also introduced (e.g., tournament selection, linear selection, uniform crossover, intermediate crossover, diversity-to-average measure (Ursem, 2002), and cluster entropy (Liu et al., 2009)). PPCea has anticipated such extension and evolution potentials and hence adopted design patterns so that future changes can be addressed with ease.

3.2.1 Evolutionary Algorithm and operator introductions using composite pattern

Because PPCea is developed by following the interpreter/compiler pattern (Mernik et al., 2005), inevitably, the syntactical representation of a PPCea program is expressed as a syntax tree structure (Aho et al., 2007). However, a commonly-seen implementation issue existing in such a tree structure is to deal with the composite-atomic hierarchies (i.e., whole-part hierarchies). For example, PPCea comprises if-else and loop statements that may embrace zero or more composite and/or atomic statements as child nodes (e.g., a nested if-else statement); and conversely, assignment and domain-specific statements are atomic ones that cannot hold any statement nested within their bodies. Because composite statements are derived from recursive productions defined in PPCea grammar (see appendix), they do not posses concrete semantics as other statements do. To reduce implementation complexity, a synergistic objective needs to be fulfilled: How to uniformly treat composite and atomic language constructs in the tree structure (i.e., hide the differences), while distinctions between these two types of language constructs can still be easily made if necessary (i.e., behave as atomic and composite ones as supposed).

A primary objective of composite pattern (Gamma et al., 1995) is to represent whole-part hierarchies and achieve the synergistic objective mentioned above. Figure 7 shows the implementation of composite pattern applied to PPCea interpreter, where *IStmt* is an abstract class that defines the interface and common behavior of both atomic (i.e., *Stmt*) and composite (i.e., *Series*) statements. The advantages of composite pattern mainly lie in the introduction of *IStmt* and the composition between *Series* and zero to more *IStmt* objects, which will be later identified as *Series* or *Stmt* concrete objects using polymorphism.

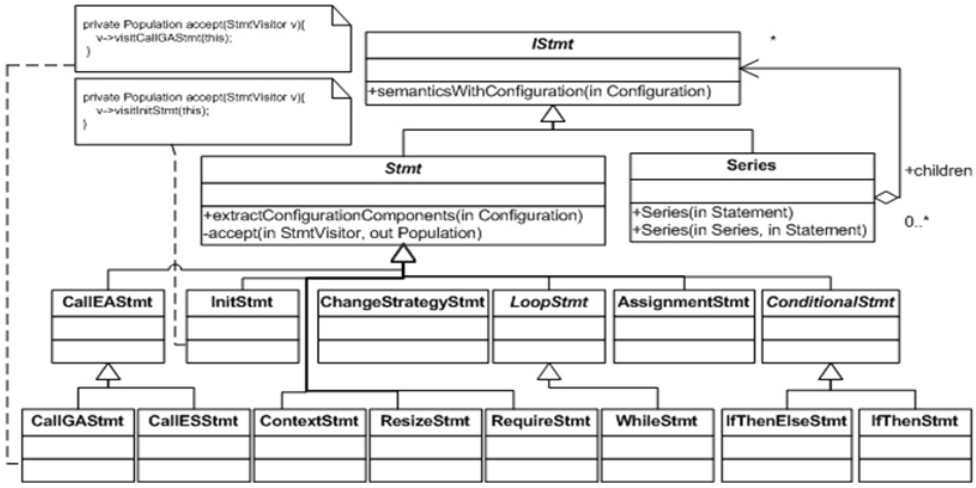


Fig. 7. Composite pattern applied to PPCea intepreter

Composite pattern also follows open-close principle (Meyer, 2000). Such a principle advocates "open for extension and close for modification": New statements can be added through inheritance; and modification on interface is closed and modification on implementation is isolated to associated methods only. For example, if one requests to introduce *callPSO* statement for Particle Swamp Optimization (Kennedy & Eberhart, 2001) in PPCea, three steps will be needed at the lexical, syntactical and semantic levels: (1) *callPSO* needs to be introduced as a token in PPCea.jlex (as seen in Figure 6); (2) A terminal that represents *callPSO* statement and the associated syntax are requested in PPCea.cup (can be found in Figure 6 too); and (3) Introduce a *CallPSOStmt* class inherited from *CallEAStmt*. Such a class defines the semantics/algorithms of Particle Swamp Optimization. A new operator that does not relate to any specific algorithm may be also introduced in the same manner. With such, introducing new algorithms or operators will not interfere with the remaining parts of PPCea. Extension and evolution of evolutionary algorithms and operators will be introduced next. Introducing evolutionary operators and parameters comprises the same steps as mentioned. For example, because *ResizeStmt* and *ChangeStrategyStmt* are two operators that can be applied to various EAs, they are not encapsulated into specific EA statements. For EA-specific operators, from the implementation's perspective, they can be introduced as standalone statements like r*esize* and *changeStrategy* or they can be encapsulated as methods in associated EA statement classes. For the sake of better design to satisfy high cohesion and responsibility driven concepts (Schach, 2010), such operators are encapsulated into EA associated statements.

3.3 Evolutionary Algorithm and operator extensions/evolutions using visitor pattern

Although composite pattern achieves the synergistic objective that allows uniformed treatments and making distinctions on atomic and composite statements when needed, extending or evolving methods encapsulated in EA statements is difficult. (Ironically, they are resulted from following good design principles as mentioned in the previous section). For example, *semanticWithConguration* in *CallEAStmt* is derived from *IStmt* that defines the semantics of a statement by executing a set of evolutionary operators (e.g., mutation, crossover, and selection). If a new operator, e.g., elite or n-point mutation, is introduced in *CallEAStmt* and invoked by *semanticWithConguration*, all *CallEAStmt*'s subclasses will be affected and recompilation is requested. Additionally, any evolution change to the existing algorithms and operators may be scattered around the entire class, which could be error-prone and resulted in regression faults. Because visitor pattern (Gamma et al., 1995) has succeeded in solving such tree-related problems with composite pattern (e.g., Wu et al., 2005), PPCea adopts visitor pattern so that the aforementioned extension and evolution problem can be solved.

As shown in Figure 8, PPCea introduces a super class, called *StmtVisitor*, which comprises two subclasses: *EAStmtVisitor* and *GenericStmtVisitor*, where the former one is to define the semantics of EA-specific operators and the latter one is to define the semantics of generic or non-EA-specific operators. For EA-specific operators, three important evolutionary operators (*selection*, *crossover*, and *mutation*) and two utility operators (*eval* and *getStat*) are introduced as *EAStmtVisitor*'s subclasses, where *eval* is to compute the fitness value of each individual and *getStat* is to compute the statistical data of an EA that are shown at the bottom of Figure 6. For generic operators, population initialization (*init*), population resizing

(*resize*), strategy changing (*changeStrategy*) for adapting different evolutionary operators on-the-fly, are introduced. Note that *Context* class introduced in Figure 8 is initialized by *ContextStmt* in Figure 7 that will store the current statement (or EA operator) that PPCea interpreter is executing and the resultant population. The usage of this class will be explained in more details in the next subsection.

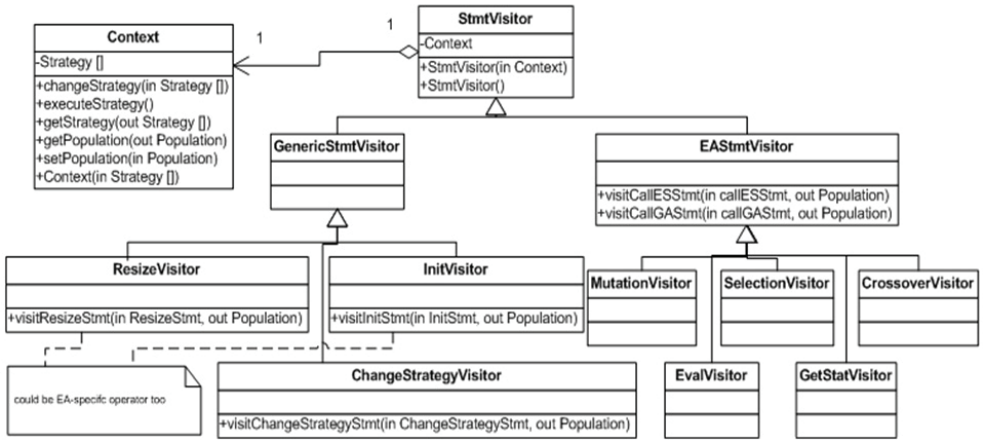


Fig. 8. Visitor pattern applied to PPCea

Readers who are not familiar with design patterns may find it difficult to see how composite and visitor patterns work together to tackle introduction, extension, and evolution problems by decoupling the syntax tree structure and operators within each statement, while at the same time allow the semantics of each statement as well as the entire program to be functioned correctly. To explain such correlation, the first step is to understand how CUP works with associated semantics written in Java classes. Because CUP is a compiler generator that generates a bottom-up parser, when each non-terminal is traversed, the corresponding Java class is instantiated. All the necessary classes (statements, expressions, and operators as seen in Figure 6) that define associated semantics will be available after the root of the parse tree is traversed. Then the root node will trigger semantics of each line of program to be interpreted and executed. When a statement, called *init*, is reached, the *semanticWithConfiguration* method within such a class will invoke a private method, called *accept* with an object of *InitVisitor* class passed in (see Figure 7). Within the *accept* method, *visitInitStmt* method of *InitVisitor* class will perform node/statement identification, called double dispatch (Gamma et al., 1995). If the current statement to be executed is *init*, *visitInitStmt* will execute the semantics of population initialization accordingly. Similarly, if *callGA* (or *callES*) is executed, objects of EvalVisitor, SelectionVisitor, MutationVisitor, CrossoverVisitor, and GetStatVisitor will be passed as parameters of *accept* method. The semantics defined in each visitor subclass will be executed after *callGA* statement is identified by double dispatch.

Important advantages that visitor pattern is capable of attacking extension and evolution problem can be expressed as the following three examples:

1. If a new EA-related operator is requested (e.g., *clone* or *repair*), we do not introduce an operator in *CallEAStmt*, *CallGAStmt* or *CallESStmt*, which will result in recompilation of almost the entire composite class hierarchy of Figure 7, as mentioned in the previous subsection. Instead, a subclass of *EAStmtVisitor* that defines the semantics of the newly introduced operator can be extended/introduced without affecting the remaining part of PPCea. There is no need to revise any part of JFlex and CUP files of PPCea either;
2. If a new generic operator is needed (e.g., *randomize* that introduces new random individuals into current population), a subclass that defines such an operator can be inherited from *GenericStmtVisitor* without editing other parts. If the new operator is also requested to be added to PPCea grammar, there is a need to introduce an associated token, syntax, and a subclass of *Stmt* respectively at the lexical, syntactical and semantic levels. Note, however, such extensions will still not interfere other parts of the existing code; and
3. If an existing EA-specific or generic operator is requested to be changed (i.e., evolution), the focus will be only on the specific subclass. Other parts will not be emphasized so the opportunities of regression faults will be minimized.

Although utilizing composite and visitor patterns to solve tree structure problems is not new, our implementation slightly varies the traditional solution and results in an additional advantage that can be observed in Figure 8: Even though introducing an EA at the statement level (e.g., *callGA*) may give readers impression that it is inflexible to control lower level evolutionary operators. Instead, it is a wrong impression! The visitor pattern utilized in PPCea is a variant – it is implemented along with strategy pattern (Gamma et al., 1995), where different evolutionary operators can be controlled through *changeStrategy* at the granularity of operator rather than algorithm. Such implementation avoids possible frequent recompilation while allowing ease of extension and evolution. Namely, only when a new EA, for example, *callPSO,* is introduced, existing subclasses of *EAStmtVisitor* need to add and compile a new operator, called *visitCallPSO.* More discussions will be covered in Section 3.5.

3.4 Strategic operator adaptation

Section 2.4 introduced three categories of operator adaptation: (1) Adaptation delegated to parameters (i.e., parameter control); (2) Adaptation among different types of operators; and (3) Adaptation among same types of operators. Strategy pattern (Gamma et al., 1995) is applied and integrated with visitor pattern to realize categories (2) and (3).

A primary objective of strategy pattern is to introduce a set of functionalities that can be interchanged upon request. Different kinds of evolutionary operators for GAs and ESs are introduced as subclasses of *Strategy* class in Figures 9 and 10, respectively. For example, we have implemented linear, non-linear, ranking, tournament and roulette wheel selections as subclasses of *GASelectionStrategy*. Similarly, the implementation of one-point and n-point mutation/crossover is defined in the associated subclasses of *GAMutationStrategy*/*GACrossoverStrategy*. If there is more than one way to initialize population, subclasses that specify such differences can be inherited from *GAInitStrategy*.

Extension and evolution to different algorithms of an operator can be also easily done by introducing subclasses. For example, fitness proportional selection may be introduced as a subclass of *GASelectionStrategy* without interfering the remaining parts of the source code;

and revising existing strategies will be isolated in their own classes – Again, this design follows open-close principle.

Figure 5 in Section 2.4 has presented how to adapt operators using PPCea. The dynamics of such code snippet with respect to strategy pattern is summarized as follows. Figure 10 is a simplified version of Figure 5 for ease of reading. Line 1 sets the default operators for selection, mutation, and crossover into context. A *Context* object will be instantiated with the *Strategy* objects of linear selection, 1-point mutation and n-point crossover as parameters.

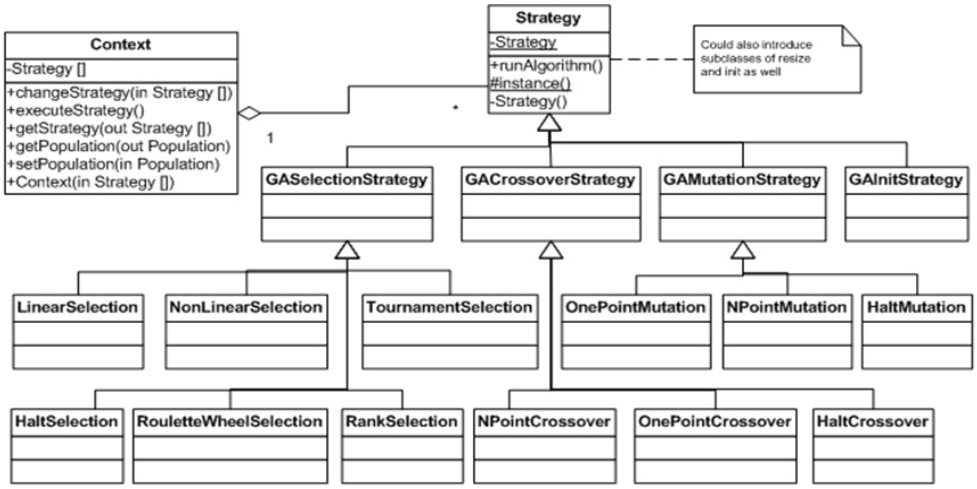


Fig. 9. Strategy pattern applied to GAs in PPCea

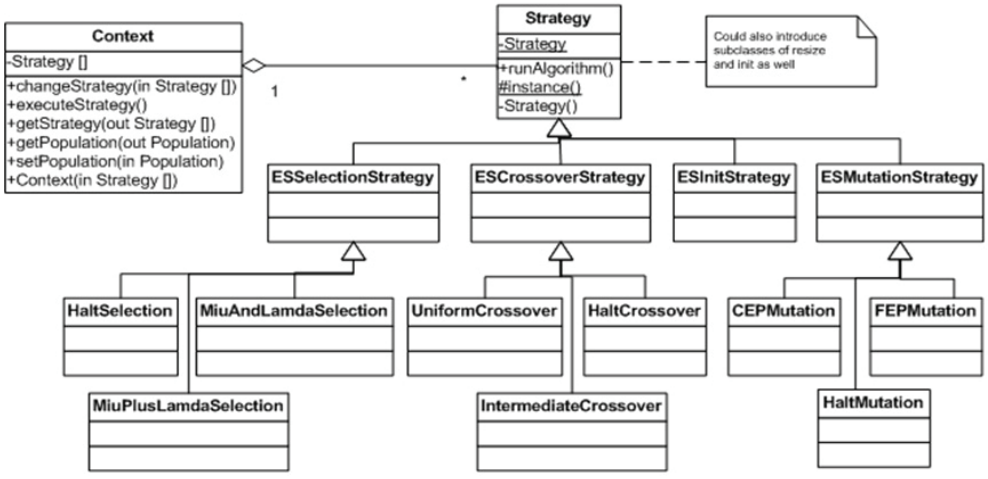


Fig. 10. Strategy pattern applied to ESs in PPCea

Line 2 initializes a population for the GA. Then *callGA* performs the GA using such strategies. Namely, *CallGA* statement in Figure 7 will invoke its *semanticWithConfiguration* method that accepts objects of *SelectionVisitor*, *MutationVisitor*, *CrossoverVisitor*, *EvalVisitor*, and *GetStatVisitor* shown in Figure 8. Each visitor object accesses the *Context* object and executes the associated *Strategy* object within *visitCallGAStmt* method. For example, the object of *SelectionVisitor* by default will execute an object of *LinearSelection* set in Line 1 of Figure 11. Between two *if*-statements, by using *changeStrategy* selection operator switches between tournament and rank selections every 10 generations. Additionally, mutation also switches between 1-point and n-point mutations every 10 generations. Crossover, on the other hand, will remain during the entire evolutionary process. Such behaviour is performed based on the following steps: (1) *ChangeStrategy* statement accepts an object of *ChangeStrategyVisitor*, which allows new *Strategy* objects (e.g., rank selection and n-point mutation at generation 10) to interchange with the existing *Strategy* object (e.g., linear selection and 1-point mutation). After exchange, *Context* object will execute new strategies. Note that, to avoid class explosion, Singleton pattern (Gamma et al., 1995) is applied to force each operator only has one associated object instantiated all the time. So that when the strategy pairs (tournament selection and n-point mutation) and (rank selection and 1-point mutation) are swapped every 10 generations, there is no new object instantiated, but the existing ones are reused. Halting an operator temporarily is also a feasible solution by replacing the current *Strategy* object to *HaltSelection*, *HaltMutation*, or *HaltCrossover*, which simply choose not to execute the current strategies. With such, EAs categorized as uni-process approaches in (Liu et al., 2009) can be reproduced using PPCea (e.g., DGEA in Figure 4).

```
1 context(LINEAR_SELECTION, ONE_PT_MUTATION, N_PT_CROSSOVER);
  //... skip some code
2 callGA;
3 if (( t % 10) == 0) then
4     changeStrategy(TOURNAMENT_SELECTION, N_PT_MUTATION) then
5 fi;
6 if (( t % 20) == 0) then
7     changeStrategy(RANK_SELECTION, ONE_PT_MUTATION) then
8 fi;
  //...skip some code
```

Fig. 11. A simplified version of Figure 5.

3.5 Parameter extension and evolution

Domain-specific parameters are those predefined in a DSL grammar and may facilitate productivity and other advantages of DSLs mentioned before. In PPCea, domain-specific parameters, shown in Table 2, can be categorized in two groups: (1) Parameters that are used for controlling an evolutionary process; and (2) Parameters that are computed at the end of each generation and may be treated as feedback to adjust an evolutionary process.

As seen in Table 2, parameters in group (1) are quite diverse. Some may be accessed across the entire project (e.g., *Popsize*, *Maxgen*, and *Epoch*) and others may be used by specific operators (e.g., *Alpha*, *Beta*, *Miu*, and *Lamda*). From the perspective of compiler/interpreter implementation, this kind of parameters usually already has identities stored in a symbol table (i.e., predefined). When such parameters are initialized by assignment statement, their values are stored in the symbol table accordingly. Whenever and wherever needed, the

values can be accessed through the symbol table. Extension of parameters falling in group (1) usually means introducing new domain-specific notations at the lexical, syntactical and semantic levels, which therefore has the same way of implementing domain-specific statements. Conversely, evolution for such a kind of parameters is usually renaming and

Group	Parameter Name	Description
(1)	Function	Fitness function to be evaluated (Obtained from (Yao et al., 1999))
	Popsize	Number of individuals of a population
	Maxgen	Maximum number of generation for an evolutionary process
	Epoch	Generation stride for parameter control adaptation
	pm	Mutation rate
	pc	Crossover rate
	psr	Stochastic ranking rate
	Alpha	Selection pressure (α) for linear/nonlinear selection
	Beta	Selection pressure (β) for linear/nonlinear selection
	Miu	Selection parameter (μ) for ESs
	Lambda	Selection parameter (λ) for ESs
	TourQ	Selection parameter for tournament selection
	KMeans	Number of centroids for clustering entropy
(2)	Best	Best fitness value of all individuals
	Average	Average fitness value of all individuals
	Worst	Worst fitness value of all individuals
	RatioM	Success mutation rate
	RatioC	Success crossover rate
	Stdv	Standard deviation of all individuals
	Euclidean	Euclidean distance of all individuals
	LinearEntropy	Linear Entropy (Liu et al., 2009)
	RoscaEntropy	Rosca Entropy (Liu et al., 2009)
	GaussianEntropy	Gaussian Entropy (Liu et al., 2009)
	FitProEntropy	Fitness Proportional Entropy (Liu et al., 2009)
	ClusterEntropy	Clustering Entropy (Liu et al., 2009)

Table 2. Current domain-specific parameters introduced in PPCea

changing valid scope based on our experience. Hence, refactoring (Fowler, 1999) may be applied to the lexical, syntactical and semantic levels to tackle evolution. Parameters in group (2) are computational results analyzed from population either after every (several) generation(s) or at the end of an entire evolutionary process. However, not all of such parameters are needed all the time. For example, for parameter tuning approaches, readers may be interested in fitness-related parameters only. Similarly, for non-entropy-driven approaches, one may avoid the computation of the five entropies shown in Table 2 and hence improve the performance. To achieve such a "pay-as-you-go" objective, decorator pattern is applied. As seen in Figure 12, *Parameter* class is introduced as a super class that applies singleton pattern to avoid more than one instance instantiated during an evolutionary process. *Decoratee* class is a "decoratee" super class, which means that all the objects of the subclasses inherited from *Decoratee* are mandatory to be provided by PPCea interpreter. Conversely, *DecoratorParameter* represents a super class whose subclass objects can be optionally computed upon requests.

The dynamics of how "pay-as-you-go" is achieved may be further observed by the code snippet shown in Figure 13: *visitCallGAStmt* is a method defined within *GetStatVisitor* class, accepted by *CallGAStmt* as described before, whose purpose is to analyze statistical results of a population. Lines 3 to 5 specify the mandatory domain-specific parameters to be computed. From lines 6 to 20, users may determine if any object of *Decorator* requires computation or not either directly defined in PPCea code (e.g., `require(LINEAR_ENTROPY, FITPRO_ENTROPY, ROSCA_ENTROPY);`) or through the graphical user interface we developed for PPCea interpreter. For example, if linear, fitness proportional and Rosca entropies are

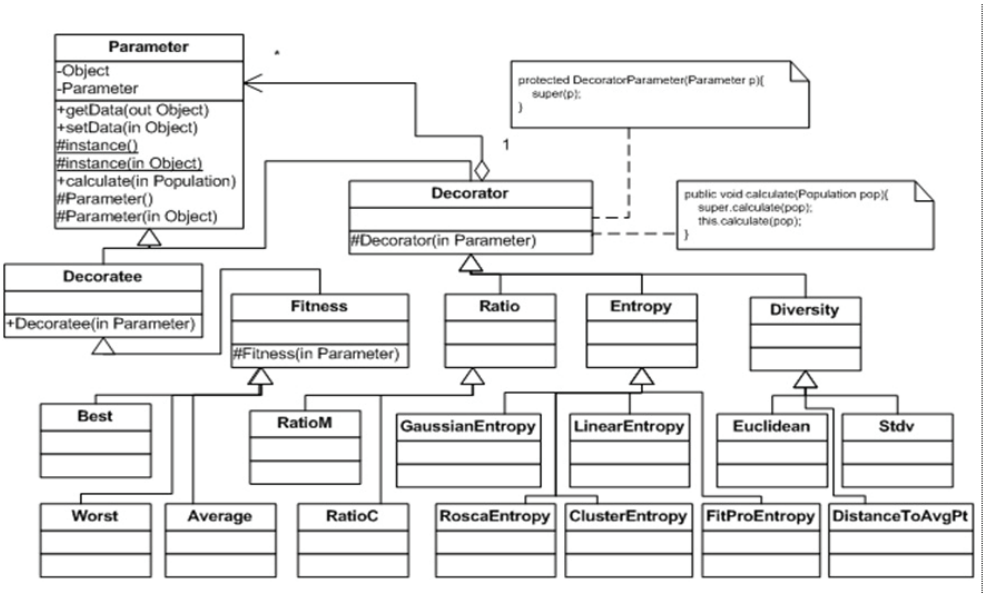


Fig. 12. Decorator pattern applied to PPCea.

selected by users (as seen above), when line 21 of Figure 13 is invoked, the *calculate* methods in *Best*, *Worst*, *Average*, *LinearEntropy*, *FitProEntropy*, and *RoscaEntropy* will be executed in a cascading and sequential order (see the notes in Figure 12).

How does decorator pattern address the extension and evolution problems of parameters in group (2)? For extension, if a new parameter is mandatory, a subclass should be inherited from *Decoratee* class. Conversely, if users are in charge of the computation necessities of newly introduced parameters, subclasses of *Decorator* will be introduced. The computational algorithms of the new parameters will be defined in their own *calculate* methods, along with an invocation to its super class' *calculate* method. One drawback of applying Decorator pattern is that when extension (i.e., introducing a new domain-specific parameter of group (2)) occurs, inevitably Figure 13 needs to be revised to incorporate such a change. It is because the purpose of the *if*-statements is to retrieve the answer of optional parameters that users determine to include. As for evolution, if any computational algorithm of a parameter changes, it is isolated in the associated *calculate* method.

```
1 Individual[] visitCallGAStmt(CallGAStmt gaStmt){
2   /* skip the code of retrieving parameters from symbol table ...*/
3   Parameter decorator = new Best(); //required parameter
4   decorator = new Worst(decorator); //required parameter
5   decorator = new Average (decorator); //required parameter
6   if (Linear_Entropy == 1) {
7       decorator = new LinearEntropy(decorator); //optional parameter
8   }
9   if (Gaussian_Entropy == 1) {
10      decorator = new GaussianEntropy(decorator); //optional parameter
11  }
12  if (FitPro_Entropy == 1) {
13      decorator = new FitProEntropy(decorator); //optional parameter
14  }
15  if (Rosca_Entropy == 1) {
16      decorator = new RoscaEntropy(decorator); //optional parameter
17  }
18  if (Cluster_Entropy == 1) {
19      decorator = new ClusterEntropy(decorator); //optional parameter
20  }
21  decorator.calculate(population);
22  return population;
23 }
```

Fig. 13. Decorator pattern applied to PPCea.

3.6 Software metrics

Software metrics (Lincke et al., 2008) are measures that assist in providing comprehensibility of software being assessed. Therefore, the quality of the software can be observed through such metrics. PPCea utilizes Eclipse Metrics plug-in (Sauer, 2010) to compare the current version implemented with DSL and design patterns and the original version introduced in (Liu et al., 2004), shown in Figure 14.

As seen in the figure, PPCea using patterns has much higher design and implementation quality in terms of Method lines of code, McCabe cyclomatic complexity (suggested maximum value: 10), and weighted methods per class. All other metrics listed in Figure 14 surpass the suggested maximum values. Namely, the design and implementation of PPCea with patterns is in good quality and refactoring may not be necessary.

Metric	Total	Mean	Std. Dev.	Maximum
Number of Overridden Methods	0			
Number of Attributes	1			
Number of Children	0			
Method Lines of Code (avg/max per method)	938	67	122.46	476
Number of Methods	14			
Nested Block Depth (avg/max per method)		2.857	1.457	6
Depth of Inheritance Tree	2			
McCabe Cyclomatic Complexity (avg/max per		13.643	24.04	94
Number of Parameters (avg/max per method)		1.786	1.081	4
Lack of Cohesion of Methods	0			
Number of Static Methods	0			
Specialization Index	0			
Weighted methods per Class	191			
Number of Static Attributes	0			

(a) Metrics of callGA in old PPCea

Metric	Total	Mean	Std. Dev.	Maximum
Number of Overridden Methods	0			
Number of Attributes	1			
Number of Children	0			
Method Lines of Code (avg/max per method)	92	13.143	13.357	42
Number of Methods	7			
Nested Block Depth (avg/max per method)		2	0.926	4
Depth of Inheritance Tree	2			
McCabe Cyclomatic Complexity (avg/max per		2.857	1.641	6
Number of Parameters (avg/max per method)		1	0.535	2
Lack of Cohesion of Methods	0			
Number of Static Methods	0			
Specialization Index	0			
Weighted methods per Class	20			
Number of Static Attributes	0			

(b) Metrics of callGA in new PPCea

Metric	Total	Mean	Std. Dev.	Maximum
Number of Overridden Methods	0			
Number of Attributes	0			
Number of Children	0			
Method Lines of Code (avg/max per method)	652	54.333	63.535	239
Number of Methods	12			
Nested Block Depth (avg/max per method)		2.75	1.164	4
Depth of Inheritance Tree	2			
McCabe Cyclomatic Complexity (avg/max per		11.833	13.662	46
Number of Parameters (avg/max per method)		1.917	1.256	4
Lack of Cohesion of Methods	0			
Number of Static Methods	0			
Specialization Index	0			
Weighted methods per Class	142			
Number of Static Attributes	0			

(c) Metrics of callES in old PPCea

Metric	Total	Mean	Std. Dev.	Maximum
Number of Overridden Methods	0			
Number of Attributes	0			
Number of Children	0			
Method Lines of Code (avg/max per method)	65	13	15.258	42
Number of Methods	5			
Nested Block Depth (avg/max per method)		1.6	0.49	2
Depth of Inheritance Tree	2			
McCabe Cyclomatic Complexity (avg/max per		2.4	1.2	4
Number of Parameters (avg/max per method)		0.8	0.4	1
Lack of Cohesion of Methods	0			
Number of Static Methods	0			
Specialization Index	0			
Weighted methods per Class	12			
Number of Static Attributes	0			

(d) Metrics of callES in new PPCea

Fig. 14. Metrics comparison between old and new PPCea

3.7 Potentials of PPCea

The first five subsections discuss how to overcome introduction, extension and evolution problems in four specific levels: (1) Evolutionary algorithms; (2) Evolutionary and generic operators; (3) Functionalities (i.e., strategies) of an operator; and (3) Domain-specific parameters. For (1), composite pattern facilitates the introduction/extension of evolutionary algorithms by introducing EAs as domain-specific statements, subclasses inherited from *IStmt*. For (2), visitor pattern promotes introduction/extension of evolutionary and generic operators by introducing subclasses of *StmtVisitor*. If evolutionary algorithms or its operators evolve, the changes will be isolated in the subclasses of *StmtVisitor* (or associated strategies), because such subclasses define the semantics of PPCea statements. Additionally, the decision of not introducing EA-specific operators at the PPCea statement level reduces the possibility of frequent changes/recompilation of the *StmtVisitor* and its subclasses. For (3), strategy pattern assists introduction/extension of different algorithmic strategies of an operator by introducing subclasses of *Strategy*. Evolution of such strategies is also isolated in associated classes. Also, PPCea is capable of adapting with operators on-the-fly under the support of strategy pattern. Lastly, decorator pattern addresses the problem of introduction/extension and evolution of domain-specific parameters by introducing subclasses of *Decorator* and *Decoratee*. Users are also allowed to determine which parameters to be analyzed so that unnecessary computation cost can be reduced.

To illustrate how a new EA can be introduced or an existing EA can be reproduced, let us use GAVaPS as an example. The algorithm of GAVaPS is as follows.

```
begin
  t=O
  initialize P(t)
  evaluate P(t)
  while (not termination-condition) do
    begin
    t = t + l
    increase the age of each individual by 1
    recombine P(t)
    evaluate P(t)
    remove from P(t) all individuals with age greater than the lifetime
  end
end
```

Fig. 15. The GAVaPS algorithm from (Arabas et al., 1994)

Based on (Arabas et al., 1994), *recombine* in Figure 15 performs normal mutation and crossover and then selection chooses offspring from all individuals with equal opportunity. As for *remove*, it will kill all individuals older than a predefined *lifetime* threshold. The population will be then resized based on the formula defined in the paper. To realize GAVaPS using PPCea, age attribute needs to be introduced in *Individual* class. Also, *lifetime* parameter may be introduced as a parameter in group (1). With such, users can adjust the value of *lifetime* by PPCea code. Then *GAVaPSSelection*, inherited from *GASelectionStrategy* in Figure 9, implements selection mechanism with equal opportunity and increments age if needed. Because there is more than one *resize* algorithm, a *ResizeStrategy* subclass may be inherited from *Strategy*. Then *ResizeByAge* may be introduced as a subclass of *ResizeStrategy* that kills all overage individuals and randomly introduces the number of new individuals using the formula if needed. Figure 16 is a pseudo PPCea code to simulate Figure 15 under the assumption all necessary subclasses are introduced in PPCea. Another possible implementation option is to introduce an entire new PPCea statement, called *callGAVaPS*. Nothing is really different except that *callGAVaPS* encapsulates all needed *Visitor* objects, which invoke objects of *GAVaPSSelection, OnePointMutation, OnePointCrossover,* and *ResizeByAge*.

```
Lifetime := 5; // any individual older than 5 will be killed
context(GAVAPS_SELECTION, ONE_PT_MUTATION, ONE_PT_CROSSOVER,
RESIZE_BY_AGE);
init;
while ( g < Maxgen ) do
  callGA;
  Popsize := ... // ... means the formulae from Arabas et al. 94
  resize( Popsize );
  g := g + 1
end
```

Fig. 16. The pseudo PPCea code that reproduces GAVaPS

In summary, PPCea utilizes DSL patterns and five design patterns so that the introduction/extension and evolution problems at the algorithm, operator, strategy, and parameter levels can be respectively addressed.

4. Related work

Evolving Objects (Keijzer et al., 2002) is an evolutionary computation framework that constructs an EA through component composition. Namely, each EA operator/statement is considered as a component and users need to select which specific components (similar to strategies) to be filled in to a specific spot of an evolutionary process. User defined parameters can be introduced to a file, which will be interpreted by the framework. ECJ (Luke et al., 2010) is a Java-based evolutionary computation system that requests users to describe an EA in Java by reusing/invoking a great number of packages for different EA operators. A set of predefined parameters with fixed identities also need to be defined in a specific file, acting like domain-specific parameters in PPCea. ESDL (Dower & Woodward, 2010) is a DSL that introduces SQL-like syntax for users to construct EAs. Name conventions for both EA operators and parameters need to be followed, which is same as PPCea. Because of interoperability advantage of XML, Veenhuis et al. introduced EAML (Veenhuis et al., 2000), a modeling language that utilizes XML to represent an EA. With such, different EA framework/software may introduce their own evolutionary processes by interpreting EAML files. There are also many EA framework or software that the book chapter is not able to fully cover. We leave this part to interested readers.

5. Conclusion

Controlling parameter settings to reach optimization and/or convergence of an EA has been a challenging topic in the evolutionary computation community. Firstly, due to meat-heuristic and stochastic nature, there is a need to conduct a sufficient number of experiments of an EA under different parameter settings. Additionally, many practitioners and scholars have put forth various algorithms, operators, and parameters to improve the optimization and/or convergence. Without automatic tools for EA users, conducting EA experiments would become tedious and error-prone. Without capabilities to extend and evolve automatic tools, EA developers would not be able to invent new algorithms, operators, strategies, and parameters. PPCea offers a synergistic solution to address the aforementioned problems from the perspectives of both users and developers. The contributions of PPCea are three-folds: (1) PPCea is an automatic EA tool in a language format that assists EA users to conduct experiments using three parameter setting approaches introduced by Eiben et al.; (2) PPCea is an open-ended EA tool that allows EA developers to introduce, extend and evolve EA constructs in algorithm, operator, strategy and parameter levels; and (3) PPCea offers a fair platform to perform EA comparison – both reproduced algorithms and new algorithms can be described in PPCea code and run under PPCea interpreter.

We have identified several future directions: (1) Multi-populations and multi-objective EAs are missing in current version. With such, more EAs can be reproduced (e.g., parameter-less GA) and invented; (2) As can be seen in Figure 6, PPCea currently cannot handle self adaptation algorithms. How to represent such algorithms and still offer open-end solutions is an emerging issue to tackle; (3) With the metrics from (Črepinšek et al., in press) introduced to PPCea, explicit balance between evolutionary and exploitation in a programmable fashion can be foreseen; and (4) Existing algorithms, operators, and strategies are effective in the individual granularity. Similar algorithms, operators, and

strategies working at the genotypical level may result in finer-grained experiments. With the aforementioned issues resolved, more EA users and developers may be benefited by PPCea.

Appendix: PPCea Grammar

```
Program          -> genetic Series end genetic
Series           -> Series Statement ; Statement
Statement        -> if Cond then Series fi | if Cond then Series else Series fi |
                    while Cond do Series end | ID := E |
                    init | callGA | callES | resize( N ) |
                    context ( Strategies ) | changeStrategy( Strategies ) |
                    require ( Parameters ) |
                    read Ids | write Ids |
                    readfile FileName | writeresult |
                    validateMsg | invalidateMsg

Condition        -> (Expression RelationOp Expression)
Expression       -> Expression Operator Expression | (Expression) |
                    sqrt(Expression) | exp(Expression) | Number | Double | ID |
Operator         -> + | - | * | / | && | ||
RelationOp       -> < | > | <= | >= | != | ==
Strategies       -> Strategies , Strategy
Strategy         -> Selection | Mutation | Crossover
Selection        -> LINEAR_SELECTION | NON_LINEAR_SELECTION |
                    TOURNAMENT_SELECTION | RANK_SELECTION |
                    ROULETTEWHEEL_SELECTION | GA_HALT_SELECTION |
                    MIU_AND_LAMDA_SELECTION | MIU_PLUS_LAMDA_SELECTION |
                    ES_HALT_SELECTION
Mutation         -> ONE_PT_MUTATATION | TWO_PT_MUTATION | N_PT_MUTATION |
                    GA_HALT_MUTATION | CEP_MUTATION | FEP_MUTATION |
                    ES_HALT_MUTATION
Crossover        -> ONE_PT_CROSSOVER | TWO_PT_CROSSOVER | N_PT_CROSSOVER |
                    GA_HALT_CROSSOVER | INTERMEDIATE_CROSSOVER |
                    UNIFORMCROSSOVER | ES_HALT_CROSSOVER
Parameters       -> Parameters , Parameter | ε
Parameter        -> RATIO_M | RATIO_C | STDV | EUCLIDEAN | LINEAR_ENTROPY |
                    ROSCA_ENTROPY | GAUSSIAN_ENTROPY | FITPRO_ENTROPY |
                    CLUSTER_ENTROPY
Number           -> integer
Double           -> double
FileName         -> string
```

6. References

Aho, A., Lam, M., Sethi, R. & Ullman J. (2007). *Compilers: Principles, Techniques, and Tools*, Addison Wesley.

Arabas, J., Michalewicz, Z. & Mulawka, J. (1994). GAVaPS - a Genetic Algorithm with Varying Population Size. *The 1st International Conference on Evolutionary Computation*, pp. 73-78

Bäck, T. & Schwefel, H. P. (1995). Evolution Strategies I: Variants and Their Computational Implementation. *Genetic Algorithms in Engineering and Computer Science*. John Wiley & Sons

Bäck, T. & Schütz, M. (1996). Intelligent Mutation Rate Control in Canonical Genetic Algorithms. *Foundations of Intelligent Systems*, pp. 158–167.

Bäck, T., Eiben, A. & van der Vaart., N. A. L. (2000). An Empirical Study on GAs Without Parameters. *In Parallel Problem Solving from Nature VI*, pp. 315–324.

Brooks, F. P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer*, 20(4):10–19.

Črepinšek, M., Mernik, M. & Liu, S.-H. (In Press). Analysis of Exploration and Exploitation in Evolutionary Algorithms by Ancestry Trees. *International Journal of Innovative Computing and Applications*.

Dower, S. & Woodward, C. (2010). Evolutionary System Definition Language. Swinburne University of Technology, Tech. Rep. TR/CIS/2010/1

Eiben, A. E., Hinterding, R. & Michalewicz, Z. (1999). Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2): 124-141.

Eiben, A. E., Marchiori, E. &Valkó, V. A. (2004). Evolutionary Algorithms with On-the-Fly Population Size Adjustment. *Parallel Problem Solving from Nature*, pp. 41-50.

Fogarty, T. C. (1989). Varying the Probability of Mutation in the Genetic Algorithm. *The 3rd International. Conference on Genetic Algorithms*, pp. 104-109

Fowler, M. (1999). Refactoring: Improving the Design of Existing Code. Addison Wesley.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design Patterns*, Addison-Wesley

Ghosh, A., Tsutsui, S. & Tanaka, H. (1996). Individual Aging in Genetic Algorithms. *Australian New Zealand Conference on Intelligent Information Systems*, pp. 276–279

Gray, J., Fisher, K., Consel, C., Karsai, G., Mernik, M. & Tolvanen, J.-P. (2008). DSLs: the Good, the Bad, and the Ugly. *The 23rd ACM Conference on Object-Oriented Programming Systems Languages and Applications*, pp. 791-794

Grefenstette, J. J. (1986). Optimization of Control Parameters for Genetic Algorithms. IEEE Transaction on Systems, Man & Cybernetics, SMC-16(1): 122-128

Harik, G. & Lobo, F. (1999). A Parameter-less Genetic Algorithm. Technical Report IlliGAL 9900, University of Illinois at Urban-Champaign

Hesser, J. & Männer, R. (1991). Toward an Optimal Mutation Probability for Genetic Algorithms. 1st Conference of Parallel Problem Solving from Nature, pp. 23–32.

Herrera, F. & Lozano, M. (1996). Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers. *Genetic Algorithms and Soft Computing*, pp. 95-125

Hudson, S. E. (2010). CUP LALR Parser Generator for Java. http://www2.cs.tum.edu/projects/cup/

Kang, K., Cohen, S., Hess, J., Novak, W. & Peterson., S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University

Keijzer, M., Merelo, J. J., Romero, G. & Schoenauer, M. (2002). Evolving Objects: A General Purpose Evolutionary Computation Library. *Artificial Evolution*, 2310: 829-888.

Kennedy, J. & Eberhart, R.C. (2001). *Swarm Intelligence*. Morgan Kaufmann

Klein, G. (2010). JFlex User's Manual. http://jflex.de/manual.html

Lincke, R., Lundberg, J. & Löwe, W. (2008). Comparing Software Metrics Tools. *International Symposium on Software Testing and Analysis*, pp. 131-142.

Liu, S.-H. (2010). PPCea Web Page. http://www.zimmer.csufresno.edu/~shliu/research/PPCea.html

Liu, S.-H., Mernik, M. & Bryant, B. R. (2004). Parameter Control in Evolutionary Algorithms by Domain-Specific Scripting Language PPCea. *The 1st International Conference on Bioinspired Optimization Methods and their Applications,* pp. 41-50.

Liu, S.-H., Mernik, M. & Bryant, B. R. (2009). To Explore or to Exploit: An Entropy-Driven Approach for Evolutionary Algorithms. *International Journal of Knowledge-based and Intelligent Engineering Systems 13(3-4): 185-206.*

Luke., S. et al. (2010). ECJ: A Java-based Evolutionary Computation Research System. http://cs.gmu.edu/~eclab/projects/ecj/

Meyer, B. (2000). Object-Oriented Software Construction. Prentice Hall.

Michalewicz, Z. (1996). Genetic Algorithm + Data Structures = Evolution Programs. Springer

Mernik, M., Heering, J. & Sloane, A. (2005). When and How to Develop Domain-Specific Languages, *ACM Computing Surveys*, 37(4): 316-344.

Sauer, F. (2010). Eclipse Metrics plug-in. http://sourceforge.net/projects/metrics/

Schach, S. (2010). Object-Oriented and Classical Software Engineering, McGraw Hill.

Smith, R. & Smuda, E. (1995). Adaptively Resizing Populations: An Algorithm, Analysis, and First Results. *Complex Systems*, 1(9): 47-72.

Storn, R. & Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous *Spaces. Journal of Global Optimization,* 11: 341–359.

Tsutsui, S., Ghosh, A., Corne, D. & Fujimoto, Y. (1997). A Real Coded Genetic Algorithm with an Explorer and an Exploiter Populations. *The 7th International Conference on Genetic Algorithms*, pp. 238–245.

Ursem, R. (2002). Diversity-Guided Evolutionary Algorithms. *Parallel Problem Solving from Nature* VII, LNCS 2439: pp. 462-471

Veenhuis, C., Franke, K. & Köppen, M. (2000). A Semantic Model for Evolutionary Computation. *6th International Conference on Soft Computing*.

Wu, X., Bryant, B. R., Gray, J. & Mernik, M. (2010). Component-Based LR Parsing. *Computer Languages, Systems, and Structures*, 36(1): 16-33.

Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary Programming Made Faster. *IEEE Transactions on Evolutionary Computation*, 3(2): 82-102.

11

Evolution Algorithms in Fuzzy Data Problems

Witold Kosiński [1] , Katarzyna Węgrzyn-Wolska [2] and Piotr Borzymek [3]
[1,3] *Polish-Japanese Institute of Information Technology, Warsaw, Poland*
[1] *Kazimierz-Wielki University, Bydgoszcz, Poland*
[2] *École Supérieure d'Ingénieurs en Informatique et Génie de Télécommunication (ESIGETEL), France*

1. Introduction

Genetic algorithms (GA) and their general class – evolutionary algorithms (EA) belong to a set of optimization methods that are nature inspired. In recent applications of computational intelligence tools very often we deal with situation when imprecise data appear. The data can be fuzzy. Hence, especially when an optimization of some object function has to be analyzed, the problem appears which model of fuzzy data should be used. In the literature two models of fuzzy numbers are mainly used: one is the classical model which follows from the Zadeh fuzzy set model (Zadeh, 1965), restricted, however, to convex membership functions defined on reals (Nguyen, 1978), and called convex fuzzy numbers (CFN), or the model with restricted forms of membership functions, called (L, R)–numbers (Dubois & Prade, 1978). The concept of convex fuzzy numbers has been introduced by Nguyen in 1978 in order to improve calculation and implementation properties of fuzzy numbers. However, the results of multiply operations on the convex fuzzy numbers are leading to the large grow of the fuzziness, and depend on the order of operations since the distributive law, which involves the interaction of addition and multiplication, does hold there. If one works with the second model of fuzzy numbers, (L, R)–numbers, approximations of fuzzy functions and operations are needed if one wants to follow the extension principle and stay within (L, R)–numbers (Dubois & Prade, 1978). As long as one works with fuzzy numbers that possess continuous membership functions the two procedures: the extension principle of Zadeh from 1975 and the α-cut and interval arithmetic method give the same results (Buckley & Eslami, 2005). They lead, however, to some drawbacks as well as to unexpected and uncontrollable results of repeatedly applied operations (Wagenknecht et.al., 2001). From this several drawbacks of convex fuzzy numbers and operations on them follow. One of them is non-existence of the solution of the most general and simple algebraic equation $A + X = C$, when A and C are quite arbitrary fuzzy numbers. In order to omit those drawbacks in 2002 the present author (W.K.) with two co-workers developed a generalization of the classical concept of fuzzy numbers and defined so-called ordered fuzzy numbers (OFN), in which membership function is not a primitive concept, but a pair of real-valued functions defined on the unit interval $[0, 1]$ (cf. (Kosiński et.al., 2002a; Kosiński et. al., 2003a;b)). Then all operations are natural defined on those pairs, as a space of functions. The arithmetics of ordered fuzzy numbers becomes an efficient tool in dealing with unprecise, fuzzy quantitative terms. Moreover, each convex fuzzy number is included in this class, moreover it defines two different OFN: they differ

by their orientations. The space of OFN is partially ordered, since a cone of positive fuzzy numbers may be defined.
When data set of an optimization problem are not accurate, imprecise or just fuzzy, EA methods may be difficult to apply. This is due to the fact that in the classical, Zadeh's theory (Zadeh, 1965; 1975) of fuzzy sets, the main object, namely fuzzy numbers, are not ordered. Moreover, algebraic operations defined on classical fuzzy numbers (i.e. convex of Nguyen or (L,R)-type of Dubois and Prade) which use either Zadeh's extension principle or the interval analysis on α-sections, do not have distributive property, which make the big problem when repeated operations are performed (Wagenknecht et.al., 2001; Wagenknecht, 2001).
Order fuzzy numbers (OFN) invented by the present author and his two co-workers in 2002-2003 in order to omit these and other drawbacks, make possible to deal with fuzzy inputs quantitatively, exactly in the same way as with real numbers. The space of OFN can give us a natural setup to deal with optimization problems when data are fuzzy. Moreover, new defuzzification functionals which attach to each fuzzy number a real, crisp, number, may be used to supply the search space with additional fitness measure and order relations. The case when fuzzy numbers are presented as pairs of step functions, with finite resolution, simplifies all operations as well as the representation of defuzzification functionals. This helps us to formulate a general optimization problem with fuzzy data.
In the paper we present model of OFN and show its application in formulation of optimization problem when date for the object function are fuzzy. Those fuzzy data are regarded as OFN. Then values of object function are fuzzy, as well. However, the space of OFN may be equipped with the lattice structure, and hence the question of maximization of fuzzy-valued fitness function may be solved. Some application will be given in the case, when we confine our interest to step functions, appearing in the representation of OFN. Then each fuzzy number can be identified with a point in 2K dimensional vector space, when K is the resolution parameter, which is responsible for the maximal number of steps each fuzzy number possesses. Then genetic algorithm can be formulated. The important role in dealing with fuzzy evolutionary(genetic) algorithms play defuzzification functionals, which map each OFN into reals. They should be homogeneous of order one and restrictive additive
The second problem considered in this chapter is related to the application of evolutionary algorithms in finding forms of linear and nonlinear defuzzification functionals, knowing their action on a subset of the space OFN. This forms a kind of approximation problem in which data are given as fuzzy numbers.

2. Fuzzy numbers

Fuzzy numbers (Zadeh, 1965) are very special fuzzy sets defined on the universe of all real numbers $\mathbb{R}$. In applications the so-called (L,R)–numbers proposed by Dubois and Prade (Dubois & Prade, 1978) as a restricted class of membership functions, are often in use. In most cases one assumes that membership function of a fuzzy number A satisfies convexity assumptions (Nguyen, 1978). However, even in the case of convex fuzzy numbers (CFN) multiply operations are leading to the large grow of the fuzziness, and depend on the order of operations.
This as well as other drawbacks have forced us to think about some generalization [1]. Our main observation made in (Kosiński et.al., 2002a) was: a kind of quasi-invertibility (or quasi-convexity (Martos, 1975)) of membership functions is crucial. Invertibility of membership functions of convex fuzzy number A makes it possible to define two functions

[1] A number of attempts to introduce non-standard operations on fuzzy numbers has been made (Drewniak, 2001; Klir, 1997; Sanschez, 1984; Wagenknecht, 2001)

a_1, a_2 on $[0,1]$ that give lower and upper bounds of each α-cut of the membership function μ_A of the number A

$$A[\alpha] = \{x : \mu_A(x) \geq \alpha\} = [a_1(\alpha), a_2(\alpha)] \text{ with } a_1(\alpha) = \mu_A|^{-1}_{incr}(\alpha) \text{ and } a_2(\alpha) = \mu_A|^{-1}_{decr}(\alpha)\,,$$

where $|_{incr}$ and $|_{decr}$ denote the restrictions of the function μ_A to its sub-domains on which is increasing or decreasing, respectively. Both functions $a_1(\alpha), a_2(\alpha)$ were used for the first time by the authors of (Goetschel & Voxman, 1986) in their parametric representation of fuzzy numbers, they also introduced a linear structure to convex fuzzy numbers.

2.1 Ordered fuzzy numbers

In the series of papers (Kosiński et.al., 2002a; Kosiński et. al., 2003a;b) we have introduced and then developed main concepts of the space of ordered fuzzy numbers (OFNs). In our approach the concept of membership functions has been weakened by requiring a mere *membership relation* .

Definition 1. *A pair (f,g) of continuous functions such that $f,g : [0,1]\to\mathbb{R}$ is called an ordered fuzzy number A.*

Notice that f and g need not be inverse functions of some membership function. If, however, f is increasing and g – decreasing, both on the unit interval I, and $f \leq g$, then one can attach to this pair a continuous function μ and regard it as a membership function a convex fuzzy number with an extra feature, namely the orientation of the number. This attachment can be done by the formula $f^{-1} = \mu|_{incr}$ and $g^{-1} = \mu|_{decr}$. Notice that pairs (f,g) and (g,f) represents two different ordered fuzzy numbers, unless $f = g$. They differ by their orientations.

Definition 2. *Let $A = (f_A, g_A), B = (f_B, g_B)$ and $C = (f_C, g_C)$ are mathematical objects called ordered fuzzy numbers. The sum $C = A + B$, subtraction $C = A - B$, product $C = A \cdot B$, and division $C = A \div B$ are defined by formula*

$$f_C(y) = f_A(y) \star f_B(y)\,, \qquad g_C(y) = g_A(y) \star g_B(y) \tag{1}$$

where "$\star$" works for "$+$", "$-$", "$\cdot$", and "$\div$", respectively, and where $A \div B$ is defined, if the functions $|f_B|$ and $|g_B|$ are bigger than zero.

Scalar multiplication by real $r \in \mathbb{R}$ is defined as $r \cdot A = (rf_A, rg_A)$. The subtraction of B is the same as the addition of the opposite of B, and consequently $B - B = 0$, where $0 \in \mathbb{R}$ is the crisp zero. It means that subtraction is not compatible with the the extension principle, if we confine OFNs to CFN. However, the addition operation is compatible, if its components have the same orientations. Notice, however, that addition, as well as subtraction, of two OFNs that are represented by affine functions and possess classical membership functions may lead to result which may not possess its membership functions (in general $f(1)$ needs not be less than $g(1)$).

A relation of partial ordering in the space $\mathcal{R}$ of all OFN, can be introduced by defining the subset of *positive* ordered fuzzy numbers: a number $A = (f,g)$ is not less than zero, and write

$$A \geq 0 \text{ if } f \geq 0,\, g \geq 0\,, \text{and } A \geq B \text{ if } A - B \geq 0\,. \tag{2}$$

In this way the space $\mathcal{R}$ becomes *a partially ordered ring* . Neutral element of addition in $\mathcal{R}$ is a pair of constant function equal to crisp zero.

Operations introduced in the space $\mathcal{R}$ of all ordered fuzzy numbers (OFN) make it an algebra, which can be equipped with a sup norm $||A|| = \max(\sup_{s\in I}|f_A(s)|, \sup_{s\in I}|g_A(s)|)$ if $A = (f_A, g_A)$. In $\mathcal{R}$ any algebraic equation $A + X = C$ for X, with arbitrarily given

fuzzy numbers A and C, can be solved. Moreover, $\mathcal{R}$ becomes a Banach space, isomorphic to a Cartesian product of $C(0,1)$ - the space of continuous functions on $[0,1]$. It is also a Banach algebra with unity: the multiplication has a neutral element - the pair of two constant functions equal to one, i.e. the crisp one.
Some interpretations of the concepts of OFN have been given in (Kosiński et.al., 2009a). Fuzzy implications within OFN are presented in (Kosiński et. al., 2009b).

Step functions

It is worthwhile to point out that a class of ordered fuzzy numbers (OFNs) represents the whole class of convex fuzzy numbers with continuous membership functions. To include all CFN some generalization of functions f and g in Def.1 is needed. This has been already done by the first author who in (Kosiński, 2006) assumed they are functions of bounded variation. Then operations are defined in the similar way, the norm, however, will change into the norm of the cartesian product of the space of functions of bounded variations (BV). Then all convex fuzzy numbers are contained in this new space $\mathcal{R}_{BV}$ of OFN. Notice that functions from BV (Łojasiewicz, 1973) are continuous except for a countable numbers of points.
Important consequence of this generalization is a possibility of introducing a subspace of OFN composed of pairs of step functions. If we fix a natural number K and split $[0,1)$ into $K-1$ subintervals $[a_i, a_{i+1})$, i.e. $\bigcup_{i=1}^{K-1}[a_i, a_{i+1}) = [0,1)$, where $0 = a_1 < a_2 < ... < a_K = 1$, and define a step function f of resolution K by putting u_i on each subinterval $[a_i, a_{i+1})$, then each such function f is identified with a K-dimensional vector $f \sim \boldsymbol{u} = (u_1, u_2 ... u_K) \in \mathbb{R}^K$, the K-th value u_K corresponds to $s = 1$, i.e. $f(1) = u_K$. Taking a pair of such functions we have an ordered fuzzy number from $\mathcal{R}_{BV}$. Now we introduce
Definition 3. By a step ordered fuzzy number A of resolution K we mean an ordered pair (f, g) of functions such that $f, g : [0,1] \to \mathbb{R}$ are K-step function.

We use $\mathcal{R}_K$ for denotation the set of elements satisfying Def. 3. The set $\mathcal{R}_K \subset \mathcal{R}_{BV}$ has been extensively elaborated by our students in (Gruszczyńska & Krejewska, 2008) and (Kościeński, 2010). We can identify $\mathcal{R}_K$ with the Cartesian product of $\mathbb{R}^K \times \mathbb{R}^K$ since each K-step function is represented by its K values. It is obvious that each element of the space $\mathcal{R}_K$ may be regarded as an approximation of elements from $\mathcal{R}_{BV}$, by increasing the number K of steps we are getting the better approximation. The norm of $\mathcal{R}_K$ is assumed to be the Euclidean one of $\mathbb{R}^{2K}$, then we have a inner-product structure for our disposal.

2.2 Defuzzification functionals

In the course of defuzzification operation in CFN to a membership function a real, crisp number is attached. We know a number of defuzzification procedures from the literature (Van Leekwijck & Kerre, 1999). Continuous, linear functionals on $\mathcal{R}$ give a class of defuzzification functionals . Each of them, say ϕ, has the representation by the sum of two Stieltjes integrals with respect to two functions h_1, h_2 of bounded variation,

$$\phi(f,g) = \int_0^1 f(s)dh_1(s) + \int_0^1 g(s)dh_2(s)\,. \tag{3}$$

Notice that if for $h_1(s)$ and $h_2(s)$ we put $\lambda H(s)$ and $(1-\lambda)H(s)$, respectively, with $0 \le \lambda \le 1$ and $H(s)$ as the Heaviside function with the unit jump at $s = 1$, then the defuzzification functional in (3) will lead to the classical MOM – middle of maximum, FOM (first of maximum), LOM (last of maximum) and RCOM (random choice of maximum), with an

appropriate choice of λ. For example if for $h_1(s)$ and $h_2(s)$ we put $1/2H(s)$ then the defuzzification functional in (3) will represent the classical MOM – middle of maximum

$$\phi(f,g) = 1/2(f(1) + g(1)) . \tag{4}$$

New model gives a continuum number of defuzzification operators both linear and nonlinear, which map ordered fuzzy numbers into reals. Nonlinear center of gravity defuzzification functional (COG) calculated at OFN (f,g) is

$$\bar{\phi}_G(f,g) = \int_0^1 \frac{f(s)+g(s)}{2}[f(s)-g(s)]ds\{\int_0^1 [f(s)-g(s)]ds\}^{-1} . \tag{5}$$

If $A = c^{\ddagger}$ then we put $\bar{\phi}_G(c^{\ddagger}) = c$. When $\int_0^1 [f(s)-g(s)]ds = 0$ in (5) a correction needs to be introduced. Here by writing $\bar{\phi}(c^{\ddagger})$ we understand the action of the functional $\bar{\phi}$ on the crisp number $c^{\ddagger}$ from $\mathbb{R}$, which is represented by a pair of constant functions $(c^{\dagger}, c^{\dagger})$, with $c^{\dagger}(s) = c, s \in [0,1]$. New model gives a continuum number of defuzzification operators both linear and nonlinear, which map ordered fuzzy numbers into reals. Nonlinear functional can be defined, see (Kosiński & Wilczyńska-Sztyma, 2010).
In our understanding a most general class of continuous defuzzification functionals ϕ should satisfy three conditions:

1. $\phi(c^{\ddagger}) = c$,
2. $\phi(A + c^{\ddagger}) = \phi(A) + c$,
3. $\phi(cA) = c\phi(A)$, for any $c \in \mathbb{R}$ and $A \in \mathcal{R}$.

Here by writing $\phi(c^{\ddagger})$ we understand the action of the functional ϕ on the crisp number $c^{\ddagger}$ from $\mathbb{R}$, which is represented, in the case of an element from $\mathcal{R}_K$, by a pair of constant functions $(c^{\dagger}, c^{\dagger})$, with $c^{\dagger}(i) = c, i = 1,2,...,K$. The condition 2. is a *restricted additivity*, since the second component is crisp number. The condition 3. requires from ϕ to be homogeneous of order one, while the condition 1. requires $\int_0^1 dh_1(s) + \int_0^1 dh_2(s) = 1$, in the representation (3).
On the space $\mathcal{R}_K$ a representation formula for a general non-linear defuzzification functional $H : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$ satisfying the conditions 1.– 3., can be given as a linear composition (Rudnicki, 2010) of arbitrary homogeneous of order one, continuous function G of $2K-1$ variables, with the 1D identity function, i.e.

$$H(\underline{u},\underline{v}) = u_1 + G(u_2-u_1, u_3-u_1, ..., u_K-u_1, v_1-u_1, v_2-u_1, ..., v_K-u_1) , \tag{6}$$

with

$$\underline{u} = (u_1, ..., u_K) , \underline{v} = (v_1, ..., v_K) .$$

Remark. It can be shown from this representation that a composition of arbitrary homogeneous of order one, continuous function F of k–variables, which is additionally restrictive additive, with a set of k defuzzification functionals $\varphi_1, \varphi_2, ..., \varphi_k$, leads to a new defuzzification functionals, i.e. $F \circ (\varphi_1, \varphi_2, ..., \varphi_k)$ on $\mathcal{R}$ (or on $\mathcal{R}_K$) is a new nonlinear (in general) defuzzification functional. Moreover, the function F may be written in the form of (6), in the case of $\mathcal{R}_K$. When the space $\mathcal{R}$ appears, we have to substitute its arguments with

$\varphi_1, \varphi_2, ..., \varphi_k$; in general case it will be:

$$F(\varphi_1, \varphi_2, ..., \varphi_k) = \varphi_j + G(\varphi_1 - \varphi_j, \varphi_2 - \varphi_j, ..., \varphi_k - \varphi_j)\,, \text{with some } 1 \leq j \leq k\,, \qquad (7)$$

where the function G is homogeneous of order one and depends on $k-1$ variables, since between its arguments the difference $\varphi_j - \varphi_j$ does not appear. In fact G is given by F in which its j-th argument was put equal to zero.
Due to the fact that $\mathcal{R}_K$ is isomorphic to $\mathbb{R}^K \times \mathbb{R}^K$ we conclude, from the Riesz theorem and the condition 1. that a general linear defuzzification functional on $\mathcal{R}_K$ has the representation

$$H(\underline{u}, \underline{v}) = \underline{u} \cdot \underline{b} + \underline{v} \cdot \underline{d}\,, \text{with arbitrary } \underline{b}\,, \underline{d} \in \mathbb{R}^K\,, \text{such that } \underline{1} \cdot \underline{b} + \underline{1} \cdot \underline{d} = 1\,, \qquad (8)$$

where $\cdot$ denotes the inner (scalar) product in $\mathbb{R}^K$ and $\underline{1} = (1, 1, ..., 1) \in \mathbb{R}^K$ is the unit vector in $\mathbb{R}^K$, while the pair $(\underline{1}, \underline{1})$ represents a crisp one in $\mathcal{R}_K$. It means that such functional is represented by the vector $(\underline{b}, \underline{d}) \in \mathbb{R}^{2K}$. Notice that functionals of the type $\phi_j = \underline{e}_j, j = 1, 2, ..., 2K$, where $\underline{e}_j \in \mathbb{R}^{2K}$ has all zero component except for 1 on the j-th position, form a basis of ${\mathcal{R}_K}^*$ - the space adjoint to $\mathcal{R}_K$, they are called *fundamental functionals* .
Notice that each real-valued function $\psi(z)$ of a real variable $z \in \mathbb{R}$ may be transformed to a fuzzy-valued function on $\mathcal{R}_{BV}$, and even simpler on $\mathcal{R}_K$. Since each OFN from $\mathcal{R}_K$ is a pair of two vectors, each from $\mathbb{R}^K$, say $(\underline{u}, \underline{v})$, the fuzzy counterpart of the function ψ at $(\underline{u}, \underline{v})$ will be a pair of vectors $(\psi(u_1), ..., \psi(u_K), \psi(v_1), ..., \psi(v_K))$, which [2] are in $\mathcal{R}_K$. Further on for these compositions we will use the denotation $\psi \circ \underline{u}$ and $\psi \circ \underline{v}$, or $\psi \circ (\underline{u}, \underline{v})$.

3. Optimization with fuzzy data

Let us assume that we face with an optimization problem on a set $\mathcal{D}$, a subset of the space $\mathbb{R}^{2K}$ and a fuzzy-valued fitness function $\Psi : \mathcal{D} \subset \mathbb{R}^{2K} \to \mathbb{R}^{2K}$ has been constructed from a real-valued one. The question is how to define an evolutionary algorithm for such problem?
Notice, that in case of fuzzy numbers $A = (\underline{u}_a, \underline{v}_a)$ and $B = (\underline{u}_b, \underline{v}_b)$ from $\mathcal{R}_K$ the relation (2) means componentwise inequality $u_{ai} - u_{bi} \geq 0$ and $v_{ai} - v_{bi} \geq 0$ for $i = 1, 2, ..., K$. This set of inequalities may be written in terms of inequalities between values of defuzzification functionals forming the basis of ${\mathcal{R}_K}^*$, namely $\phi_j(A) - \phi_j(B) \geq 0$ for $j = 1, 2, ..., 2K$.
Notice, that for each two fuzzy numbers A, B as above, we may define $\inf(A, B)$ and $\sup(A, B)$, both from $\mathcal{R}_K$, by the formula $\inf(A, B) = C =: (\underline{u}_c, \underline{v}_c)$, where each $u_{ci} := \min\{u_{ai}, u_{bi}\}$ and $v_{ci} := \min\{v_{ai}, v_{bi}\}$ with $i = 1, ..., K$. Similarly we define $\sup(A, B) = D =: (\underline{u}_d, \underline{v}_d)$.
It is evident that our definitions are in agreement with the relation (2), since $\inf(A, B) \leq A\,, \inf(A, B) \leq B$, and similarly $\sup(A, B) \geq A\,, \sup(A, B) \geq B$; moreover $A = \inf(A, B)$ in the case when $A \leq B$. Similar relation follows with $\sup(A, B)$. These definitions allow us to define a lattice structure on the space of $\mathcal{R}_K$. It will be the subject of the next paper.
We know that due to the order relation (2) for two ordered fuzzy numbers $A, B \in \mathcal{R}_K$ we may have: either $A \geq B$ or $A \leq B$, or we cannot say anything. Hence we should have for our disposal another, additional set of measures, which will give us a chance to compare any two different fuzzy values of the fitness function Ψ. We do this by introducing the next definition.
Definition 4. Let a set of defuzzification functionals (linear or nonlinear) $\Phi_1, ..., \Phi_L$ be given together with a fuzzy-valued fitness function $\Psi : \mathcal{D} \subset \mathbb{R}^{2K} \to \mathbb{R}^{2K}$. Let A, B be from $\mathcal{R}_K$. We say that $\Psi(A) \succ \Psi(B)$ if $\Phi_k(\Psi(A)) \geq \Phi_k(\Psi(B))$, for $k = 1, ..., L$.
Notice, that if $L = 2K$ and each Ψ is equal to $\phi_k \in {\mathcal{R}_K}^*$ the relation $\succ$ corresponds to $\geq$ from (2). We are rather interested in different ordering.

[2] Here we have used the representation for $\underline{u} = (u_1, ..., u_K)$ and for $\underline{v} = (v_1, ..., v_K)$.

However, if we use the convex combination of fundamental functionals given by the defuzzification functional H from (8) and superpose it with the fitness function Ψ, then a new real-valued fitness function $\widehat{\Psi}(\cdot) := H(\Psi(\cdot)) : \mathcal{D} \to \mathbb{R}$ may be defined, and use in further evolutionary computation.
Finally we propose some genetic operators acting on arguments of the fitness function Φ. Let two individuals $A = (\underline{u}_a, \underline{v}_a)$ and $B = (\underline{u}_b, \underline{v}_b)$ be given. We may define a one (or many-point) cross-over operator as an exchange at some position (positions) a part of components of two vectors from $\mathbb{R}^{2K}$. Another operator could be a two-point mutation when after selection of two positions $1 \leq j_1, j_2 \leq K$ the corresponding components of vectors $\underline{u}_a$ and $\underline{v}_a$ have to be exchanged. It may be added that using different denotation for individuals, say A, as a K-dimensional vector of pairs $((u_{a1}, v_{a1}), ..., (u_{aK}, v_{aK}))$, next genetic operations can be easily defined. It will be the subject of the next paper, when a numerical implementation will be performed.

4. Approximation of defuzzification functionals

Ultimate goal of fuzzy logic is to provide foundations for approximate reasoning. It uses imprecise propositions based on a fuzzy set theory developed by L.Zadeh, in a way similar to the classical reasoning using precise propositions based on the classical set theory. Defuzzification is the main operation which appears in fuzzy controllers and fuzzy inference systems where fuzzy rules are present. In the course of this operation to a membership function representing a classical fuzzy set a real number is attached. We know a number of defuzzification procedures from the literature, such as: FOM (first of maximum), LOM (last of maximum), MOM (middle of maximum), RCOM (random choice of maximum), COG (center of gravity), and others which were extensively discussed by the authors of (Van Leekwijck & Kerre, 1999). They have classified the most widely used defuzzification techniques into different groups, and examined the prototypes of each group with respect to the defuzzification criteria.
The problem arises when membership functions are not continuous or do not exist at all. The present chapter is devoted to a particular subsets of fuzzy sets, namely *step ordered fuzzy numbers* on which an approximation formula of a set of defuzzification functionals will be searched based on some number of training data.

4.1 Problem formulation

Let us think how recent representation can help us in the following approximation problem.
Problem. Let a finite set of training data be given in the form of N pairs: ordered fuzzy number and value (of action) of a defuzzification functional on it, i.e. $\mathsf{TRE} = \{(A_1, r_1), (A_2, r_2), ..., (A_N, r_N)\}$. For a given small ϵ find a continuous functional $H : \mathcal{R}_K \to \mathbb{R}$ which approximates the values of the set TRE within the error smaller than ϵ, i.e. $\max\limits_{1 \leq p \leq N} |H(A_p) - r_p| \leq \epsilon$, where $(A_p, r_p) \in \mathsf{TRE}$.
Problem may possess several solution methods, e.g. a dedicated evolutionary algorithm ((Kosiński, 2007; Kosiński & Markowska-Kaczmar, 2007)) or an artificial neural network. We have use the representation (6) of the searched defuzzification functional in which a homogeneous, of order one, function Ψ appears. It means that values of this function are determined from its arguments situated on the unit sphere $\mathcal{S}_{2K-1}$ in $2K-1$ D space.
If a genetic algorithm is in use then the form of genotypes could be rather standard: it is a vector of $2K$ components. Dedicated genetic operators could be constructed: crossover and two-point mutation. Possible fitness function can be based on the inverse of an error function.

Numerical examples will be given in the next subsection. First a genetic, evolutionary, method will be presented; then artificial neural network will be in use.

4.2 Dataset

Training and test sets used in the further section (from now denoted as TRE and TES, respectively) have the following form. A set of N elements is composed of N pairs of OFN and a value of a defuzzification functional on it, i.e.: $\{(A_1, r_1), (A_2, r_2), ..., (A_N, r_N)\}$.
In order to create these sets we use the approach utilizing points on a unit sphere φ_{2K-1} in $2K-1$ D space. First, we select points on a sphere φ_{2K-1}. Part of these points is completely random and part of them has to fulfil a certain condition. Given a point on a sphere φ_{2K-1} in a form $(u_2, u_3, ..., u_K, v_1, v_2, ..., v_K)$ we select points where $u_i < u_{i+1}$, for $i = 2, 3, ..., K$ and $v_j > v_{j+1}$ for $j = 1, 2, ..., K$. This allows us to create fuzzy numbers with trapezoidal shape.
Next step involves adding a component u_1. Value of this component can be either 0 or selected from range $[-4; 4]$. Value of component u_1 is added to each other component from a point on sphere φ_{2K-1}.
Following sets were created using this method: 2 TRE sets, consisting of 40 OFNs that meet restriction mentioned earlier and 20 completely random OFNs. TRE1=TRE_0 set used u_1 component of value 0 while TRE2=TRE_4 set has a u_1 component from range $[-4; 4]$. Respective values of a defuzzification functional of each OFN were calculated. TES1 and TES2 sets were created using the same approach, each of them having 30 elements.

4.3 Genetic algorithm for linear defuzzification functional approximation

Chromosome represents the vector in the defuzzification functional H. Then we use the following procedure for approximation :

- chromosome is encoded using $2K$ real values represented as fractions and has the following form: $(c_1, c_2, ..., c_k, d_1, d_2, ..., d_k)$
- $c_i, d_j \in [0, 1]$
- $\sum_{i=1}^{K} (c_i + d_i) = 1$

Given fuzzy number A and some chromosome we can calculate the defuzzified value:

$$H(A) = \sum_{i=1}^{K} (c_i u_i + d_i v_i) \,. \tag{9}$$

Error and fitness

Having a set of P instances on which we validate a chromosome, we calculate the error with the following formula:

$$Error = \frac{1}{P} \sum_{i=1}^{i=P} (H(A^i) - r^i)^2 \,.$$

For the fitness we have chosen the simple representation:

$$Fitness = 1/Error \,.$$

Genetic operations

Two genetic operations have been used:

- mutation - a randomly chosen small value was added to the gene
- two-point crossover .

Repair operation was needed to ensure that the new values fulfill the aforementioned constrains. In case when the values failed to meet the conditions they were increased or decreased proportionally.

Results for genetic algorithm

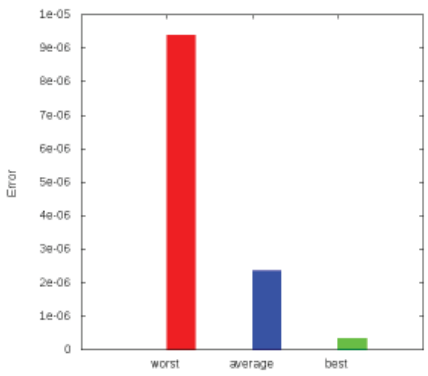
Average from results from 10 runs was:

Set	Fitness	Error
TES_0	426567.7056282262	2.344293735333886E-6
TES_4	1.1362690028303238E7	8.800732903116319E-8

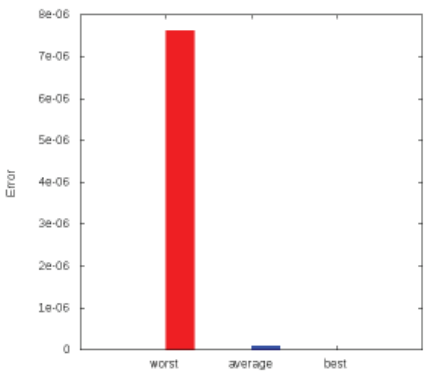
Table 1. Average results from 10 runs

Averaged and rounded chromosomes from 10 runs:

- for TRE_0:
  0, 0, 0, 0, 0, 0, 0, 0.04, 0.07, **0.37**,0, 0, 0, 0, 0, 0, 0.01, 0.03, 0.13, **0.341**
- for TRE_4:
  0, 0, 0, 0, 0, 0, 0, 0, 0.01, 0.16, **0.37**,0, 0, 0, 0, 0, 0, 0.01, 0.03, 0.15, **0.30**



(a) Subfigure 1



(b) Subfigure 2

Fig. 1. Results for genetic algorithm: a) on TRE_0, b) on TRE_4

Best error equals 0 as the algorithm managed to find the exact operator.

4.4 Results obtained using genetic programming

In order to approximate a value of a nonlinear defuzzification functional, an algorithm using genetic programming was used. Algorithm tried to build a tree, that, when evaluated, would minimize the error value of the approximation. Possible nodes consisted of a parameter node (in this case a value of u_i or v_i from processed OFN), integer constant node or a function node. The following set of functions was available to the algorithm: addition, subtraction, multiplication, division, power.

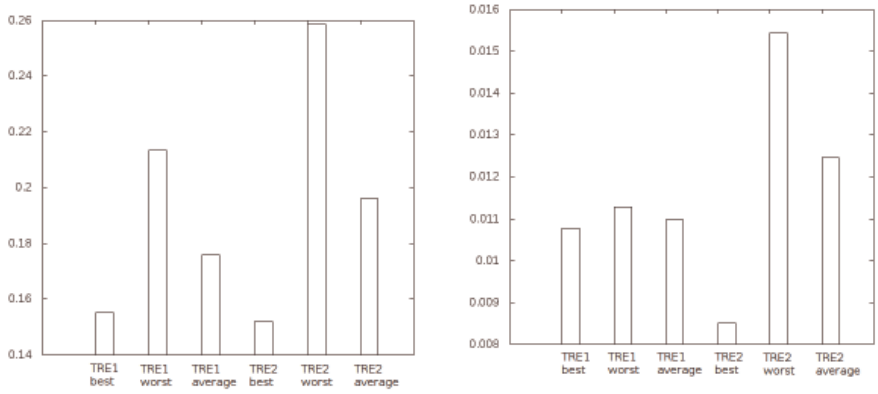
Genetic operators consisted of *mutation* and *crossover*. Mutation replaced a selected subtree of the function tree with a new tree. Crossover operation swapped a subtree between parents. Initial population was created randomly using the described building blocks, with the tree depth limit of 12. In each iteration algorithm evaluated current population of function trees, maintained 5% of best solutions and created new population using one of the two methods. First method consisted of a roulette selection of trees that were later subject to mutation and crossover operations. Second method created completely new trees and added them to the pool. 90% of population members were selected using roulette, while 10% were new trees during each iteration.
Tables 2 and 3 contain approximation numerical results obtained using this genetic approach.

Dataset	Best	Average	Worst
TRE1	0.010774742	0.0112900473	0.0109660603
TRE2	0.0084995761	0.0154348379	0.0124602126

Table 2. Genetic programming results, root mean square error, RMSE

Dataset	Best	Average	Worst
TRE1	0.1552756869	0.213444347	0.1759717912
TRE2	0.1519504725	0.2586461938	0.1960008602

Table 3. Genetic programming results, approximation error



(a) Subfigure 1 (b) Subfigure 2

Fig. 2. Errors for both sets (figure a) and RSME error (figure b)

In Fig.3 a final tree is presented. Examples of evolved functions for TRE2 data set are:

$$H\underline{u},\underline{v}) = (v_{10} - ((v_{10} - u_9)/\sqrt[3]{9}) \ , \ H(\underline{u},\underline{v}) = v_{10}\sqrt{u_9/v_{10}}\,.$$

4.5 Neural network simulations

Previous sections presented genetic algorithm method for the defuzzification approximation. This problem can be solved by neural network approximation. We present in this section our neural network approach, used for this purpose, and the results obtained of our neural

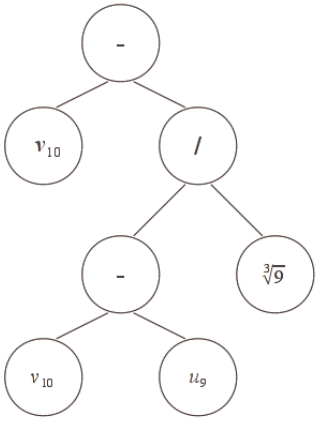


Fig. 3. Example of function tree generated in genetic programming.

network defuzzification.

Data generation

The procedure to generate TRE and TES sets was the following.

1. Generate 60 random points on a $2K-1$ dimensional hyper-sphere, where $K=10$. Let $\varphi = (u_2, u_3, ..., u_{K-1}, v_1, v_2, ..., v_K)$ be one of these points. All points fulfill the conditions $u_n < u_{n+1}$ and $v_m > v_{m+1}$. This ensures that the generated fuzzy numbers have a trapezoidal shape. In the further parts this assumption has been omitted.
2. Generate two sets of fuzzy numbers using the following methods of generating a value of u
   - $u = 0$
   - u is a random value from $(-4, 4)$
3. For each fuzzy number find the defuzzified value and split the sets in ratio 2:1 to form:
   - TRE_0 and TES_0 from fuzzy numbers with $u_1 = 0$
   - TRE_4 and TES_4 from fuzzy numbers with $u_1 \in (-4, 4)$

4.6 Implementation of a neural network

In order to make approximation of linear and the nonlinear defuzzification functionals on step ordered fuzzy numbers (SOFN) a package of artificial neural networks (ANN) has been used. The following structure of three layered MLP neural network has been assumed:

- Since each SOFN is represented by a vector of $2K$ number, each input to artificial neural networks has $2K$ real-valued components.
- one hidden layer composed of 5 neurons that build a weighted sum.
- one 1D output layer.

All of the NN Simulation was done with the help of GNU Octave 3.0.1. The structure of the network is given below on Fig.12.

4.7 Linear Defuzzification

4.7.1 ANN training

The general strategy was to train the network with data sets having $2K$ inputs and an output representing the discrete values of fuzzy output values and the crisp output calculated according to selected standard defuzzification algorithms. For the linear defuzzification we have used: MOM (middle of maximum), LOM (last of maximum), FOM (first of maximum), and COA (center of area).
Table 4 presents the final training MSE (for RSME[%]) for all the used methods. Table 5 presents the final training gradient for all the used methods.

Training Set	MOM	LOM	FOM	COA
TRE_0	1.196156E-11	1.17966E-11	3.2052E-11	3.167E-8
TRE_4	8.22773E-10	1.51997E-9	3.1339E-9	1.03805E-6

Table 4. Final training RMSE

Training Set	MOM	LOM	FOM	COA
TRE_0	3.57907E-6	1.14851E-6	1.09344E-6	2.842E-5
TRE_4	0.001232	0.0001864	0.00311	0.03940

Table 5. Final training gradient

4.7.2 ANN validation

The validation of our neural network is done by testing the network with TES_0 and TES_4 data sets generated with all of the following defuzzification methods : MOM(middle of maximum), LOM (last of maximum), FOM (first of maximum) and COA (center of area).
The validation of data TES_0 and TES_4 defuzzified with MOM strategy converges successfully. The results are presented at the figures: for TRE_0 MSE [%] (Figure 4), gradient (Figure 5), for TRE_4 RMSE (Figure 6), gradient (Figure 7). Similar results have been obtained for other defuzzification methods.
Simulation results, conclusions

Performed simulation proved that ANN can successfully represent the defuzzification strategies. Linear approximations of defuzzification functionals with MOM, LOM, FOM, and COA were correct. The trained ANN approximations for all the methods were successfully tested with TES_0 and TES_4 data sets. Table 6 presents the final validation RMSE for all the used methods.

Testing Set	MOM	LOM	FOM	COA
TES_0	1.781138E-5	3.020065E-5	0.0001056	0.00668950
TES_4	4.300E-9	2.02054E-6	0.0006829	0.007569

Table 6. Final linear validating RMSE[%]

4.8 Nonlinear defuzzification functional

Similar method has been used for nonlinear defuzzification functional, namely for the center of gravity (COG). The validation of data TES_0 and TES_4 defuzzified with COG strategy converges successfully.

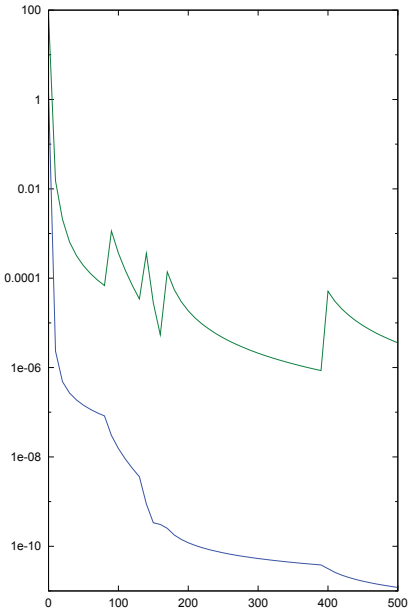


Fig. 4. MOM: TRE_0 MSE [%] and Gradient

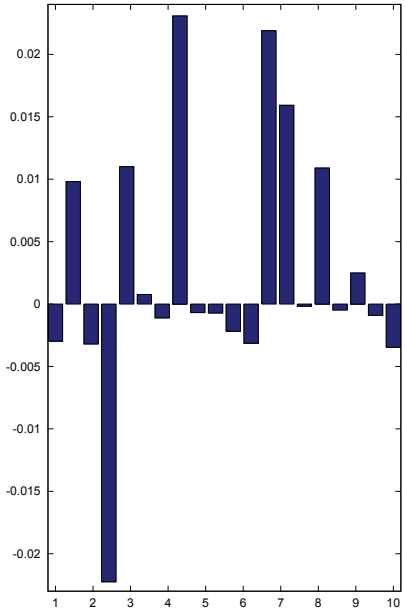


Fig. 5. MOM TES_0 MSE [%]

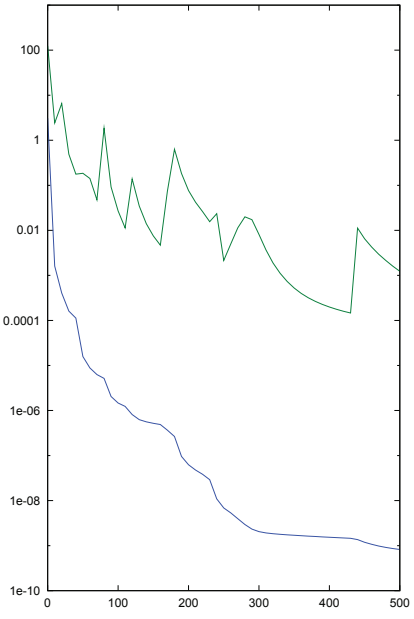


Fig. 6. MOM TES_4: MSE [%] and Gradient

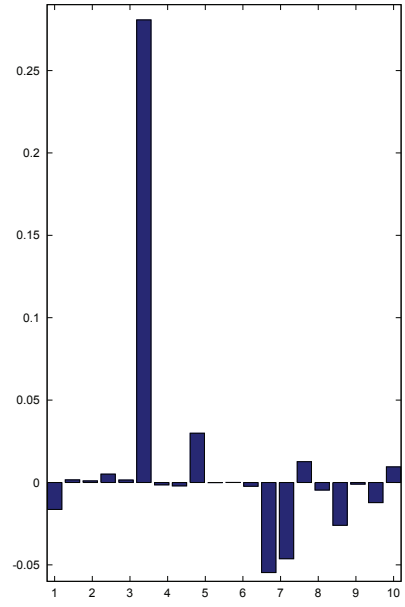


Fig. 7. MOM TES_4: RMSE

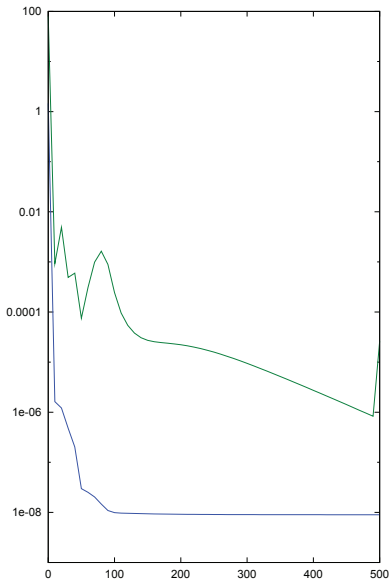


Fig. 8. COG TRE_0: RMSE and gradient

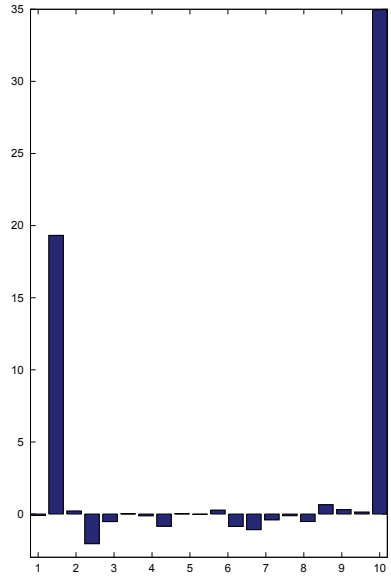


Fig. 9. COG on TES_0: RMSE

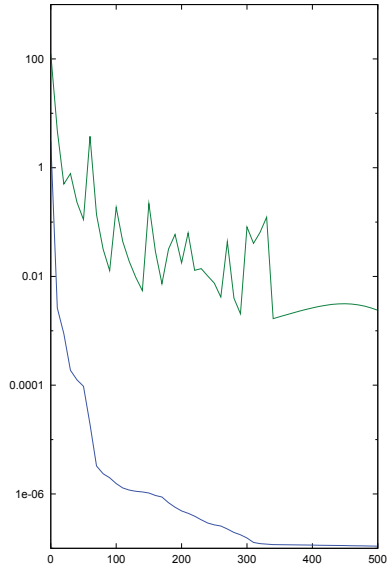


Fig. 10. COG on TES_4: RMSE and gradient

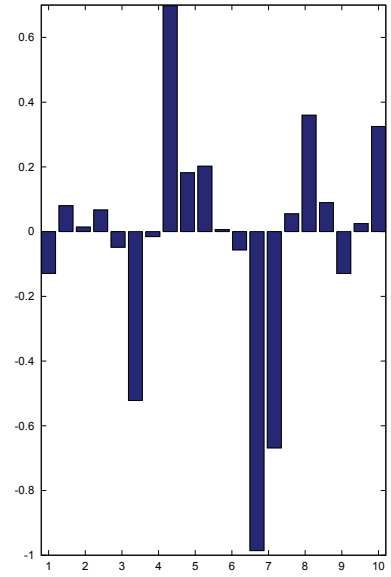


Fig. 11. COG on TES_4: RMSE

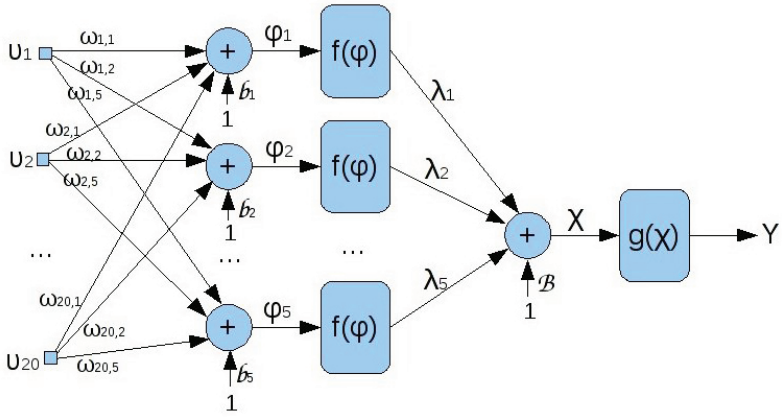


Fig. 12. Neural network structure

4.8.1 Function representation of neural network

In this subsection we present the complex function composition realized by the neural network. Its detailed structure is on Fig.12.

Transfer functions

The first layer transfer function is given by the formula :

$$f(x) = \frac{2}{1+e^{-2x}} - 1$$

The hidden layer transfer function is given by $g(x) = x$, and the output is given by

$$Y = g(X) = X = \sum_{j=1}^{5} \Phi_j \lambda_j + B$$

where $\Phi_j = f(\varphi_j) = f(\sum_{i=1}^{20} u_i * \omega_{i,j} + b_j)$. Hence we have

$$Y = \sum_{j=1}^{5} f(\sum_{i=1}^{20} u_i \omega_{i,j} + b_j))\lambda_j + B\,.$$

The weights and other parameters are listed below in tables.

5. Conclusion

The present paper brings an outline of a model of an evolutionary algorithm defined on a space of fuzzy date represented by ordered fuzzy numbers. Moreover, it shows how evolutionary algorithms and genetic programming can be used to find an approximation formula of defuzzification functionals defined on the space of step ordered fuzzy numbers.

Input	ω_{i1}	ω_{i2}	ω_{i3}	ω_{i4}	ω_{i5}
U_1	-1.500000	1.500000	-0.500000	0.500000	0.100000
U_2	-1.764203	1.721917	-1.163269	0.325977	-0.766464
U_3	-0.909134	0.969637	-1.070874	0.164166	-0.183957
U_4	-1.888874	1.870025	-1.096356	0.652329	-0.950350
U_5	-1.551447	1.618123	-0.028711	-0.164044	-1.205991
U_6	-1.339349	1.297634	0.030845	0.789391	-1.386863
U_7	-0.924551	0.758894	-0.292342	0.406089	0.273688
U_8	-1.444772	1.426924	0.270408	0.433603	-0.521424
U_9	-2.290977	2.078550	0.003299	0.771404	1.510400
U_{10}	-2.035505	1.984894	0.007624	.370735	-0.662857
U_{11}	-1.543991	1.181070	0.511984	0.784363	0.820112
U_{12}	-1.893619	1.545661	0.350312	0.843925	1.318368
U_{13}	-1.690412	1.296557	-0.424741	1.410183	1.098513
U_{14}	-1.583763	1.208434	0.315078	0.782784	1.159102
U_{15}	-1.477475	1.123565	-0.048310	0.786998	0.350696
U_{16}	-1.629397	1.389360	-0.185788	0.619010	0.175965
U_{17}	-1.758531	1.715217	0.851437	0.196586	-0.565551
U_{18}	-1.477277	1.287993	-0.478476	0.377743	1.132364
U_{19}	-0.877315	0.820842	-0.353786	-0.066219	0.204631
U_{20}	-2.090962	1.904204	-0.355554	0.233310	0.775427

Table 7. NN structure after learning

Training Set	b_1	b_2	b_3	b_4	b_5
Nonlinear TRE_0	0.562239483	0.425288679	1.616618821	1.759599096	1.0704185326

Table 8. First layer bias

Training Set	$\lambda 1$	λ_2	λ_3	λ_4	λ_5
Nonlinear TRE_0	0.0814377120	0.001417061	-28.63429759	-0.0005601	14.19819490

Table 9. Hidden layer weight

Moreover, the results of approximation have been compared with that obtained with a help of different tool of the computational intelligence, namely of artificial neural networks. We can, therefore, conclude that both tools are helpful. It is rather evident that further research in this field should follow.

6. Acknowledgments

The research work on the paper was partially funded by grant N N519 5788038 from the Polish Ministry of Science and Higher Education (MNiSW).

7. References

Buckley James J. and Eslami E., (2005).*An Introduction to Fuzzy Logic and Fuzzy Sets*, Physica-Verlag, Springer-Verlag, Heidelberg.

Chen Guanrong, Pham Trung Tat,(2001). *Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems*, CRS Press, Boca Raton, London, New York, Washington, D.C.

Drewniak J., (2001): Fuzzy numbers (in Polish), in: *Zbiory rozmyte i ich zastosowania*, Fuzzy sets and their applications, J. Chojcan, J. Łęski (eds.), Wydawnictwo Politechniki Śląskiej, Gliwice, pp. 103–129.

Dubois D., Prade H., (1978). Operations on fuzzy numbers, *Int. J. System Science*, 9, 576–578.

Goetschel R. Jr., Voxman W., (1986): Elementary fuzzy calculus, *Fuzzy Sets and Systems*, 18, 31–43.

Gruszczyńska A., Krajewska I.,(2008). *Fuzzy calculator on step ordered fuzzy numbers*, in Polish, UKW, Bydgoszcz, 2008.

Kaucher E., (1980). Interval analysis in the extended interval space IR, *Computing, Suppl.* 2, (1980), 33–49.

Klir G.J., (1997): Fuzzy arithmetic with requisite constraints, *Fuzzy Sets and Systems*, 91, 165–175.

Koleśnik R., Kosiński W., Prokopowicz P., Frischmuth K., (2004). On algebra of ordered fuzzy numbers, in: *Soft Computing – Foundations and Theoretical Aspects*, K. T. Atanassov, O. Hryniewicz, J. Kacprzyk (eds.) Akademicka Oficyna Wydawnicza EXIT, Warszawa 2004, pp. 291–302.

Kosiński W., (2004): On defuzzyfication of orderd fuzzy numbers, in: *Artifical Intelligence and Soft Computing ICAISC 2004*, Proc. of the 7th Intern. Conf. Zakopane, Poland, June 2004, Rutkowski L., Siekmann J., Tadeusiewicz R., Zadeh L A.,(eds), Springer, Berlin, 2004, pp. 326–333

Kosiński W., Słysz P., (1993): Fuzzy reals and their quotient space with algebraic operations, *Bull. Pol. Acad. Sci., Sér. Techn. Scien.*, 41 (30), 285-295.

Kosiński W., Weigl M .,(1998). General mapping approximation problems solving by neural networks and fuzzy inference systems, *Systems Analysis Modelling Simulation*, 30 (1) (1998) 11-28.

Kosiński W., Piechór K., Prokopowicz P., Tyburek K., (2001). On algorithmic approach to operations on fuzzy reals, in: *Methods of Artificial Intelligence in Mechanics and Mechanical Engineering*, T. Burczyński, W. Cholewa (eds.), PACM, Gliwice, pp. 95-98.

Kosiński W., Prokopowicz P., (2004). Algebra of fuzzy numbers (in Polish), *Matematyka Stosowana. Matematyka dla Społeczeństwa*, 5/46, 37–63.

Kosiński W., Prokopowicz P., Ślęzak D., (2002). Fuzzy numbers with algebraic operations: algorithmic approach, in: *Intelligent Information Systems 2002*, M. Kłopotek, S.T. Wierzchoń, M. Michalewicz (eds.), Proc. IIS'2002, Sopot, June 3-6, 2002, Poland, Physica Verlag, pp. 311-320.

Kosiński W, Prokopowicz P, Ślęzak D., (2003). Ordered fuzzy numbers. *Bulletin of the Polish Academy of Sciences, Sér Sci Math* **51** (3), 327–338.

Kosiński W., Prokopowicz P., Ślęzak D., (2003). On algebraic operations on fuzzy reals, in: *Advances in Soft Computing*, Proc. of the Sixth Int. Conference on Neural Network and Soft Computing, Zakopane, Poland, June 11-15, 2002, Physica-Verlag, Rutkowski L., Kacprzyk J., (eds.), pp. 54–61.

Kosiński W., P. Prokopowicz P., Ślęzak D., (2003). On algebraic operations on fuzzy numbers, in *Intelligent Information Processing and Web Mining*, M. Kłopotek, S.T. Wierzchoń, K. Trojanowski (eds.), Proc. of Int. IIS: IIPWM'03, Conference held in Zakopane, Poland, June 2-5,2003, Physica-Verlag, pp. 353–362.

Kosiński W., Prokopowicz P., Ślęzak D., (2002). Drawback of fuzzy arithmetics Ű new intuitions and propositions, in: *Proc. AI METH, Methods of Aritificial Intelligence* , T. Burczyński, W. Cholewa, W. Moczulski, (eds), Gliwice, Poland, pp. 231–237.

Kosiński W., Prokopowicz P., Ślęzak D., (2005). Calculus with fuzzy numbers in: *Proc. Intern.Workshop on Intelligent Media Communicative Intelligence, Warszawa, September, 2004*, L.Bolc, T. Nishida, Z. Michalewicz,(eds), LNCS, vol. 3490, Springer, Heidelberg, (2005), in print.

Kosiński, W. (2007), Evolutionary algorithm determining defuzzyfication operators, *Engineering Applications of Artificial Intelligence*,ă**20** (5),ă 2007, 619–627, doi:10.1016/j.engappai.2007.03.003.

Kościeński K., (2010). *Modul of step ordered fuzzy numbers in control of material point motion*, in Polish, PJWSTk, Warszawa, 2010.

Kosiński W.,(2006). On fuzzy number calculus, *Int. J. Appl. Math. Comput. Sci.*, 16 (1), 51–57.

Kosiśki W., Markowska-Kaczmar U.,(2007). An evolutionary algorithm determining a defuzzyfication functional, *Task Quarterly*, **11** (1-2), 47–58.

Kosiśki W., Prokopowicz P., Kacprzak D., (2009). Fuzziness - representation of dynamic changes by ordered fuzzy numbers, Chapter in *Views of Fuzzy Sets and Systems from Different Perspectives*, Rudolf Seising (Ed.): Studies in Fuzziness and Soft Computing Vol. 243, Springer, Berlin,Heidelberg, pp. 485–508.

Kosiśki W., Piasecki W., and Wilczyñska-Sztyma D.,(2009). On Fuzzy Rules and Defuzzification Functionals for Ordered Fuzzy Numbers, in*Proc. of AI-Meth'2009 Conference, Nowvember, 2009*, Burczyñski T., Cholewa W., Moczulski W., (eds.) AI-METH Series, pp.161-178, Gliwice 2009

Kosiński W. & Wilczyńska-Sztyma D.,(2010). Defuzzification and implication within ordered fuzzy numbers, in *WCCI 2010 IEEE World Congress on Computational Intelligence July, 18-23, 2010 - CCIB, Barcelona, Spain*, 2010, pp. 1073–1079.

Kosiński W.,(2010) under preparation, 2010

Łojasiewicz S. (1973), Introduction to the Theory of Real Functions (in Polish), Biblioteka Matematyczna, Tom 46, PWN, Warszawa.

Martos B.,(1983), Nonlinear Programming - Theory and methods, (Polish translation of the English original published by Akadémiai Kiadó, Budapest, 1975) PWN, Warszawa (1983).

Nguyen H.T., (1978). A note on the extension principle for fuzzy sets, *J. Math. Anal. Appl.* 64, 369–380.

Prokopowcz P., (2005). *Algorithmisation of operations on fuzzy numbers and its applications*(in Polish) *Algorytmizacja działań na liczbach rozmytych i jej zastosowania*, Ph. D. Thesis, IPPT PAN, Warszawa.

Prokopowicz P.,(2006). Using ordered fuzzy numbers arithmetic, in: *Fuzzy Control in Artificial Intelligence and Soft Computing*, Proc. of the 8th International Conference on Artificial Intelligence and Soft Computing Zakopane, Polska, 2004, June 25-29, 2006, Cader A, Rutkowski L., Tadeusiewicz R., Zurada J. (Eds.), Academic Publishing House EXIT, Warsaw, pp.156–162.

Rudnicki R.,(2010). Private communication.

Sanchez E., (1984). Solutions of fuzzy equations with extended operations, *Fuzzy Sets and Systems*, 12, 237-248.

Van Leekwijck W. and Kerre E. E.,(1999). *Defuzzification: criteria and classification*, *Fuzzy Sets and Systems*, **108**, 159–178.

Wagenknecht M., Hampel R., Schneider V., (2001). Computational aspects of fuzzy arithmetic based on Archimedean *t*-norms, *Fuzzy Sets and Systems*, 123/1, 49–62.

Wagenknecht M., (2001): On the approximate treatment of fuzzy arithmetics by inclusion, linear regression and information content estimation, in: *Zbiory rozmyte i ich zastosowania*, *Fuzzy sets and their applications*, J. Chojcan, J. Łęski (eds.), Wydawnictwo Politechniki Śląskiej, Gliwice, 291-310.

Zadeh L.A., (1965). Fuzzy sets, *Information and Control*, 8 338–353.

Zadeh L.A., (1975). The concept of a linguistic variable and its application to approximate reasoning, Part I, *Information Sciences*, 8, 199–249.

12

Variants of Hybrid Genetic Algorithms for Optimizing Likelihood ARMA Model Function and Many of Problems

Basad Ali Hussain Al-Sarray and Rawa'a Dawoud Al-Dabbagh
Baghdad University (Computer Science Dept; Collage of Science)
Iraq

1. Introduction

Optimization is essentially the art, science and mathematics of choosing the best among a given set of finite or infinite alternatives. Though currently optimization is an interdisciplinary subject cutting through the boundaries of mathematics, economics, engineering, natural sciences, and many other fields of human Endeavour it had its root in antiquity. In modern day language the problem mathematically is as follows - Among all closed curves of a given length find the one that closes maximum area. This is called the Isoperimetric problem. This problem is now mentioned in a regular fashion in any course in the Calculus of Variations. However, most problems of antiquity came from geometry and since there were no general methods to solve such problems, each one of them was solved by very different approaches.

Generally, optimization algorithms can be divided in two basic classes: deterministic probability algorithm. Deterministic algorithm are most often used if a clear relation between the characteristic of possible solutions and their utility for a given problem exists. If the relation between a solution candidate and its fitness are not so obvious or too complicated, or the dimensionality of the search space is very high, it becomes harder to solve a problem deterministically. Trying it would possible result in exhaustive enumeration of the search space, which is not feasible even for relatively small problem.

Then, the probabilistic algorithm come in to play. The increased availability of computing power in past two decades has been used to develop new techniques of optimization Today's computational capacity and the widespread Availability of computers have enabled development of new generation of intelligent computing techniques, such as genetic algorithm.

Evolutionary Algorithm are population met heuristic optimization algorithms that use biologic- inspired mechanisms like mutation, crossover, natural selection, and survival of the fittest in order to refine a set of solution candidates iteratively [Weise, 2009].

All evolutionary algorithms proceed in principle according to the scheme illustrated in fig.(1).

A simple Genetic Algorithm sGA is search algorithms based on the mechanics of natural selection and neutral genetics. They combine survival of fittest among string structures with a structure yet randomized information exchange to form a search algorithm with some of

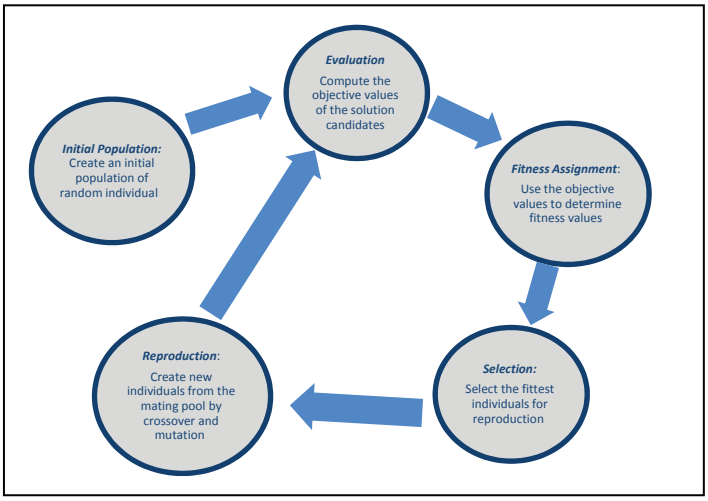


Fig. 1. Cyclic life of an evolutionary algorithms

the innovative flair of human search. In every generation; a new set of artificial creatures (string) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. They efficiently exploit historical information to speculate on a new search points with expected improved performance. A hybrid genetic algorithm (HGA) is the coupling of two processes: the simple GA and a local search algorithm. HGAs have been applied to a variety of problems indifferent fields, such as optical network design [Sinclair,2000], signal analysis [Sabatini, 2000], and graph problems [Magyar et al, 2000], among others. In these previous applications, the local search part of the algorithm was problem specific and was designed using trial-and-error experimentation without generalization or analysis of the characteristics of the algorithm with respect to convergence and reliability. The purpose of this study is to develop variants of hybrid simple genetic algorithm with local search algorithm represent by gradient or global algorithm present by evolution strategy to optimize solution of some functions where classifies as multimodal function and unimodel functions. One of import function of this study is likelihood function of time series autoregressive moving average $ARMA(1,1)$ model, this function defined as a unimodel function it is one of fundamental importance in estimation theory. The other functions used in this study as a test function used widely as benchmark functions. This study presents the($HGA1$) which is represent hybrid of simple genetic algorithm with an widely local search algorithm used steepest decent algorithm the other approach of hybrid denoted by $HGA2$ is coupling simple genetic algorithm with global search algorithm multimember evolution strategy, compares its performance with the simple(sGA), steepest descent algorithm(SDA), multimember evolution strategy ES; to study the behaviours many of functions classified as its kind multimodal or unimodel function which is used as test functions. The reminder of this chapter, section 2 presents definitions needed, section 3 giving a brief overview of genetic algorithms, representation of search points and their fitness evolution, selection, recombination, and mutation mechanisms. Then to be consistent, section 4 introduce the characteristic components of local search and its operators , also section 5 issue of the multimember evolution strategy. Section 6 address the issue of coupling simple genetic algorithm with multimember evolution strategy. Section 7 is an

extension of the results of section, in which are representative of the classes of unimodel , and multimodal function. In which competition is raised.

2. Definitions

Definition 2.1 (Objective Function) An objective function $f: \mathbb{X} \to \mathbb{Y}$ with $\mathbb{Y} \subseteq \mathbb{R}$ is a mathematical function which is subject to optimization.
The co-domain $\mathbb{Y}$ of an objective function as well as its range must be a subset of the real numbers $\mathbb{Y} \subseteq \mathbb{R}$. The domain $\mathbb{X}$ of f is called problem space and can represent any type of element like numbers, lists, construction plans, and so on. It is chosen according to the problem to be solved with the optimization process. Objective functions are not necessarily mere mathematical expressions, but can be complex algorithms that, for example , involve multiple simulations. Global optimization comprises all techniques that can be used to find the best element $x^* \in \mathbb{X}$ with respect to such criteria $f \in F$.
Definition 2.2 (local Maximum) A local maximum $\hat{x}_l \in \mathbb{X}$ of one (objective) function $f: \mathbb{X} \to \mathbb{R}$ *is a*n input element $f(\hat{x}_l) \geq f(x)$ for all x neighbouring $\hat{x}_l$. If $\mathbb{X} \in \mathbb{R}^n$, we can write:

$$\forall \hat{x}_l \exists \epsilon > 0: f(\hat{x}_l) \geq f(x) \forall x \in \mathbb{X}, |x - \hat{x}_l| < \epsilon.$$

Definition 2.3 (Local Optimum). A (local) minimum $\hat{x}_l \in \mathbb{X}$ of one (objective) function $f: \mathbb{X} \to \mathbb{R}$ is an input element with

$f(\hat{x}_l) \leq f(x)$ for all x neighbouring $\hat{x}_l$, $\forall \hat{x}_l \exists \epsilon > 0: f(\hat{x}_l) \leq f(x) \forall x \in \mathbb{X}, |x - \hat{x}_l| < \epsilon$.

Definition 2.4 (Local Optimum).A local optimum $x^* \in \mathbb{X}$ of one (objective) function $f: \mathbb{X} \to \mathbb{R}$ is either a local maximum or a local minimum.
Definition 2.5 (Global Maximum). A global maximum $\hat{x} \in x$ of one (objective) function $f: \mathbb{X} \to \mathbb{R}$ is an input element with

$$f(\hat{x}) \geq f(x) \forall x \in \mathbb{X}.$$

Definition 2.6 (Global Maximum). A global maximum $\hat{x} \in x$ of one (objective) function $f: \mathbb{X} \to \mathbb{R}$ is an input element with

$$f(\hat{x}) \leq f(x) \forall x \in \mathbb{X}.$$

Definition 2.7 (Local Optimum): A global optimum $x^* \in \mathbb{X}$ of one (objective) function $f: \mathbb{X} \to \mathbb{R}$ is either a global maximum or a global minimum. Even a one-dimension function $f: \mathbb{X} = \mathbb{R} \to \mathbb{R}$ may have more than one global maximum, multiple global minimum, or even both in its domain $\mathbb{X}$. Take the sine or cosine function for example; for cosine function it has global maximum $\hat{x}_i = 2i\pi, (i = 0,1,2, \dots)$ and global minimum $\hat{x}_i = (2i + 1)\pi, (i = 1,2, \dots)$.
Definition 2.8 (Solution Candidate): A solution candidate x is an element of the problem space $\mathbb{X}$
Definition 2.9 (Solution Space): we call the union of all solutions of an optimization problem its solution space $\mathbb{S}$. $\mathcal{X}^* \subseteq \mathbb{S} \subseteq \mathbb{X}$
This solution space contain (and can be equal to) the global optimal set $\mathcal{X}^*$. There may exist valid solution $x \in \mathbb{S}$ which are not elements of $\mathcal{X}^*$, especially in the context of constraint optimization.

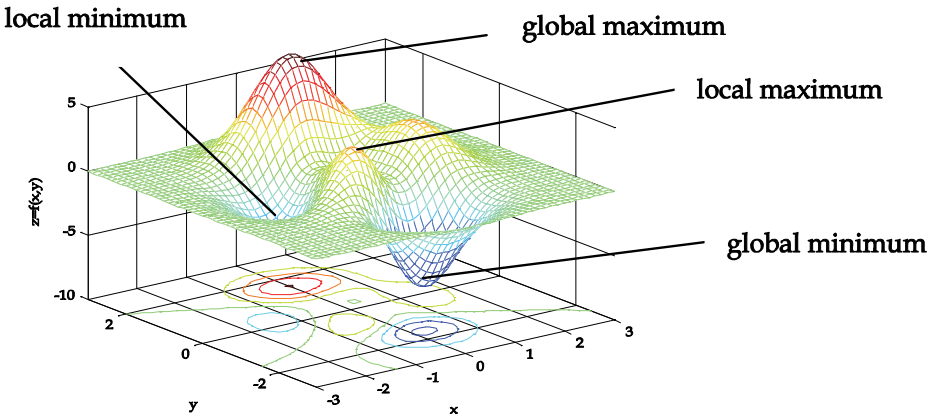


Fig. 2. An example of function with multi global and local maximum and minimum optimal point.

Definition 2.10 (Search space) : The search space $\mathbb{G}$ of an optimization problem is the set of all elements $\mathfrak{g}$ which can be processed by the search operations. The type of the solution candidates depends on the problem to be solved. Since there are many different applications for optimization, there are many different forms of problem spaces. It would be cumbersome to develop search operations time and again for each new problem space encounter.

Definition 2.11 (Genotype): the elements $\mathfrak{g} \in \mathbb{G}$ of the search space $\mathbb{G}$ of a given optimization problem are called the genotypes.

The elements of the search space rarely are unconstraint aggregations. Instead, they often consist of distinguishable parts, hierarchical units, or well-type data strictures. The same goes for DNA in biology. It consists of genes, segments of nucleic acid, that contain the information necessary to produce RNA strings in a controlled manner. A fish, for instance, may have a gene for the colour of its scales. This gene, in turn, could have two possible "values" called alleles, determining whether the scales will be brown or grey. The genetic algorithm community has adopted this notation long ago and we can use it for arbitrary search space.

Definition 2.12 (Gene). The distinguishable units of information in a genotype that encode the phonotypical properties are called gene.

Definition 2.13 (Allele): An allele is a value of specific gene.

Definition 2. 14 (Locus): The locus is the position where a specific gene can be found in a genotype.

Definition 2.15 (Search Operation): the search operation search OP are used by optimization algorithm in order to explore the search space $\mathbb{G}$.

Definition 2.16 (individual): An individual p is a tuple $(p.g, p.x)$of an element $p.g$ in the search space $\mathbb{G}$ and the corresponding element $p.x = gpmp.g$ in the problem space $\mathbb{X}$.

Definition 2.17 (Population): A population (pop) is a list of individuals used during an optimization process.

$$Pop \subseteq \mathbb{G} \times \mathbb{X}: \forall p = (p.g, p.x) \in Pop \Rightarrow p.x = gpm(p.g)$$

As already mention, the fitness $v(x)$ of an element $x$in the problem space $\mathbb{X}$ often not solely depends on the element itself. Normally, it is rather a relative measure putting the features of x in to the context of a set of solution candidates x.

2.1 Genotype-phenotype mapping

The genotype –phenotype mapping (GPM, or ontogeny mapping) $gpm: \mathbb{G} \to \mathbb{X}$ is a left-total binary relation which maps the elements of the search space $\mathbb{G}$ to elements in the problem space

$$\mathbb{X}; \forall g \in \mathbb{G}\, \exists x \in \mathbb{X} : \; gpm(g) = x$$

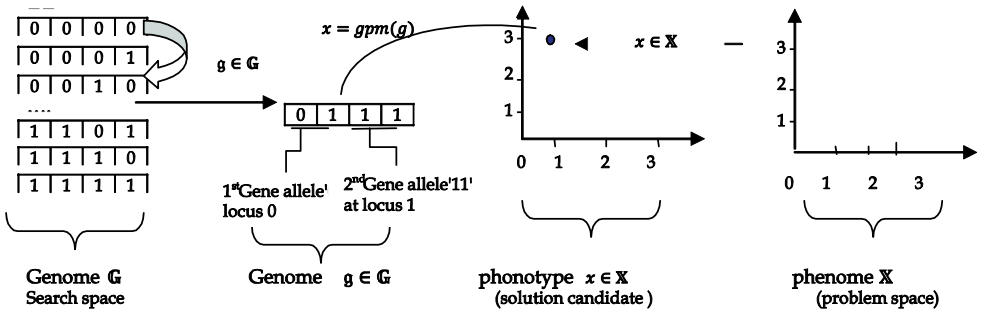


Fig. 3. The relation of genome, genes, and the problem space.

3. Genetic algorithm

3.1.1 Initialization

The first step is the creation of an initial population of solutions, or chromosomes. The populations of chromosomes generally chosen at random, for example, by flicking a coin or by letting a computer generate random numbers. There are no hard rules for determining the size of the population. Larger populations guarantee greater diversity and may produce more robust solutions, but use more computer resources. The initial population must span a wide range of variable settings, with a high degree of diversity among solutions in order for later steps to work effectively.

3.1.2 Fitness evaluation

In the next step, the fitness of the population's individuals evaluated. In biology, natural collection means that chromosomes that are more fit tend to produce more offspring than do those that are not as fit. Similarly, the goal of the genetic algorithm is to find the individual representing a particular solution to the problem, which maximizes the objective function, so its fitness is the value of the objective function for a chromosome. Genetic algorithms can of course also solve minimization problems. The fitness function (also called objective function or evaluation function) used to map the individual's chromosomes or bit strings into a positive number, the individual's fitness. The genotype, the individual's bit string, has to be decoded for this purpose into the phenotype, which is the solution alternative. Once the genotype has been decoded, the calculation of the fitness is simple: we use the fitness function to calculate the phenotype's parameter values into a positive number, the fitness. The fitness function plays the role of the natural environment, rating solutions in terms of their fitness. To apply the GA to real – valued parameters optimization problems of the form $f: \prod[u_i, v_i] \to R(u_i <$

v_i), the bit string is logically divided in to n segments of (in most cases)equal length $l_x (l = nl_x)$ and each segment is interpreted as the binary code of the corresponding object variable $x_i \in [u_i, v_i]$. A segment decoding function $\Gamma^i: \{0,1\}^{l_x} \to [u_i, v_i]$ typically looks like

$$\Gamma^i(a_{i1}a_{i2} \ldots . a_{il_x}) = u_i + \frac{v_i - u_i}{2^{l_x - 1}} [\sum a_{ij} 2^{j-1}] \quad (1)$$

where $(a_{i1}a_{i2} \ldots . a_{il_x})$ denotes the i[th] segment of an individual $\vec{a} = (a_{i1}a_{i2} \ldots a_{nl_{xx}}) \epsilon I^{n.l_x} = I^l$. Associated with each individual is fitness value. This value is a numerical quantification of how good of solution to optimization problem the individual is .Individual with chromosomal strings. Representing better solution has higher fitness values, while lower fitness values attributed to those whose bit string represents inferior solution. Combining the segment-wise decoding function to individual – decoding function $\Gamma = \Gamma^1 \times \ldots . \times \Gamma^n$, fitness values are obtained by setting

$$\Phi(\vec{a}) = \delta(f(\Gamma(\vec{a}))) \quad (2)$$

where δ denotes a scaling function ensuring positive fitness values such that the best individual receives largest fitness.

3.1.3 Selection

In the third step, the genetic algorithm starts to reproduce. The individuals that are going to become parents of the next generation selected from the initial population of chromosomes. This parent generation is the "mating pool" for the subsequent generation, the offspring. Selection determines which individuals of the population will have all or some of their genetic material passed on to the next generation of individuals. The object of the selection method is to assign chromosomes with the largest fitness a higher probability of reproduction.

3.1.4 Tournament selection

The tournament selection method select μ times the best individual from a random subset β_k of size $|\beta_k| = \xi,\ 2 \leq \xi \prec \mu\ \ \forall k \in \{1, \ldots, \mu\}$ and transfers it to the mating pool (note hat there may appear duplicates). The best individual within each subset β_k selected according to the relation $\succ^k$ (read: better then). A formal definition of the tournament selection operator $S: I^\mu \to I^\mu$ follows (Schowefel &Bäck, 1997):
Let $\beta_k \subset p(t)\ \forall\ k\ \in (1, \ldots, \mu)$ be random subsets of $P(t)$ each of size$|\beta_k| = \xi.\ \forall\ k\ \in (1, \ldots, \mu)$ choose $a^k \in \beta_k$ such that $\forall \vec{b} \in \beta_k: \vec{a}_k \overset{k}{\succ} \vec{b}$ where

$$\vec{a}_k \overset{k}{\succ} \vec{b} :\Leftrightarrow \Phi(\vec{a}) > 0\ \wedge f(\Gamma(\vec{a})) > 0\ \leq f\left(\Gamma(\vec{b}_k)\right) \quad (3)$$

3.1.5 Genetic operators

3.1.5.1 Crossover

The primary exploration operator in genetic algorithms is crossover, a version of artificial mating. If two strings with high fitness values mated, exploring some combination of their genes may produce an offspring with even higher fitness. Crossover is a way of searching the range of possible existing solutions. There are many ways in which crossover can implemented, such as one point crossover, two-point crossover, n-point crossover, or uniform crossover. In the following, we will stay with the simplest form, Holland's one-point crossover technique. Single-point crossover is the simplest form, yet it is highly effective.

One point crossover, is often used in sGA, it work first randomly picking a point between 0 and l. The participating parent individuals $\vec{x} = (x_1, \ldots . x_l)$ and $\vec{y} = (y_1, \ldots . y_l)$ are then split at the point , followed by a swapping of the split halves to form two offspring individual $\vec{x}$ *and* $\vec{y}$ as follows (Kargupta, 1995):

$$\vec{x} = (x_1, \ldots ., x_{\chi-1}, x_\chi, y_{\chi+1}, \ldots ., y_l)\vec{y} = (y_1, \ldots ., y_{\chi-1}, y_\chi, x_{\chi+1}, \ldots ., x_l \quad (4)$$

where $\chi\epsilon\{1, \ldots, l-1\}$ denotes a uniform random variable .

3.1.5.2 Mutation

If crossover is the main operator of genetic algorithms that efficiently searches the solution space, then mutation could called the "background operator" that reintroduces lost alleles into the population. Mutation occasionally injects a random alteration for one of the genes. Similar to mutation in nature, this function preserves diversity in the population. It provides innovation, possibly leading to exploration of a region of better solutions. Mutation performed with low probability. Applied in conjunction with selection and crossover, mutation not only leads to an efficient search of the solution space but also provides an insurance against loss of needed diversity, on a single individual , mutation operator $m\{p_m\}: I \to I$ formally works as follows (Back & Schwefel, 1993): $m\{p_m\}(x_1, \ldots, x_l) = (\acute{x_1}, \ldots, \acute{x_l}), (\forall i \in \{1, \ldots, l\})$:

$$\acute{x_\iota} = \begin{bmatrix} \mathrm{x_i} & , \chi_\mathrm{i} > \mathrm{P_m} \\ 1 - \mathrm{x_i} & , \chi_\mathrm{i} \leq \mathrm{P_m} \end{bmatrix} \quad (5)$$

where $\chi_i\epsilon[0,1]$ is a uniform random variable, sampled anew for each bit.

3.1.6 Conceptual algorithm

The conceptual algorithm of sGA can then formulated as
t:=0; t is the generation number
Initialize $P(0) = \{\vec{a}_1(0), \ldots, \vec{a}_\mu(0)\}\epsilon I^\mu$ Where $I = \{0,1\}^l$
Evaluate $P(0) = \{\Phi(\vec{a}_1(0)), \ldots, \Phi(\vec{a}_\mu(0))\}\epsilon I^\mu$
Where $\Phi(\vec{a}_k(0) = \delta(f\left(\Gamma(\vec{a}_1(0))\right), P(0)$
While $(\tau((P(T)) \neq true$ do // while termination criterion not fulfilled
Recombine: $\vec{\acute{a}}_k(k) = r\{p_c\}(P(t))\quad \forall k \in \{1, .., \mu)$
Mutate: $\overrightarrow{a"}_k(t) = m\{p_m\}(\vec{\acute{a}}(k) \quad \forall k \in \{1, .., \mu\}$
Evaluate $P"(t) = \left\{\overrightarrow{a"}_1(t), \ldots, \overrightarrow{a"}_\mu(t)\right\} : \{\Phi(\overrightarrow{a"}_1(t)), \ldots, \Phi(\overrightarrow{a"}_\mu(t))\}$
Where $\Phi\left(\overrightarrow{a"}_k(0)\right) = \delta\left(f\left(\Gamma\left(\overrightarrow{a"}_k(t)\right)\right), P(t-w)\right);$
end

Fig. 4. Pseudo code of sGA algorithm

3.2 Theorems and definitions needed

Definition 2.1

The directional derivatives of f(x, y) at the point (a, b) and in the direction of the unit vector $u = \langle u_1, u_2 \rangle$ is given by

$$\mathrm{D_u f(a,b)} = \lim_{\mathrm{h}\to 0} \frac{\mathrm{f(a+hu_1, b+hu_2) - f(a,b)}}{\mathrm{h}} \quad (6)$$

provided the limit exists.

Theorem 2.1

Suppose that f is differentiable at (a, b) and u = $\langle u_1, u_2 \rangle$ is any unit vector. Then we can write

$$D_u f(xa, b) = f_x(a, b) + f_y(a, b) \tag{7}$$

Clearly 2.1

For convenience, we define the gradient of a function to be vector –valued function whose component are the first –order partial derivatives of f . we denote the gradient of a function f by grad of f or ∇f read "del f" and define by the given theorem .

Theorem 2.2

If f is a differentiable function of x and y and u is any unit vector, then

$$D_u f(x, y) = \nabla f(x, y).\, u \tag{8}$$

Clearly 2.2

This theorem clear how to compute directional derivatives. Further, writing directional derivatives as a dot products. This theorem generalized to vector valued $F: R^{nx1} \rightarrow R^{nx1}$.

Theorem 2.3

Suppose that f is differentiable function of x and y at the point (a, b). Then

i. the maximum rate of change of f at (a, b) is $\|\nabla f(a, b)\|$ and occurs in the direction .

ii. of the gradient, $u = \frac{\nabla f(a,b)}{\|\nabla f(a,b)\|}$ the minimum rate of change of f at (a, b) is $-\|\nabla f(a, b)\|$ and occurs in the direction opposite the gradient u = $-\frac{\nabla f(a,b)}{\|\nabla f(a,b)\|}$.

iii. the gradient ∇f(a, b) is orthogonal to the level curve f(x,y)=c at the point (a ,b), where c=f(a, b).

Definition 2.2

We call f(a, b) a local maximum of f if there is an open disk R centred at (a, b), for which $f(a, b) \geq f(x, y)$ for $(x, y) \in R$. Similarly , f(a, b) is called a local minimum of f if there is an open disk cantered at (a, b), for which $f(a, b) \leq f(x, y)$ for $(x, y) \in R$. In either case f(a, b) is called a local extreme of f.

Theorem 2.4

suppose that f(x,y) has continuous second order partial derivatives in some open disk containing the point (a ,b) and that $f_x(a, b) = f_y(a, b) = 0$. Define the discriminant D for the point (a, b) by

$$D(a, b) = f_{xx}(a, b) f_{yy} - [f_{xy}(a, b)]^2 \tag{9}$$

if $D(a, b) > 0$ and $f_{xx}(a, b) > 0$, then f has a local minimum at (a, b).

(ii) if $D(a, b) > 0$ and $f_{xx}(a, b) < 0$, then f has a local maximum at (a, b).

(iii) if $D(a, b) < 0$, then f has a saddle point at (a, b).

(iv) if $D(a, b) = 0$, then no conclusion can be drawn.

4. Local search

The local search operator looks for the best solution starting at a previously selected point, in this case a solution in the sGA population. For this application, the steepest descent method was chosen as the local search operator. This method moves along the direction of the steepest gradient until an improved point found, from which a new local search starts.

The algorithm ends when no new relationship shown point can found (this is equivalent to a gradient equal to zero).
For functions with multiple local optimum, the method find one local optima but it is not guaranteed to find the global minimum. For geometric with conical shape, for example, the method finds the local optimum in one local search starting from any point in side the basin of attraction. For other geometries, the local search operator required more than one iteration to achieve the solution.

4.1 Descent method

Cauchy (1847), Kantorovich (1940-1945), Leven berg (1944), and Curry (1944) are the originators of the gradient strategy, which started life as a method of solving equations and systems of equations. It first referred to as aid to solving variation problems by Hadamard (1908) and Courant (1943). This variant of the basic strategy, known under the name optimum gradient method, or method of 'steepest descent'. Theoretical investigations of convergence and rate of convergence of the method can be found e.g. in Akaike (1960),Goldstein (1962), Ostrowski (1967), Zangwill(1969) and Wolfe(1969,1970,1971)[6] The general rule of steepest descent where used to find optimal solutions of nonlinear problems is

$$x^{(k+1)} = x^{(k)} + \alpha_k d^k \tag{10}$$

Where d^k is an a suitably chosen direction and α_k is a positive parameters (called step-size) that measures the step along the direction d^k. This direction is a descent direction if

$$d^{k^T} \nabla f(x^k) < 0 \qquad if \ \nabla f(x^k) \neq 0$$

$$d^k = 0 \qquad \text{if} \quad \nabla f(x^k) = 0 \tag{11}$$

4.1.1 Steepest descent algorithm

To approximate a solution p to the minimization problem $G(p) = \min_{x \in R^n} G(x)$
Given an initial approximation x:
Step 1. set $k = 1$
Step 2. While ($k \leq N$) do steps (3-8)
Step 3. Set $g_1 = G(x_1, \dots, x_n)$ // note: $g_1 = G(x^k)$;
$z = \nabla G(x_1, \dots, x_n)$ // note: $z = \nabla G(x^{(k)})$;
$z_0 = ||z||_2$
Step 4. if $z_0 = 0$ then output ("zero Gradient") Output $(x_1, \dots, x_n, g_1)$//[Procedure completed may have minimum check further]
Step 5. choose δ s.t
$g = \min (G(x_0 + \delta z)$
Step 6. Set $x = x + \delta z$
Step 7. if $|g - g_1| < Tol$ then
output $(x_1, \dots, x_n, g_1)$// [Procedure completed successfully]
Stop
Step 8. set k=k+1;
Step 9. Output ('Minimum Iterations Exceeded') // (Procedure completed unsuccessfully)
Stop

Fig. 5. Pseudo code of gradient algorithm

4.2 Hybrid genetic algorithm

In this section we define a hybrid of sGA with gradient method and we denoted as (HGA1) A hybrid genetic algorithm (HGA) is the coupling of two processes: the simple GA and a local search algorithm. The local search part of the algorithm was problem specific and designed using trial-and-error experimentation without generalization or analysis of the characteristics of the algorithm with respect to convergence and reliability. The HGA algorithm is a standard, which combines an sGA with local search. The local search step defined by three basic parameters: frequency of local search, probability of local search, and number of local search iterations. The first element for the definition of the algorithm is the frequency of local search, which is the switch between global and local search. In the HGA algorithm, this switch performed every "G!" global search generations, where "G!" is a constant number called the local search frequency. The second element of the algorithm is the probability of local search P, which is the probability that local search will be performed on each member of the sGA population in each generation where local search is invoked. This probability is constant and defined before the application of the algorithm. Finally, each time local search is performed, it is performed a constant number of local search iterations before local search is halted.

4.2.1 Basic elements

4.2.1.1 Genetic algorithm

Three basic operators define the simple Genetic Algorithm (sGA): binary tournament selection, single point crossover, elitism, and simple mutation. Through the successive application of these three operators, an initial population of solutions evolved into a highly fit population.

4.2.1.2 Local search

The local search operator looks for the best solution starting at a previously selected point, in this case a solution in the SGA population. For this application, the steepest descent method was chosen as the local search operator. This method moves along the direction of the steepest gradient until an improved point found, from which a new local search starts.
The algorithm ends when no new relationship shown point can found (this is equivalent to a gradient equal to zero) and this satisfied in our formula adaptation [10].

4.3 Hybrid genetic algorithm with local search algorithm

A hybrid genetic algorithm (HGA) is the coupling of two processes: the sGA and a local search algorithm. The local search part of the algorithm was problem specific and was designed using trial-and-error experimentation without generalization or analysis of the characteristics of the algorithm with respect to convergence and reliability. It is defined by three basic parameters: frequency of local search, probability of local search, and number of local search iterations. The first element for the definition of the algorithm is the frequency of local search, which is the switch between global and local search. In the HGA1 algorithm, this switch performed every "G!" global search generations, where "G!" is a constant number called the local search frequency. For example, if G!=3, local search would perform every 3 generations during the sGA. The second element of the algorithm is the probability of local search P, which is the probability that local search will be performed on each member of the sGA population in each generation where local search is invoked. This probability is constant and defined before the application of the algorithm. Finally, each time local search performed; it performed for a constant number of local search iterations before local search halted.

4.3.1 Conceptual algorithm of

The coupling approach in this paper consist in the introduction of generation interval for hybrid activation operator (HAO).Through selection, crossover, and mutation operators, the canonical sGA works on population of bit string encoding scheme generation by generation. When HAO is active (again implemented here every G? generation), the intermediate generation created by GA is fed into an adopted selection strategy which select subpopulation, usually of small size. Then each binary string individual in this subpopulation is convert into a real number vector, to be the initial value of steepest descent (sDA) algorithm that operate on this subpopulation for fixed small of generation. The vectors converted back into bit string values to manipulate again by master GA, sDA used here in this hybridization as a tool operates in small number of generation fashion in an order to enhance the selected points driven from the master GA. It is appropriate that the version of sDA tools to be of preservative survivor property to have worthy adjustment. The conceptual algorithm of the (HGA) given by the following steps:

Input: sample size, number of generation, ..
Output : approximate value
Step1: Initialize GA(generate Initial population of parameters).
Step2.1: for t $=$ 1to number of generation Do
Step2.2: for I $=$ 1: population size Do
The canonical genetic algorithm (sGA) operators:
Step2.2.1: Local Recombination,
Step2.2.2: Mutation,
Step2.2.3 Selection,
Step3: Binary coded sGA individual remapped in to real vector individual.
Step4.1: Select $\left(\frac{1}{3}\text{of population size}\right)$ use as initial solution of sDA perform for each generation
Step4.2. Start local search evaluation// Starting of Steepest Descent Algorithm (sDA).
Step4.3. : Real _coded local search algorithm individual remapped in to binary vector individual.
Step5. Repeat steps until all of generation complete or termination criterion is satisfied.
Step6: end

Fig. 6. Pseudo code of HGA algorithm

5. (μ^+, λ)-Evolution strategies

H.-P. Schwefel proposed the multi-member evolution strategies, the so-called (μ^+, λ)-ES. In their most general form, these strategies are described in the coming subsections.

5.1 Representation and fitness evaluation

An individual $\vec{a} = (\vec{x}, \vec{\sigma}) \epsilon I$ in (μ^+, λ)-ES can consist of the components (Robert, Roland, 2002):
$\vec{x} \epsilon R^n$: The vector of object variables.
$\vec{\sigma} \epsilon R_+^{n_\sigma}$: A vector of step length or standard deviations $(1 \leq n_\sigma \leq n)$ of the normal distribution. The strategy parameter $\vec{\sigma}$ (also called the internal model) determines the variances of the n-dimensional normal distribution, which is used for exploring the search space. The user of an evolution strategy, depending on his feeling about the degree of freedom required, can vary the amount of strategy parameters attached to an individual. As a rule of thumb, the global search reliability increases at the cost of computing time when

the number of strategy parameters is increased. The setting most commonly used which form the extreme cases are:

- $n_\sigma = 1$: (Uncorrelated mutation with one single standard deviation controlling mutation of all components of $\vec{x}$).
- $n_\sigma = n$: (Standard mutations with individual step sizes $\sigma_1, .., \sigma_n$ controlling mutation of the corresponding object variables x_i individually). The only part of $\vec{a}$ entering the objective function evaluation is $\vec{x}$, and the fitness of an individual $\phi(\vec{a})$ is identical to its objective function value $f(\vec{x})$, i.e. ($\phi(\vec{x}) = f(\vec{x})$).

5.2 Mutation operator

The generalized structure of $(\boldsymbol{\mu}^+, \boldsymbol{\lambda})$-ES mutation operator consists of the addition of a normally distributed random number to each component of the object variable vector, corresponding to a step in the search space. The variance of the step-size distribution is itself subject to mutation as a strategy variable. Formally speaking, mutation operator $\boldsymbol{m}\{\boldsymbol{\tau_0}, \boldsymbol{\tau}\}: \boldsymbol{I} \to \boldsymbol{I}$, is defined as follows [5]

$$m\{\tau_0, \tau\}(\vec{a}) = m_x(\vec{x}) \circ m_\sigma(\vec{\sigma}) = (\vec{x}', \vec{\sigma}') \tag{13}$$

Which proceeds by first mutating the strategy parameters $\vec{\boldsymbol{\sigma}}$: $\boldsymbol{m_\sigma}: \boldsymbol{R_+^{n_\sigma}} \to \boldsymbol{R_+^{n_\sigma}}$;

$$m_\sigma(\vec{\sigma}) = \vec{\sigma}' = (\sigma_1 \exp(z_1 + z_0), \dots, \sigma_{n_\sigma} \exp(z_{n_\sigma} + z_0)) \tag{14}$$

Where $z_0 \sim N(0, \tau_0^2)$, $z_i \sim N(0, \tau^2)$ $\forall\, i \in \{1, \dots., n_\sigma\}$ To prevent standard deviations from becoming practically zero, a minimal value of $\varepsilon_\sigma$is algorithmically enforced for all σ_i.
Secondly, modifying $\vec{x}$ according to the new set of strategy parameters obtained from mutating $\vec{\sigma}: m_\chi: R^n \to R^n$

$$m_x(\vec{x}) = \overrightarrow{x'} = (x_1 + z_1, \dots, x_n + z_n) \tag{15}$$

5.3 Recombination operators

In (μ^+, λ)-ES, different recombination mechanisms are used in either local form, producing one new individual from two randomly selected parent individuals, or in global form, allowing components to be taken for new individual from potentially all individuals available in the parent population. Furthermore, recombination is performed on strategy parameters as well as the object variables, and the recombination type may be different for object variables, and standard deviations.
Depending on the recombination types [4][6]:

$$Rec = \begin{cases} 0 & \text{No recombination} \\ 1 & \text{Discrete recombination of pair of parents} \\ 2 & \text{Intermediate recombination of pair of parents} \\ 3 & \text{Discrete recombination of all parents} \\ 4 & \text{Intermediate recombination of parents in pairs} \end{cases} \tag{16}$$

Sometimes, the choice of a useful recombination operator for a particular optimization problem is relatively difficult and requires performing some experiments [2].The rules of recombination operator $r: I^\mu \to I$ for creating an individual, $r\{rec_x, rec_\sigma\}(P) = \overrightarrow{a'} =$

$(\overrightarrow{x'}, \overrightarrow{\sigma'}) \in I$, are given respectively by referring to arbitrary vectors $\vec{b}$ and $\overrightarrow{b'}$ where $\vec{b}$ and $\overrightarrow{b'}$denote here the part (i.e., either $\vec{x}$ or $\vec{\sigma}$) of a pre-selected parent individuals and the part of an offspring vector receptively. Each of $\vec{b}$ and $\overrightarrow{b'}$ are of length $m \in \{n, n_{\vartheta\sigma}\}, \forall i \in \{1, \dots, m\}$

$$b_i' = \begin{cases} b_i & if \quad rec = 0 \\ b_{\chi_{1,i}} \; or \; b_{\chi_{2,i}} & if \quad rec = 1 \\ (b_{\chi_{1,i}} + \; b_{\chi_{2,i}}).\, 0.5 & if \quad rec = 2 \\ b_{\chi_{3,i}} & if \quad rec = 3 \\ (b_{\chi_{3,i}} + \; b_{\chi_{4,i}}).\, 0.5 & if \quad rec = 4 \end{cases} \tag{17}$$

Where $\chi_1, \chi_2 \sim U(\{1, \dots, \mu\}$for each offspring, and $\chi_3, \chi_4 \sim U(\{1, \dots, \mu\}$ for each i.

5.4 Selection operator

There are two main classifications for selection according to the survival property of the parents [6]:

Extinctive_ (μ, λ) strategy; where parents live for a single generation only.$S: I^{\lambda} \to I^{\mu}$

$S(P) = P'$ where $|P| = \lambda$ & $|P'| = \mu$, & $\forall \overrightarrow{a'} \in P' : \nexists\, \vec{a} \in P - P' : f(\vec{x}) \leq f(\overrightarrow{x'})$

Preservative_$(\mu + \lambda)$ strategy; where selection operates on the joined set of parents and offspring, i.e., very fit individuals may survive indefinitely:

$S: I^{\mu+\lambda} \to I^{\mu}$

$S(P) = P'$ where $|P| = \mu + \lambda$, & $|P'| = \mu$, , and $\forall \overrightarrow{a'} \in P' : \nexists\, \vec{a} \in P - P' : f(\vec{x}) \leq f(\overrightarrow{x'})$

The ratio μ/λ is known as selection pressure. In the choice of μ and λ, there is no need to ensure that λ is exactly divisible by μ. The association of offspring to parents is made by a random selection of evenly distributed random integers from the range $[1, \mu]$. It is only necessary that λ exceeds μ by a sufficient margin that on average at least one offspring can be better than its parent. Hoffeister and Bäck in [7] have stated that $\mu/\lambda \approx \frac{1}{6}$ are tuned for a maximum rate of convergence, and as a result tend to reduce their genetic variability, i.e., the number of different alleles (specific parameter setting) in a population, as soon as they are attracted by some local optimum.

6. Cross- fertilization space of conical GAS and Standard Variant (μ^+, λ)-ES

The coupling approach followed in this section consists in the introduction of generation interval for hybrid activation operator(HAO). Through selection , crossover, and mutation operators, the simple GA works on population of bit string encoding scheme generation by generation. When HAO is active (implemented for every G? generation), the intermediate generation created by sGA is fed into adopted selection strategy which select ub population, usually of small size. Then each binary string individual in this subpopulation is converted in to a real number vector , to be the parents of the first generation of ES tool that operate on this subpopulation for a fixed small number of generations. The vectors are converted back in to bit string values to be manipulated a gain by the master GA. As ES is used here in this hybridization as a tool operator in a small number of generation fashion in an order to enhance the selected point driven from the master GA, then it is appropriate that the version of the ES too is to be of preservative survivor property to have worthy adjustment i.e., a (μ^+, λ)-ES is used.

t=0; {is the generation number}
ls=t_{max} {is the HAO age}
Initialize $P(0) = \{\vec{a}_1(0), \dots, \vec{a}_\mu(0)\} \epsilon I^\mu$ Where $I = \{0,1\}^l$
where I ∈ $\{0,1\}^l$;
Evaluate $P(0) = \{\Phi(\vec{a}_1(0)), \dots, \Phi(\vec{a}_\mu(0))\} \epsilon I^\mu$
where $\Phi(\overrightarrow{a_k}(0)) = \delta\left(f\left(\Gamma(\vec{a}_k(0))\right), P(0)\right)$;
while (τ(P(t)) ≠ true) do {while termination criterion not fulfilled}
recombine: $\vec{a}''_k(t) = r\{p_c\}(P(t))$ $\forall k \in \{1, \dots, \mu\}$;
mutate: $\vec{a}''_k(t) = m\{p_m\}(\vec{a}_k(t))$ $\forall k \in \{1, \dots, \mu\}$;
evaluate $P''(t) = \{\vec{a}''_k(t), \dots \vec{a}''_\mu(t)\}: \{\Phi(\vec{a}''_k(t)), \dots \Phi(\vec{a}''_\mu(t))\}$:
where $\Phi\overrightarrow{a}''_k(t)) = \delta(f(\Gamma(\overrightarrow{a''}_k(t))), (P(t-w))$;
if $(ls > 0\ and\ HAW = true)$do {if HAO still live and active
Select :$P_{sub} - pop(t) = \{\vec{b}_1(t), \dots \vec{b}_{\mu 1}(t)\} \in I^\mu$
Where $I = R^{n+n_{sp}}$ and
$\vec{b}_k(t) = (x_i, \sigma_j \quad \forall i \in \{1 \dots n\} \forall j \in \{1, \dots n_\sigma\}$;
{binary _coded GA individual is remapped in to $(\mu + \lambda) - ES$}
{real vector individual}
For $t_{\mu+\lambda)-ES} = 1$ to $t_{max\ \_(\mu+\lambda)-ES}$ {do tmax sexual propagation
Recombine : $\overrightarrow{b''}_k(t) = r\{rec_x, rec_\sigma\}(P(t))$ $\forall k \in \{1, \dots, \lambda\}$;
Mutate: $\overrightarrow{b''}_k(t_{(\mu+\sigma)ES}) = m_{\{\tau_0,\tau\}}\left(\vec{b}\left(t_{\mu_1+\lambda_1)_{ES}}\right)\right)$ $\forall k \in \{1, \dots, \lambda\}$
Evaluate : $P''\left(t_{(\mu+\lambda)_{ES}}\right) = \left\{\overrightarrow{b''}_1(t_{(\mu 1+\lambda 1)} - ES)\right\}, \dots, \left\{\overrightarrow{b''}_{\lambda 1}(t_{(\mu 1+\lambda 1)} - ES)\right\}$;
$\left\{\Phi\left(\overrightarrow{b''}_1 t_{(\mu 1+\lambda 1)} - ES\right)\right), \dots, \Phi(\overrightarrow{b''}_1 t_{(\mu 1+\lambda 1)} - ES))\}$
Where $\Phi\left(\overrightarrow{b''}_1 t_{(\mu 1+\lambda 1)} - ES\right) = f\left(\overrightarrow{x''}_1 t_{(\mu 1+\lambda 1)} - ES\right)$;
Select: $P\left(t_{(\mu 1+\lambda 1)} - ES^{+1}\right) = S(P\left(t_{(\mu 1+\lambda 1)-ES^{+1}} \cup P''(t_{(\mu 1+\lambda 1)-ES^{+1}}\right)$
End $(\mu 1 + \lambda 1) - ES$ generation loop
Evaluate
$P''(t) = \left(P''(t) - P_{sub-pop}(t)\right) \cup (P_{t_{max\ \_(\mu 1+\lambda 1)-ES)}}$
$= \{\overrightarrow{a''}_1(t), \dots, \overrightarrow{a''}_\mu(t)\} \in I^\mu$
Where $I = \{0,1\}^l$
and $\Phi\left(\overrightarrow{a''}_1(t)\right) = \delta\left(f\left(\Gamma\left(\overrightarrow{a''}_1(t)\right)\right), P(t-w)\right)$; re-evaluate fitness
end
Select $P(t+1) = S(P''))\epsilon I^\mu$
$ls = ls - 1$; when ls=0 then the loop is turned into pure GA phase
$t = t + 1$
end

Fig. 7. Conceptual algorithm of HGA2 (hybrid Genetic algorithm with multimember evolution strategy)

7.2 Test functions

In order to evaluate the behaviours of hybrid genetic algorithms, a set of test problem have been carefully selected to illustrate the performance of the algorithms and to indicate that it has been successful in practice. The nine test functions, which is classifies as multimodal or unimodel function; these function given with more details in section below.

7.3 Simulations

Multi- functions used as a test functions classified as unimodel and multi model it is deployed to verify the proposed hybrid genetic algorithms. The firs test function is likely hood function of ARMA(1,1) model, this function classifies as a unimodel the simulating experiment described in the following.

7.3.1 $\mathbf{F_1}$: Test function / Likelihood function

The likelihood function is one of fundamental importance in estimation theory. This principle says that the data has to tell us about the parameters contained in the likelihood function, all other aspects of the data being irrelevant. In moderate and large samples, the likelihood function will be unimodel and can be adequately approximated over a sufficiently extensive region near the maximum by a quadratic function. Hence, in these cases the log-likelihood function can be described by its maximum and its second derivatives at the maximum. The values of parameters which maximize the likelihood function, or equivalently the log-likelihood function, are called maximum likelihood (ML) estimates. The second derivatives of the log –likelihood provide measurers of "spread" of the likelihood function and can be used to calculate approximate standard errors for the estimates [ ].

Now, to study the likelihood function of $ARMA(1,1)$ let as suppose the $N = n + d$ original observations Z from a time series which can be denoted by $\mathrm{Z_{-d+1}}, .., \mathrm{Z_0 Z_1, Z_2, .., Z_n}$

we assume that this series is generated by an $ARMA(1,1)$ model. From these observations, we can generate a series w of $\mathrm{n = N - d}$ differences $w_1, w_2, .., w_n$, where $w_t = \nabla^d z_t$. The stationary mixed $ARMA(1,1)$ model in eq.7 may be written as [2]:

$$a_t = w_t - \phi_1 w_{t-1} + \theta_1 a_{t-1} \tag{18}$$

Where $E(w_t) = 0$. Suppose that $\{a_t\}$ has the normal distribution with zero mean and constant variance equal to $\sigma^2_{\mathrm{a_t}}$, then the likelihood function can get as follows [2]:

$$L = (2\pi\sigma^2)^{\frac{-n}{2}} |M^{(1,1)}|^{\frac{1}{2}} \exp\left(\frac{-S(\phi_1,\theta_1)}{2\sigma_a^2}\right. \tag{19}$$

Where $M^{(1,1)} = var - cov(\phi_1, \theta_1) = I^{-1}(\phi_1, \theta_1)$

$$= \frac{1}{I(\phi_1,\theta_1)} adj(I(\phi_1, \theta_1)) \tag{20}$$

$$I(\phi_1, \theta_1) = \frac{n}{\sigma_a^2} \begin{bmatrix} \frac{\sigma_a^2}{1-\phi_1^2} & \frac{\sigma_a^2}{1-\phi_1\theta_1} \\ \frac{\sigma_a^2}{1-\phi_1\theta_1} & \frac{\sigma_a^2}{1-\theta_1^2} \end{bmatrix} \tag{21}$$

then the log- likelihood function is:

$$\ln(\mathrm{L}) = -\frac{\mathrm{n}}{2}(2\pi\sigma_{\mathrm{a}}) + \frac{1}{2}\ln\left(\left|\mathrm{M}^{(1,1)}\right|\right) - \frac{\mathrm{S}(\phi_1,\theta_1)}{2\sigma_{\mathrm{a}}^2} \tag{22}$$

where:

$$S(\phi_1, \theta_1) = \sum_{t=-\infty}^{n} (a_t|\phi_1, \theta_1, w)^2 \quad (23)$$

is the sum squares errors,n is the sample size, and $[a_t|\phi_1, \theta_1, w] = E([a_t|\phi_1, \theta_1, w])$ denotes the expectation of a_t conditional onϕ_1, θ_1 and w. Sum squares errors can be found by unconditional calculation of the $[a]'s$ are computed recursively by taking conditional expectations in eq.13. A back-calculation provides the values $[w_{-j}], j = 0,1,2,..$
This back-forecasting needed to start off the forward recursion.
For moderate and large values of n in eq.17 is dominated by $S(\phi_1, \theta_1)/2\sigma_a^2$ and thus the contours of the unconditional sum squares function in the space of the parameters (ϕ_1, θ_1) are very nearly contours of likelihood and of log likelihood . It follows, in particular, that the parameter estimates obtained by minimizing the sum of squares in eq.17, called least square estimates will usually provide very close approximation to the (maximum likelihood estimator).

7.3.1.1 Drive formula of gradient of likelihood function

These section, we try to drive general form of steepest descent to estimate $ARMA(1,1)$ model parameters, sDA is an iterative strategy depends on the following rule for numerical computation:

$$\beta_{i-1}^{*} = \beta_{i-1} - k\overline{\nabla}e^2 \quad (24)$$

Where
β_{i-1} Parameter model
k Constant value depend
$\overline{\nabla}e^2$ is the gradient which approximate by $\overline{\nabla}e^2 = [\frac{\partial e^2}{\partial \beta_1}, \frac{\partial e^2}{\partial \beta_2}, \dots, \frac{\partial e^2}{\partial \beta_m}]$
we can see that the estimation of parameters depend on iterative algorithm which start with initial value β_i (get by one of traditional estimation methods) this algorithm continue in modified these estimators even we get the value which don't have change in values

$$FMSE = \frac{\sum_{t=1}^{n}(z_t - \hat{z}_t)^2}{n-1} \quad (25)$$

where z_t, actual value of observed time series; $\hat{z}_t$ predicted value of actual value.
We know, ARMA(1,1) model form is

$$z_t = \phi_{1t} z_{t-1} + a_t - \theta_{1t} a_{t-1} \quad (26)$$

Where a_ts are a random variable with standard normal density function known as random shock term.
then

$$a_t^2 = (z_t - \phi_{1t} w_{t-1} + \theta_{1t} a_{t-1})^2 \quad (27)$$

$(\frac{\partial a_t^2}{\partial \phi_{1t}} = -2a_t z_{t-1}, \frac{\partial a_t^2}{\partial \theta_{1t}} = -2a_t a_{t-1})$

S0

$$[\phi_{1t}^{*}, \theta_{1t}^{*}] = [\phi_{1t} + 2ka_t z_{t-1}, \theta_{1t} - 2ka_t a_{t-1}] \quad (28)$$

The value of k gets as follows

$$a_t = z_t - \phi_{1t} w_{t-1} + \theta_{1t} a_{t-1} \tag{29}$$

$$a_t^* = z_t - \phi_{1t}^* w_{t-1} + \theta_{1t} a_{t-1} \tag{30}$$

$$|\Delta a_t| = |a_t^* - a_t| = 2k a_t (z_{t-1}^2 + a_{t-1}^2)$$

$$\because 0 < |\frac{\Delta a_t}{a_t}| < 1 \quad \rightarrow 0 < 2k(z_{t-1}^2 + a_{t-1}^2) < 1 \quad \text{so}$$

$$0 < k < \frac{1}{2(z_{t-1}^2 + a_{t-1}^2)^2}$$

7.3.2 Results of likelihood function

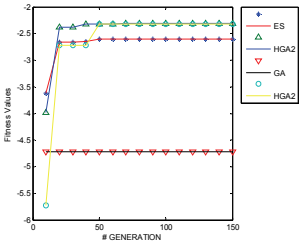
In order to evaluate the behaviour of A , ES, SDA with HGA_1 and HGA_2, we performed several experiments to test the capabilities of the methods. The results of experiments given by the following the conceptual algorithm for simple genetic algorithm and hybrid genetic algorithms adopted for the likelihood estimator of $ARMA(1,1)$. The experimental results performed here are based on different sample size (i.e.$n = 25,75,125$),(ϕ_1, θ_1) set-to $\{(0.8,0.6), (0.4,0.5), (-0.1,0.2), (-0.3, -0.4)$. The random sample are generated using Box-Muller formula which presented by using Delphi Pascal coding programming, MATLAP2008. All results obtained by running each experiment 5 different runs and each iterates with 150 generations for population size 50 and averaging the resulting data for $P_c = 0.75, P_m = 0.1$. Further, the results of (sGA, HGA) compared with those obtained by stepwise descent based on initial value computed by moment method for the same value of (parameters)(ϕ_1, θ_1) and sample size (n) (with 1000 runs). The comparison made based on Mean square error

$$MSE = var(\phi) + bias$$

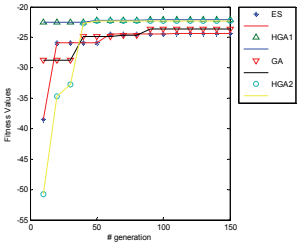
n	ϕ_1	θ_1	SDA best		sGA best		HGA_1 best		ES		HGA_2 best	
			ϕ_1	θ_1	ϕ_1	θ_1	ϕ_1	θ_1	ϕ_1	θ_1	ϕ_1	θ_1
25	0.6	0.8	1.28	1.43	0.473	0.693	0.2300	0.598	0.3668	0.3631	0.01765	0.0342
	0.4	0.5	0.74	1.01	0.517	0.51	0.276	0.419	0.1703	0.0752	0.012116	0.0312
	-0.1	0.2	0.49	0.29	0.191	0.156	0.1 02	0.139	0.1454	0.2404	0.09456	0.0104
	-0.3	-0.4	0.55	0.95	0.393	0.436	0.182	0.421	0.1351	0.2295	0.01023	0.0353
75	0.6	0.8	1.27	1.34	0.415	0.568	0.182	0.484	0.3556	0.3520	0.01198	0.0211
	0.4	0.5	0.71	0.88	0.446	0.391	0.257	0.336	0.1701	0.0751	0.01201	0.0024
	-0.1	0.2	0.21	0.15	0.157	0.154	0.054	0.093	0.1437	0.2374	0.0012	0.0065
	-0..3	-0.4	0.52	0.715	0.381	0.325	0.159	0.317	0.1336	0.2272	0.00543	0.0012
125	0.6	0.8	1.23	1.32	0.324	0.546	0.139	0.462	0.1803	0.0584	0.00156	0.0011
	0.4	0.5	0.66	0.87	0.186	0.342	0.108	0.3106	0.169	0.0746	0.009	0.0013
	-0.1	0.2	0.27	0.1237	0.137	0.136	0.031	0.0 35	0.1451	0.2401	0.0023	0.0015
	-0..3	-0.4	0.432	0.52	0.121	0.306	0.013	`0.213	0.1337	0.2274	0.00161	0.0010

Table 1. Comparisons among (GA,ES,SDA, HGA1,HGA2) algorithm based on MSE of best estimator after averaging 5 runs.

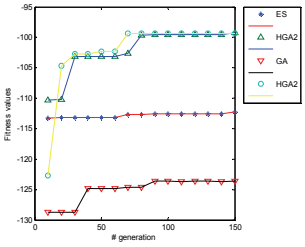
Results given in Fig. 7.and table(1).The experiments on a set of data give some impressions of the behaviours of (sGA, HGA) and sDA. As one can see that, MSE of HGA are smaller than those of steepest descent (sDA). This indicates that HGA is more reliable than HGA and sDA to give estimator of the parameters of the model under study. Moreover, one can see that value of MSE decreases as the sample size increase. For (sGA, HGA we can also see that the value of sum square decreases when increasing the number of generation and sample size. In addition, the behaviour of (HGA when the objective function parameters(ϕ_1, θ_1) take positive values are better than when they are negative. The HGA_2 algorithm was also more robust than the sGA, ES and HGA_1 performing optimally across a broad range of parameter values. In addition we can see the second best algorithm converge to best solution is HGA1.



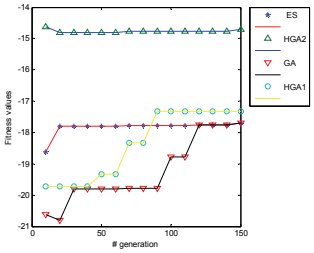
A1
Simple size
n=25,(ϕ_1=.6, θ_1=.8)



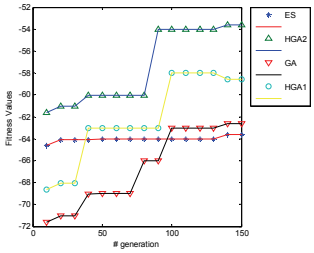
A2
Simple size
n=75,(ϕ_1=.6, θ_1=.8)



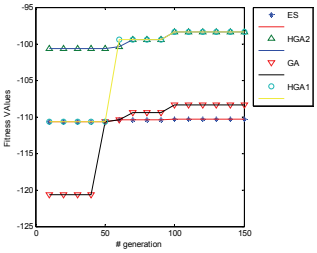
A3
Simple size
n=125,(ϕ_1=.6, θ_1=.8)



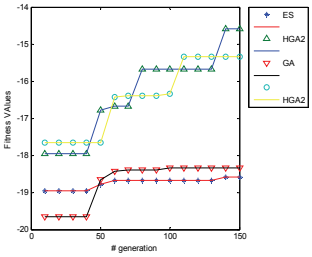
B1
Simple size
n=25,(ϕ_1=-.1, θ_1=.2)



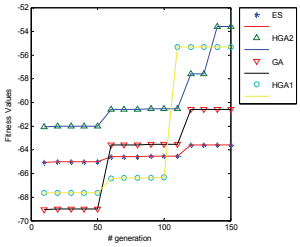
B2
Simple size
n=75,(ϕ_1=-.1, θ_1=.2)



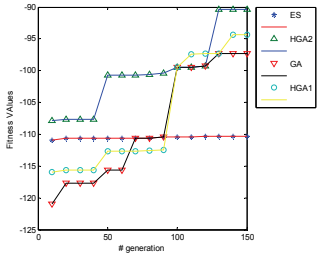
B3
Simple size
n=125,(ϕ_1=-.1, θ_1=.2)



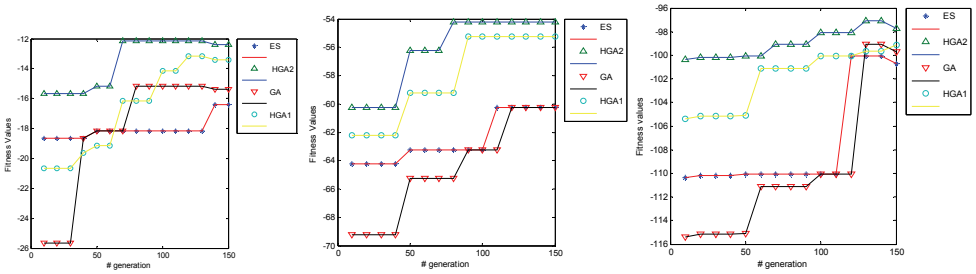
C1
Simple size
n=25,(ϕ_1=.4, θ_1=.5)



C2
Simple size
n=75,(ϕ_1=.4, θ_1=.5)



C3
Simple size
n=125,(ϕ_1=.4, θ_1=.5)



D1
Simple size
n=25,(ϕ_1=-.3, θ_1=-.4)

D2
Simple size
n=75,(ϕ_1=-.3, θ_1=-.4)

D3
Simple size
n=125,(ϕ_1=-.3, θ_1=-.4)

Fig. 8. Compression Among best fitness values respect to sample size and ARMA model parameters getting by algorithms under study

7.3.2 Benchmark test functions

The reminder of test functions using the following parameterization of the algorithms compared are used for experimental test runs

- Genetic Algorithm with population size $\mu = 200$ mutation rate $p_m = 0.1$, crossover rate $p_c = 0.75$ one-point crossover , binary code, and bit string length l =24, this algorithm denoted sGA
- Evolution strategy $((30^+, 200) - ES$ with self adaption of $n_\sigma = n$ standard deviation, no correlated mutation, local discrete recombination on object variables $rec_x = 1$, global intermediate recombination on standard deviation $rec_\sigma = 4$ and standard deviation initialize at 3.0. $(30^+, 200) - ES$ was used for unimodel functions, while $(30,200) - ES$ used for multimodal function.
- Steepest decent Algorithm(SDA): use maximum of iteration =15; with tolerance =10^{-3}. And initial size take randomly from x range .
- Hybrid Es algorithm with the algorithm GA a master and (30+200) ES as a tool for the master , for unimodel function, the best fit 30 GA individual are selected to be delivered to the (30+200)ES tool. while for multimodal functions, an evenly random selected 30 GA individuals are delivered to that ES tool.HAO is active after 3 generations of the cross-fertilization phase. HGA1
- Hybrid steepest descent with genetic algorithm Genetic Algorithm with population size $\mu = 200$ mutation rate $p_m = 0.1$, crossover rate $p_c = 0.75$ one-point crossover , binary code, and bit string length l =24, this algorithm denoted sGA, use maximum of iteration =15; with tolerance =10^{-3}. And initial solution take as the best individual from genetic algorithm for values equal (pop_size/3) from population. All results were obtained by running 5 experiments per algorithm and averaging the resulting data.

F_2: Test function

Ackley Function(multi), this function is named after Ackley who invented it

$$f(\vec{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i{}^2)}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e \qquad (31)$$

The original version was a two- dimensional function and it was later generalized to n dimension by Back. In this study form defined on n=2. The values of $\vec{x}$ defined on [-32.0,32.0]. The global minimum is located at the origin and its value is zero. The Ackley function is a nonlinear multimodal function with regularly distributed local optima

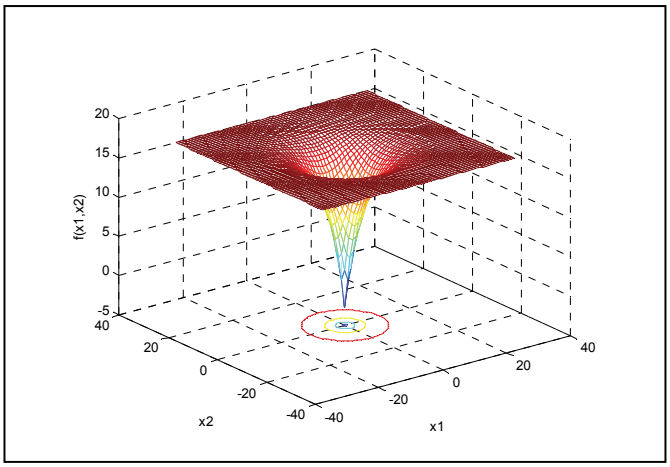


Fig. 9. F_2 Test function shape.

The results showing the ability of HGA2 to give more robust results. Also HGA1 promise to give robust results cleared in fig.16 .

F_3: ***Test Function***

$$f(x_1, x_2) = [25 - (x_1 - 5)^2 - (x_2 - 5)^2]^{\frac{1}{2}} \tag{32}$$

With constraints $\begin{matrix} -x_1^2 + 4x_2 \le 0 \\ 4x_1 - x_2^2 + 12x_2 \le 58 \\ x_1, x_2 \end{matrix}$

$$(x_1, x_2) \in [0,7]^2.$$

Optimal solution get when x_1, x_2=5.

The first set of results of this function given in table(2) for describing the SDA algorithm which is depended on experiment designed for studying the behaviour of algorithm under study, where results in table (2) explained how the gradient algorithm depend on three operators the first one is initial values x_0 generated randomly, tolerance of accuracy take equal to 10^{-3}; number of iterations (determined at the begging of experiment designed equal to 50. from results increasing number iterations when increasing variation of parameter (S), the best results get at (5.5419,5. 7225)where maximum value is(4.9178) where S=1.691, number of iterations is 6. With x_0 = (3.5,3).

When applying sGA we get the best solution get at generation 15 with X=(4.5035, 4.8091) and $F(x_1,x_2)$ = 4.9716. Hybrid of sDA with GA gave the best results at generation 95 with

x = (4.9904, 4.96268) *and* $F(x_1,x_2)$ = 4.99976 the compression among algorithms used refer to HGA2 And HGA1 to give more robust results with

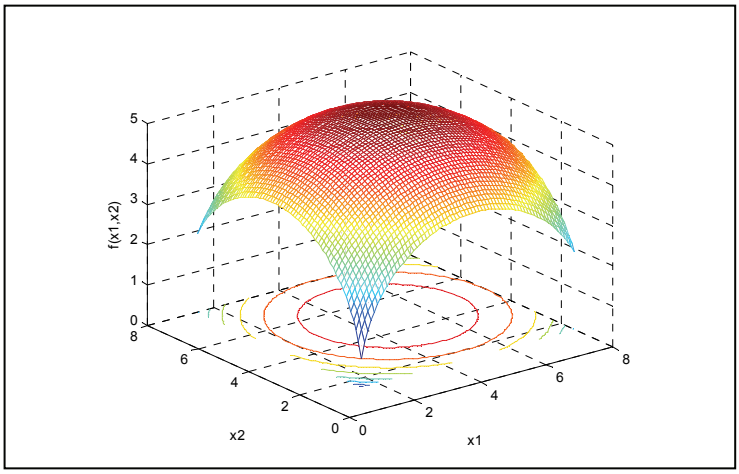


Fig. 10. F3 Test function shape.

$X_0=(x_{10},x_{20})$	Max iteration for 5 runs	s	x_1	x_2	$f(x_1,x_2)$	Min iteration for 5 runs	s	x_1	x_2	$f(x_1,x_2)$
(7 ,1)	20	2.6676	4.349	6.302	4.7834	6	2.381	4.5177	5.9647	4.8823
(5 ,4.5)	20	2.6412	5	7.1406	4.5186	2	1.839	5	6.3387	4.8175
(3.5,3)	38	2.4953	5.8697	6.1596	4.7853	6	1.691	5.5419	5.7225	4.9178
(2.5,1.2)	24	2.836	5.8308	6.2628	4.766	8	2.729	5.5737	5.872	4.8898

Table 2. Relation among number of iteration and(stepsize –s-) with initial value

F_4: Test Function

This function with multiple basins of attraction

$$f(x,y) = \frac{d_i}{r_i^2}(\bar{x}_1^2 - \bar{y}_1^2)\left(2 - \frac{\bar{x}_1^2+\bar{y}_1^2}{r_i} - d_i \qquad \bar{x}_1^2 + \bar{y}_1^2 \le r_i^2 \right. \tag{33}$$

with constraints d_i , r_i generated randomly. The test functions given in eq.29 are multi-modal functions with multiple basins of attraction. The coordinates $(x_{o,i}, y_{o,i})$ are the coordinates of the basin of attraction "i", which has random geometry (radius and depth $r_i$depth d_i), The basins of attraction for functions are randomly distributed. (Goldberg and Voessner, 1999) has conical basins of attraction represents the best case for local search, in which only one local search is required to find the local minimum.

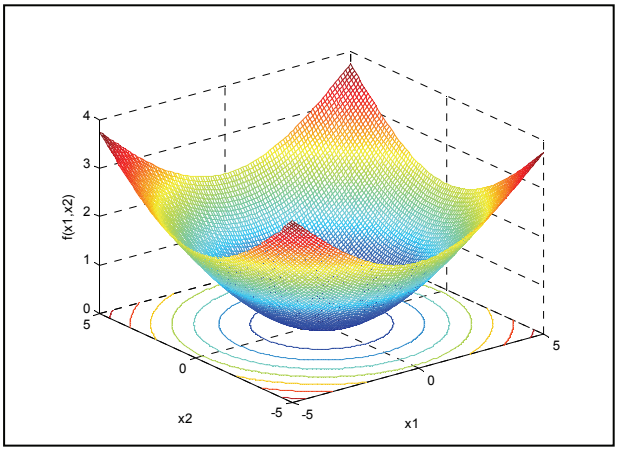


Fig. 11. F_4 Test function shape

The simulation results cleared in fig.16.

F5. Single test function

This function is a nonlinear function with single input variable

$$f(x) = e^{-2(ln2)(\frac{x-0.1}{0.8})^2}|\sin(5\pi x)| \tag{34}$$

Where $x \in [-1,1]$. Actually, this simple function has several local maximum. However, there is only one global maximum, as shown in Fig.11.

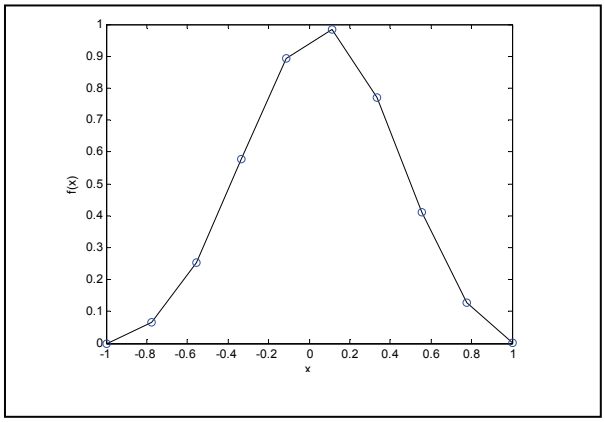


Fig. 12. F_5 Test function shape

F6: Test function

$$f(x,y) = \cos(x)^2 + \sin(y)^2 \tag{35}$$

$\vec{x}$ defined in[-5,5], this function has infinite global maximum in R^2 at points $\left(\frac{m\pi}{2}, n\pi\right), m, n = \mp 1, \mp 2 \ldots$

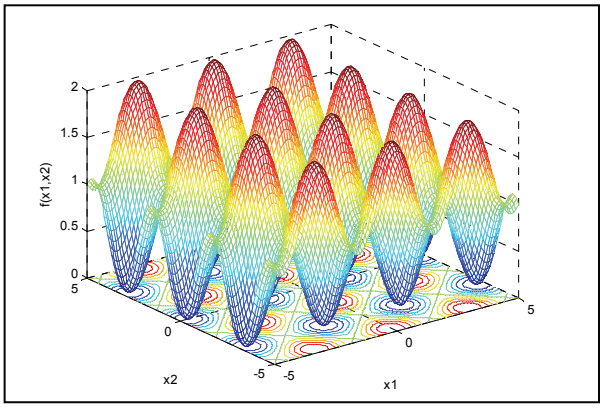


Fig. 13. F_6 Test function shape

F7: Test Function

This function known as Rosenbrock function was invented by Rosenbrock, mathematically defined as

$$f(\vec{x}) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad (36)$$

Where $\vec{x}$ is an n-dimension vector located within the range$[-30.0,30.0]^n$. the global optimum is located at (1,…,1) with a function value of zero. This function exhibit a parabolic-shaped deep valley. In the optimization literature it is considered a difficult problem due to the nonlinear interaction between variables [1].

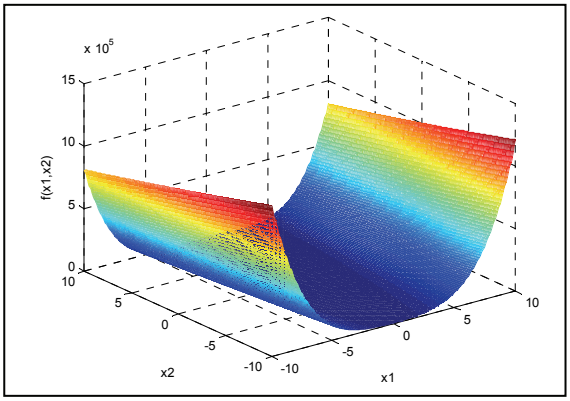


Fig. 14. F_7 Test function shape

F8: Test function

The Salmon function is rotation – invariant and was proposed by Salmon ,it is defined as

$$f(\vec{x}) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2} \quad (37)$$

Where $\vec{x}$ is an n-dimensional vector located within the range $[-100,100]$ the global optimum located at the origin with a function value of zero.

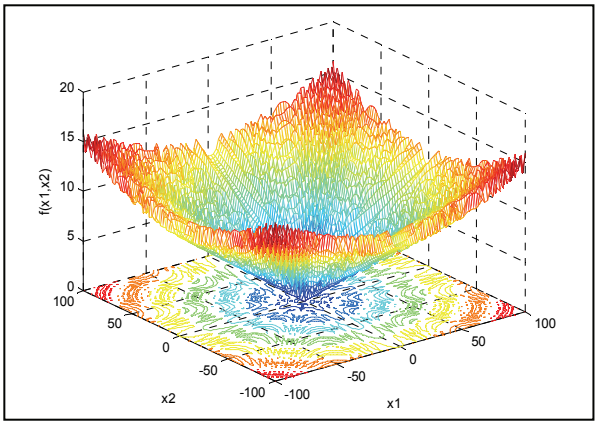


Fig. 15. F_8 Test function shape

F9. Test function

This function also known as Schaffer's function or the sine envelope sine wave. Mathematically define as

$$f(\vec{x}) = 0.5 + \frac{\sin^2(\sqrt{x_1^2+x_2^2})}{(1+0.001(x_1^2+x_2^2))^2} \tag{38}$$

Where $\vec{x}$ is a two-dimension vector located within the range $[[-100.0,100.0]^n$. The global optimal is located at the origin with a function value equal to zero.

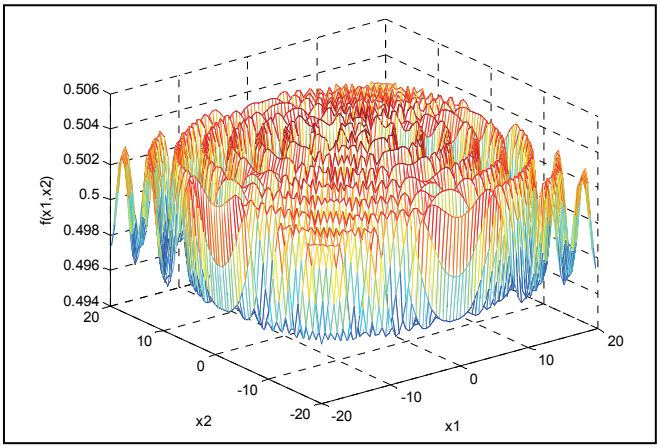


Fig. 16. F_9 Test function shape

F10 : Esom function

This function was proposed by Easom to evaluate global optimization techniques. It is n-dimensional function with single minimum that is also the global optimum. The mathematical expression of this function is

$$f(\vec{x}) = -\prod_{i=1}^{n} \cos(x_i) \cdot e^{-\sum_{i=1}^{n}(x_i-\pi)^2} \tag{39}$$

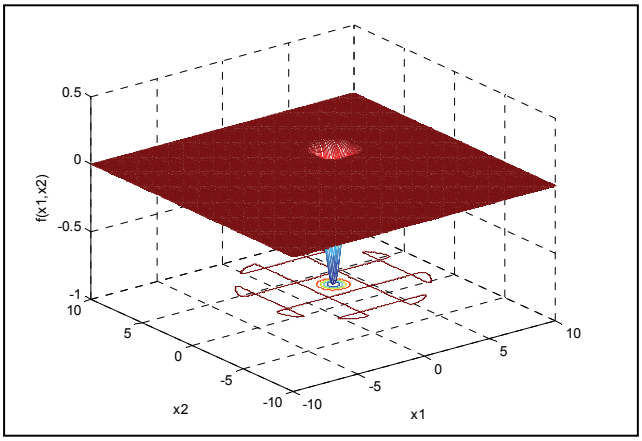
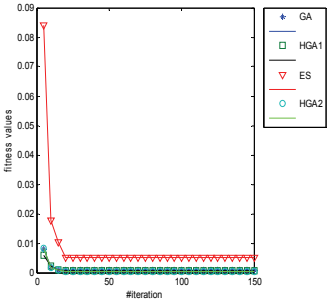
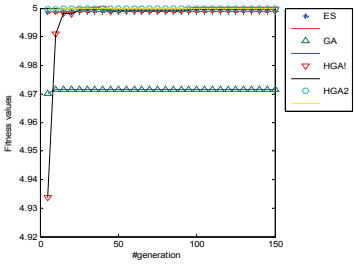


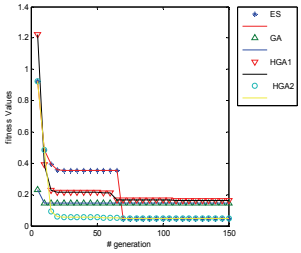
Fig. 17. F_9 Test function shape



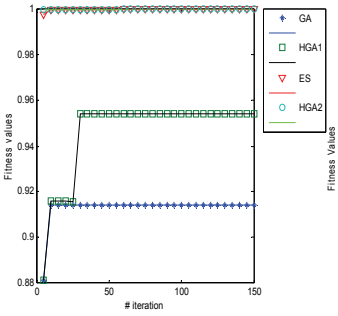
F_2 Test Function Comparison



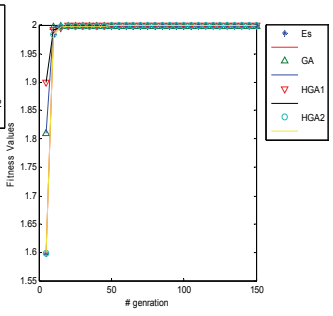
F_3 Test Function Comparison



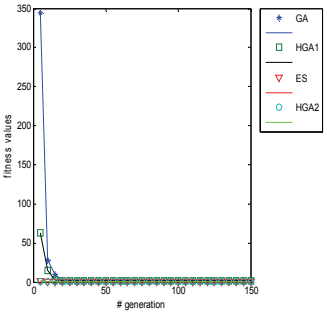
F_4 Test Function Comparison



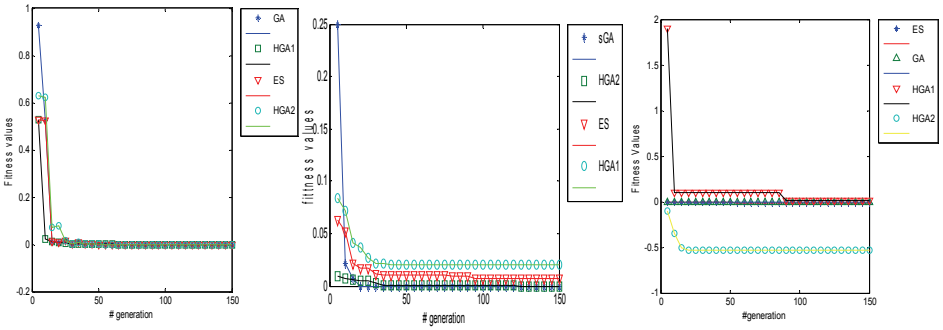
F_5 Test Function Comparison



F_6 Test Function Comparison



F_7 Test Function Comparison



F_8 Test Function Comparison F_9 Test Function Comparison F_{10} Test Function Comparison

Fig. 18. Compression among fitness values respect to number of generations for benchmark test functions

7.3.3 General results getting from designing experiments

According to the results presented above, a general trend will be drawn about the course of actions of the competent algorithms. Results show that sGA alone with its bit level crossover and mutation operators can act as a heuristic for exploration with somewhat little emphasis on search focus.

Sample GA showed to be trapped by local plateaus. One could return this behavior to the main distinguished operator of the master GA, the one point crossover operator. one can easily see that the canonical GA is the slowest of the algorithms under study. The behavior of GA is almost identical on all the unimodel functions.

The collective nature of GA tournament selection, one point crossover, and mutation operators give a clear demonstration of its missing emphasis on the convergence and local optimization. On the other hand , the ES with self adaption of $n_\sigma = n$ standard deviation is , on overall, the faster by far and its results are superior to that obtained from (GA,HGA1). The combination of self-adaption , recombination , and relatively strong selective pressure as used in ES algorithm. The nature of the preservative survival of the best individual implied by the plus selection strategy. Also, the self adaption role of the strategy parameters through intermediate recombination and mutation is shown to be fascinating. Even if all the parents start with equal $\sigma_i = \sigma = 3.0\ \forall i = 1, \dots, n_\sigma$, and all the step length components are varied by a common random factor in the production of the offspring , the σ_i of all individuals will differ from each other in the subsequent generations through self adaption. So in this way a better combination affords a higher chance of survival to its bearer. It can therefore be expected that in the course of the optimum search, the currently best combination of the $\{\sigma_i; \forall i = 1, .., n_\sigma\}$prevails.

The HGA2 reduces the speed gap between the canonical GA and standard variant ESs convergence , it does not outperformed ES with both its, variants, except for some cases . The deviation in convergence velocity of the HGA2 from ES variants can be attributed to the fact that although in the first cross-fertilization phase of HGA2, the best GA individuals are enhanced by the coupling EA algorithm, the exploration power of the master GA still remained an order of magnitude. A closer look is given here to compare the behavior of ES and HGA2 one hand , and HGA2 with HGA1,GA, SDA. comparing the results of the overall

hybridization established in HGA1 with the hybridization of HGA2, one could see that the presented results of HGA2 are more powerful than that of HGA1.

8. Conclusions

This chapter is devoted to global optimization algorithms, which are methods to find optimal solutions for given problems. It especially focuses on two major groups of optimization algorithms evolutionary computation by discussing evolutionary algorithms, genetic algorithms, evolution strategy. Second group represent by hybrid algorithms which are coupling simple GA with local algorithm steepest descent algorithm(HGA1) and GA with self adaptive global algorithm evolution strategy (HGA2). The results , depending on the standard functions presented in the test suite, it campers the performance of (sGA, (μ^+,λ)-ES, SDA, HGA1,HGA2) algorithms. The simulating experiments designed for sets of benchmark test functions classifies as unimodel and multimodal, Four unimodel functions, the hybridization was found to be advantageous for speeding up the performance of the canonical GA so the speed gap difference between the very general purpose optimizer algorithms as canonical GA and the specialized parametric optimization algorithm as multimember ES is diminished . Also, the hybridization was found to be beneficial in multimodal functions where convergence reliability is of interest. By taking the advantages of both exploration power of the GA and the exploitation power of the multimember ES, the HGA introduces more reliable solutions than GA or ES when worked individually.

9. References

A.Montes Marco; Rolda'n Oca (2006).On the optimization of Particle Swarm Optimization, Annee Academique

W. W. S. Wei, 1990, Time Series Analysis: Univariate and Multivariate Methods, Wesley Publishing com. INC, New York.

G. E. B. Box, and G. M. Jenkins(1976).Time Series Analyses is: forecasting and control", Holden-Day, San Francisco.

S. Makridakis, S. C. Wheelwright (1999). Forecasting Method and Application, John Wiley& sons, New York.

T. Bäck, and H,-P, Schwefel, 1993, An overview of evolutionary algorithms for parameter optimizations", Evolutionary Computation, Cambridge, Vol. 1, No. 1, pp. 1-23,.

T. Bäck, and H,-P, Schwefel (1995). Evolution strategies I: variants and their computational implementation", Genetic Algorithms in Engineering and Computer Science, pp. 11-26.

H,-P, Schwefel(1981). Numerical Optimization of Computer Models. New York: John Wiley & Sons,.

F.Hoffmeister, and T.Back, "Genetic Algorithms and Evolutionary Strategies: Similarities and Differences", Parallel Problem Solving from Nature PPSN I,. Vol. 496, pp.455-

B.A.Attea, B.A.Husssain (2006). An evolution strategy for likelihood Estimators of ARMA (1, 1) Mode The 4th International Multiconference and Information Technology CSIT2006 Vol.1, Amman-Jordan.

B.A.Hussain, R.D.AL.Dabbagh (2007). A Conical Genetic Algorithm for Likelihood Estimation of first order Moving Average Model parameter. Neural Network World Vol.4, pp(271-285).

Adrian E. Drake, Robert E. Marks, Genetic algorithm in economic and Finance: forecasting Stock Market Prices and Foreign Exchange, A Review, Internet

T. Smith Robert, B. Minton Roland (2002). Multivariable Calculus, second edition, McGraw-Hill.

Thomas Weise (2009). Global Optimization Algorithms– Theory and Application – ;Version: 2009-06-26; Newest Version: http://www.it-weise.de/.

Attea, Bara'a Ali (2001). Interdigitation : Hybrid of Genetic Algorithms and Evolution Strategies for Parametric Optimization; Doctoral theses, Iraq/ Baghdad.

Hussain . Basad Ali (2002) Comparison Among Forecasting Methods of Markov and Mixed Model by Using Simulation; Master thesis; Iraq Baghdad.

Rabunal . Juan R, Julian Dorado(2006).Artificial Neural Network in Real-Life Applications, Idea Group Publishing,.

X. Wang, X.Z. Gao and S.J. Ovaska (2008). A Novel Particle Swarm – based Method for Nonlinear Function Optimization, International journal of Computational Intelligence Research,Vol.4, No.3, pp.281-289.

Guoli Zhang, Haiyan Lu(2006). Hybrid Real coded Genetic Algorithm with Quasi-Simplex Technique ,IJCSNS International Journal of computer Science and Network Security, Vol.6 No.10,October .

13

Tracing Engineering Evolution with Evolutionary Algorithms

Tino Stanković, Kalman Žiha and Dorian Marjanović
University of Zagreb/Faculty of Mechanical Engineering and Naval Architecture
Croatia

1. Introduction

The mankind achieved an astonishing technological development through centuries of innovation, creation and continuous improvement. The history of engineering is the inherent component of the civilization. Moreover, outstandingly important lessons for further development can be studied in the history of engineering. The investigations of the recent complex engineering knowledge, experience, analytical and computational tools may serve to explain the technical progress and facilitate the future development. For this purpose this chapter will present how the evolutionary algorithms can simulate the developing complexity of engineering reasoning that in reverse can back-trace the primitive origins of modern technical products. The chapter will resume the evolutionary algorithms as well as the evolutionary optimization and design processes based on innovative and creative activities with the aim to define their potentialities in discovering the evolution of engineering products. More so when adding to the whole process the touch of randomness introduced in form of mutation operator the algorithm gains the property to converge to the global optimum within multi-modal search space. Both of these processes crossover and mutation have been present in the natural evolution for eons of time. From an algorithmic perspective crossover and mutation enable adaptation of the population of feasible solutions to the imposed environment conditions of the search spaces. The chapter will concentrate on the multi-objective optimization problems taking for example the NSGA-II algorithm (Deb, 2001). The result will be obtained as the population of optimal solutions distributed along the Pareto frontier. Using constraint domination condition and constrained tournament selection operator the evolution of the object under consideration will be explained. Normally engineering relies on the design process that is for technical purposes modelled as a set of cyclic activities put in a logical order to guide the procedure until the desired technical aim is reached. The design process is comprehensible as a shortcut to a satisfying product that is also in clear correlation with the formulation of an algorithm, particularly with evolutionary algorithm. The evolutionary methods affect design process and teach about process itself. They stimulate innovation and creativity during the human efforts to design and apply processes which are all in reality an attempt to produce an unbiased human performed heuristic search. The evolutionary design relies on the normal contemporary progressive engineering reasoning aspired with achievement of highly efficient products providing appropriate safety levels by employing genetic algorithms. The chapter will consider how the reverse process to the technical progress can reconstruct the origins of contemporary products using evolutionary

algorithms on engineering models in two manners. Chapter will present two examples. The first case study in this chapter investigates the evolution of a truss structure possibly in the future up to the very important engineering entity – the wheel. The second case study back-traces the development of the stiffened shells in aerospace and shipbuilding industry to their primitive origins in boats originally made of carved-out log.

2. Evolutionary Algorithms

C. Darwin wrote that evolution begins with the inheritance of the gene variations. Inspired by the natural evolution in the mid 20th century the field of evolutionary computation emerged to its dawn and new class of algorithms was born. By the principle of the survival of the fittest, solution or set of solutions to given problem evolve in time using fitness - objective function that is, as an evolutionary guide. During the search new solutions are generated by reusing and mixing together pieces of the past solutions. Like with the living organisms – information in digital computer comprehensive manner was being exchanged between the most feasible solutions. Important class of evolutionary algorithms, genetic algorithms resembled natural evolution in the most (Yokey, 2005). Information exchange between solutions was done by exchanging binary number strings by crossover operators. Information chunk exchange was described in well known Building Block Hypothesis (Goldberg, 1989). Goldberg's attempt of proving the convergence of heuristic genetic algorithm is in line with the mid 20th century genetics theories. It is known that information exchange during forming of amino acids is also linear and digital like in computers and it is build from chunks of information (Yokey, 2005). Since they were not calculus based application of such genetic algorithms was soon to be recognized as they were applied as general optimization problem solvers. Range of applications included combinatorics (scheduling, TSP problem, close packing problems), various engineering optimization problems (single or multy-objective optimization), neural network trainers etc.

Evolutionary algorithms own their properties and behavior to the process that they are trying to mimic in order to find solution - the natural evolution of living organisms. The solution or the set of solutions to the given problem evolves in time from the feasible solution population by the principle of the survival of the fittest – selection operator, with the fitness function acting as the evolutionary guide. The discreteness of algorithm is devised from its crossover operators, which when generating new solutions, are reusing and mixing together pieces of the past solutions making it very useful when dealing with non continuous problems. Such usage of the past knowledge described by Goldberg in the Building Block Hypothesis (Goldberg, 1989) gives to the algorithm property to converge to desired better solution to a given problem, which ultimately distinguish it from the plain random walk algorithms. More so when adding to the whole process the touch of randomness introduced in the form of mutation operator, the algorithm gains the property to avoid the pitfalls of local optima. Both of these processes, crossover and mutation, have been present in the natural evolution for eons of time. From an algorithms perspective crossover and mutation enable adaptation of the population of feasible solutions to the imposed environment conditions of the search spaces. The recent 15 years have presented a significant number of methods and tools (Goldberg, 2002) for application in engineering.

The general multi-objective optimization problem is tackled by the NSGA-II algorithm (Deb, 2001) that is implemented as a dynamic-link library in C# within Microsoft .NET Framework 2.0. to provide a generic multi-objective solver for various optimization models

in engineering. NSGA-II algorithm generates populations of optimal solutions distributed along the Pareto frontier, using constraint domination condition and constrained tournament selection operator (Deb, 2001). Normally the design process is structured as a set of cyclic activities put in a logical order to control and guide the procedure until the desired aim is reached (Goldberg, 2002).

3. Evolutionary optimization

Evolutionary algorithms (EA) have been used as a general optimization tool for technical systems ranging from general single objective benchmark optimization cases, such as Golinski's problem (Golinski, 1970), to multi-objective genetic algorithms (Tan, Lee&Khor, 2002), (Deb, 2001). Evolutionary algorithms provide a solution or a set of solutions of optimization problems in discrete time using the fitness function and the simple prime mechanisms, crossover, mutation and selection. The crossover operator is in fact a great recombination machine that combines bits and pieces of past solutions into a new sequence. The binary encoded genotype in the form of binary string sequences is shuffled throughout the evolution. Applicability as a discrete problem solver of the genetic algorithm is a direct result of the crossover operator. With the introduction of a reasonably small degree of randomness in the form of a bit-flip mutation operator, the algorithm gains the property to avoid pitfalls of local optima. Both of these processes have been present in the natural evolution for. Crossover and mutation enable the adaptation of the feasible solution population to the imposed environmental conditions of the search spaces. Selection, the third operator, acts as a collective learning enforcer, which will ruthlessly guide the evolution by means of the survival of the fittest. Fitness merit is assigned after the evaluation of the objective function to each and every population member. The collective learning process and a possibility to impose the search strategy distinguish GAs from the plain random walk algorithms (Bäck&Fogel, 2000). The research of evolutionary computation-based tools for enhancing the design optimization is therefore reasonable and justified because real engineering design search spaces are often multimodal, full of discontinuities and constrained. The range of applications (Bäck&Fogel, 2000) included planning (scheduling, TSP problem, close packing problems), simulations (behavior prediction), recognition, and control (adaptation and evolvable hardware).

4. Evolutionary design

The design process is comprehended in this text as a shortcut to a satisfying product using general and personalized knowledge and experience of design modeling in order to accelerate the technical development which naturally should occur evolutionary in spacio-temporal and social circumstances. The design process can be put in correlation with the formulation of an algorithm as an iterative problem solving procedure involving a finite number of steps. One could define such a procedure as a search algorithm where the search space itself is built on lists of requirements or design variables and constraints – the problem or design task formulation, and the search for the feasible solution is being conducted by iteration, abstraction, concretization and improvement (Goldberg, 2002). All of these four processes are built in core of an evolutionary algorithm. They are iterative – searching for solution during each new generation, abstracting – a common practice in multi-objective optimization where the objectives are put in order by degree of importance and evaluated

respectively (Bäck&Fogel, 2000), concretizing – in order not to hinder the process the objectives can be introduced at a desired point in evolution when solutions are evolved enough, improving – by evolving solutions in every generation using selection, crossover and mutation operators. The evolutionary methods may provide enhancement of design process or findings about process itself. Properties of search spaces will depend on complexity of the design aim and could be constrained, multimodal and full of discontinuities. Many applications of evolutionary algorithms in search spaces have been recognized (Bentley, 1999), (Golinski, 1970), (Tan, Lee&Khor, 2002). Various methods enhance design innovation and creativity such as Delphi method, 635 method and synectics (Pahl&Beitz, 1988), (Wood&Otto, 1999), or brainstorming that support an unbiased human search for technical solutions. The evolutionary algorithms for this purpose use the form of mutation operator which stochastically alters feasible solutions. It can be hypothesized that in order to produce innovative solutions a design process as well as the natural evolution should be performed without a bias. By using evolutionary design the designer is shaping and adjusting his designs enabling their existence in constraint bounded design space simlarly to the principles recurring in natural evolution.

5. Evolution of truss structures

In the realm of genetics, natural evolution and information theory, the problem of information encoding and decoding is permanently reconsidered (Yockney, 2005). A common point of all three mentioned researched areas is how to design a specific encoding of sequences carrying information, or how to decode them properly from a noisy environment to an error-free state. Genotype encoding and its counterpart, decoding into a phenotype, present a special point of interest in the evolutionary computation community (Goldberg, 2002) and (Bentley, 1999). Notions regarding overall convergence and phenomena that occur during the information exchange between chromosomes were first tackled in Goldberg's famous "building-block hypothesis". Explored further by the same author (Goldberg, 1989), an attempt was made to evaluate the quality of binary string building blocks when used in different classes of problems. Evolutionary algorithms are considered to be robust due to both their operators and their easy customization to suite different areas of application. However, to accomplish robustness specific to encoding, decoding and phenotype representations must be created. To found out the right answers to the addressed problem, evolutionary algorithms must have a good material to work with. In the paper, a new encoding/decoding scheme will be presented. The results point out that the scheme is well suited for the structural optimization of truss structures.

The applicability of evolutionary and genetic algorithms as optimization tools is elaborated in the next case study. Then, a state-of-the-art overview of the truss structure optimization and research motivation for using advanced methods is presented. Drawbacks of each method, resulting from different problem approaches are also addressed. The pseudo-code of the proposed encoding for the 2-D continuous domain is presented.

The aim of the research is to move away from the orthodox structural continuous optimization approach, where most of topology is initially fully predefined or locally constrained (Hasançeb, 2007), (Coello&Christiasen, 2000). In such a way, the search space is reduced, thus inhibiting evolution to progress towards new uncovered solutions. Such genotypes are easily coded because one can predict nodal inter-arrangements. They do not cover the possibilities of the initial randomness of the structure shape; instead, they optimize the usual truss bound design variables (cross-sectional area, length etc.). By contrast, a different type of coding is

proposed in the topological optimum design (TOD) (Jakiela at all, 2000), (Hamda&Schoenauer, 2002) and (Kim&Weck, 2004). The structure is represented in a discrete domain, in the form of material distribution, which is a straightforward approach to the shape optimization. The genotype encodings are done through matrices, Voronoi representations (Hamda&Schoenauer, 2002), or even by using 3-D FEM building blocks. In the same manner, the phenotype representation then visually depicts the resulting structure. An interesting cantilever optimization problem has been elaborated by Kim and de Weck (Kim&Weck, 2004), who addressed the quality of search with the chromosomal length (Goldberg, 1989). They increased the domain resolution throughout the evolution course. By taking this approach, they (Kim&Weck, 2004) also addressed the design concretization defined in literature (Hubka, 1992), (Pahl&Beitz, 1988) as progression from abstract to concrete through the design process stages. Although TOD is computationally demanding, it optimizes the structure in the form of the in-domain material distribution, but the other design variables, such as the cross-sectional area, remain predefined or out of reach. A more subtle approach using the the shape annealing (SA) method and shape grammars for structural optimization purposes was proposed by (Shea&Cagan, 1998). Shape grammars, as their linguistic fundament (Chomsky, 1957), provide language and in this case a design language for the structure shape manipulation. They are driven by a simple set of production IF-THEN rules. The evolution begins with the initial structural member expanding and growing slowly through the rule implementation to a complete structure. However, to search all possible truss structures that can be constructed within a search domain in order to obtain a global optimum, than the rule set must be adequate to enable the emergence of such solutions within a design language. So for generic approach, to evolve structures one should be able to evolve the rules (Gero&Louis, 1995).
The genotype encoding and decoding should enable the search space to be as large and unconstrained as possible. Genotype are a collection of binary encoded nodes and the phenotype represented as truss structure is then defined as a result of the inter-nodal arrangement. Although this idea is straightforward, it has not been explored so far. The problem in the 2-D continuous domain is depicted in Fig. 1.

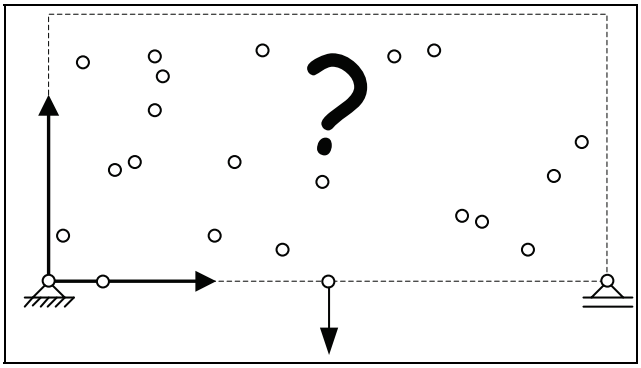


Fig. 1. Random nodal arrangement in 2-D continuous domain - predefined nodes in supports and in force node

There is no problem for the genetic operators to function properly for an ordinary binary encoded string, but getting a structure out of such genotype encoding posses a more serious difficulty.

There exist a number of connections between nodes as the ones presented in Fig. 1. In fact, the number of combination is enormous and it strongly depends on what is tried to be accomplished. How to connect 2-D collection of nodes (for a prototype planar collection of nodes see Fig. 1) with the resulting truss structure at the phenotype level is elaborated in the following chapter.

5.1 Proposed coding scheme for 2-D continuous domain

Genotype and phenotype representation coding schemes are proposed with some restrictions and are based on the following assumptions:

- Firstly, the structural model is defined as a FEM model. In each evolution turn for every new population member the system stiffness matrix is re-assembled. In the matrix, truss (rod) elements are defined as a consequence of the inter-nodal arrangements. Hence, a new structural model is to be generated according to the current system topology.
- Secondly, to use common knowledge in the design of truss structures and to avoid possible singularities of the system stiffness matrix, the system components are always arranged in triangular schemes (see Fig. 2). To clarify, it is a widely known fact that the triangle substructure presents a stiff building block common in engineering. Such a restriction narrows the search space and enhances the conversion of the algorithm by eliminating the known unfeasible and mathematically singular problems. By following the described procedure the result would not always be a structure bounded by a convex polygon.
- And finally, there exist a number of predefined nodes. These nodes have defined positions in a given 2-D domain. Predefined nodes are always supports and nodes with force vector (see Fig. 1). The total number of fixed nodes is given by *NoNx*, and the total number of free nodes is given by *NoN*.

5.1.1 Genotype encoding

Genotype is encoded with binary strings. Nodes are coded as shown in Table 1. Chromosome is then represented as a collection of nodes (see Table 2.). All of the genetic operators are easily applicable to such coding.

$Node_j$	
x coordinate - binary encoded string *lj*	*y* coordinate - binary encoded string *lj*

Table 1. Encoding of node

Chromosomes					
Node1	*Node2*	*,…,*	*Nodej*	*,…,*	*Noden* (*n = NoN+NoNx*)

Table 2. Chromosomes structures

Crossover is made by randomly selecting a crossover point among nodes on the chromosomal level (see Table 2), and then by selecting another crossover point on the nodal level (see Table 1). Mutation is performed easily, in a bit-flip manner, directly altering the nodal position.

5.1.2 Genotype decoding and phenotype representation

For every evaluation during the evolution, a genotype must be decoded into a phenotype. The algorithm for getting a phenotype, i.e. defining the rods and truss structure, based on the nodal positions is presented in the pseudo-code below:

```
1 sort nodes ascending over x, if equal compare over y
2 move to first node in chromosome j ← 0
3 while (j < n - 2) do
  A. if node(j).y >= node(j + 1).y
     for rod FEMs definition consider nodes(j + 1, j + 2 ,…, j + k) that are ordered
     ascending over y do
      break the search if node(j + k + 1):
        a. is not ordered ascending over y
        b. is second node in ascending order satisfying node(j + k + 1).y > node(j).y
        c. j + k + 1 > n
     od
  B. else: do the same as in A but considering the descending order of nodes
  C. define rod FEMs between node(j) and all found nodes within A or B
  D. move to next node j ← j + 1
4 od
```

The algorithm starts with all of the nodes being sorted ascending based on their x coordinate. From collection of nodes the first node is taken into consideration by setting counter j to zero (pseudo-code line 2). In the following *while loop* marked by number 3, the algorithm will search for possible ways to define FEM rod elements between considered node and all the nodes having greater x coordinate. Resulting structure must be triangular with no FEM elements intersections. Inside the loop two possibilities exist (marked with letters A and B); based on its position the first following node can be below or on equal height (A) or above the considered node j (B). Inside A all of the nodes will be ranked feasible for FEM definition if they do not violate the conditions inside a, b and c. Condition a takes into an account weather all of the following $j + k + 1$ nodes are in ascending order over y, b breaks the search if the node is the second one above the node j and c prevents the counter being larger the overall collection of nodes n. B takes into account situation opposite of A - the first following node being above node j thus considering the decreasing order over y. Afterwards the FEMs are defined and whole procedure is repeated for node $j + 1$.

The singular conditions that occur when two or more nodes occupy same position are regulated with general constraints. The results of inversing the order of sorting in the way that the sorting is first conducted over the y and then over the x node coordinate (pseudo-code line 1), were not explored in this paper. Besides for the reasons of common practice, the procedure runs first over x and then over y. On Fig. 2, a truss structure phenotype is depicted corresponding to the nodal arrangement already shown in Fig. 1.

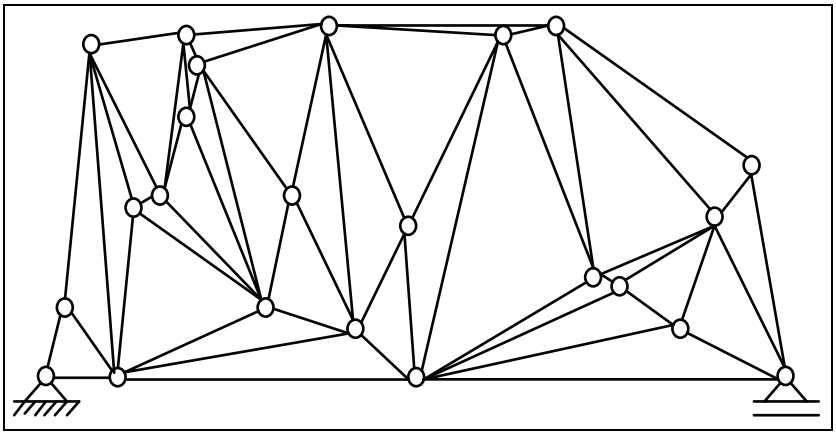


Fig. 2. Truss structure obtained from nodal positions shown in Fig. 1

5.2 Structural FEM model

The structure is modeled with FEM planar trusses with 6 degrees of freedom. It is necessary to introduce bending to trusses and implicitly convert them into beams. The result of the evolution with infinite stiffness to bending will always converge to a single horizontal rod. Such structure would have zero displacement since it cannot bend, it would be minimal in mass since it is just a horizontal line. Normal forces would also be equal to zero if the force vector is put vertically as in Fig. 3-5.

Respective force F and displacement δ vectors per element are given as follows:

$$\{F\} = \begin{Bmatrix} N_1 \\ Q_1 \\ M_1 \\ N_2 \\ Q_2 \\ M_2 \end{Bmatrix} \quad \{\delta\} = \begin{Bmatrix} u_1 \\ w_1 \\ \varphi_1 \\ u_2 \\ w_2 \\ \varphi_2 \end{Bmatrix}$$

The element stiffness matrix $\boldsymbol{K}$ is common (Zienkiewicz, 1971) and is given here by:

$$[K] = \begin{bmatrix} \frac{EA}{l} & & & -\frac{EA}{l} & & \\ & \frac{12EI}{l^3} & \frac{6EI}{l^2} & & -\frac{12EI}{l^3} & \frac{6EI}{l^2} \\ & & \frac{4EI}{l} & & -\frac{6EI}{l^2} & \frac{2EI}{l} \\ & & & \frac{EA}{l} & & \\ & & & & \frac{12EI}{l^3} & -\frac{6EI}{l^2} \\ \textit{symetrical} & & & & & \frac{4EI}{l} \end{bmatrix}$$

The vector of the nodal displacements u of the evolved structures is calculated by solving the usual linear equation system given by:

$$\{F\} = [K]\{u\}$$

5.3 Optimization model

In this optimization case the goal is to find an optimal distribution of trusses that comprise truss structure for a given 2-D domain. The result will be a structure that is a result of its interaction with the environment - the objective function, design space conditions and constraints. For the reasons of simplicity a number of involved nodes are given by the user as a process input parameter. Since the algorithm uses fixed length chromosomes nodes cannot extinct during the course of evolution.

Next, all the trusses have the same fixed cross-section area therefore significantly reducing the search space. These parameters will be introduced as variables in the future work. Finally, the optimization problem is formulated as follows:

$$\left.\begin{aligned} &\min\left[m(F, A, I, E, NoN, NoNx, BC)\right] \\ &subject\ to: \\ &\delta \le \delta_{\max} \\ &l \ge l_{\min} \\ &\sigma \le \sigma_{\max} \\ &x \in \left[x_{\min}, x_{\max}\right] \\ &y \in \left[y_{\min}, y_{\max}\right] \end{aligned}\right\}$$

The objective function is the minimization of structure mass m.

The optimization parameters are given as follows:

- F – force vector in vertical direction,
- A, I, E – truss cross-section properties and material Young's modulus,
- NoN – number of free nodes,
- $NoNx$ – number of fixed nodes,
- BC – boundary conditions – number and type of supports at particular nodes.

Problem variables:

- x and y coordinates of each node considered,
- δ – absolute deflection vector,
- l – length of respective rod.

Constraints are defined as the maximally allowable absolute nodal deflection δ_{max} and the minimally allowable beam length l_{min}. The domain or design space is defined as a 2-D bounding box.

5.3.1 Implementation, control parameters and evolutionary operators

The applied genetic algorithm is a struggle genetic algorithm. Reasons for applying this particular GA lay in its elitist steady-state evolution. Only the best solution from the offspring population replaces the closest solution from the parent population if it is better. The distance between solutions is measured in the Euclidian objective space. The struggle GA manages to maintain diversity during the evolution process, thus preventing premature convergence. The control parameters of algorithm (Bäck&Fogel, 2000) are as follows:

- population size $\lambda = 60$,
- offspring population $\lambda = \lambda$,
- one point crossover probability $p_c = 1$,
- bit-flip mutation probability when static $p_m'' = 0.02$.

Mutation operator was introduced to help guide and boost the process in order to avoid local optima in the early stages of evolution. In literature, there exist a number of such mutation approaches (Goldberg, 1989), (Bentley, 1999), (Deb, 2001) and (Bäck&Fogel, 2000). This one is driven by the notions from the natural evolution, where initial mutation rates were much higher because of the imposed environmental conditions. Bit flip mutation rate is a function of its initial rate $p_m' = 0.1$ and the number of evolution iteration N. Mutation rate is simply linearly scaled over a desired number of iterations by the following formula:

$$p_m = \begin{cases} \frac{p_m'' - p_m'}{1000} N + p_m' & N < 1000 \\ p_m'' & N \geq 1000 \end{cases}$$

Constraint handling is done by measuring how potential solutions violate constraints. Two populations of feasible and unfeasible solutions were ranked accordingly; the feasible one sorted ascending over the objective function and the latter sorted in the same manner but over the violation measure. Constraint violation measure $\Omega(x^{(i)})$ (Deb, 2001) of *i-th* solution $x^{(i)}$ is derived as the summation of normalized violations $\omega_j(x^{(i)})$:

$$\Omega(x^{(i)}) = \sum_{j=1}^{3} R_j \omega_j(x^{(i)})$$

No violations were favored so the weighting factor used is R=1 for all constraints. Every chromosome is a collection of n nodes. For encoding of the nodal x and y coordinates, binary strings were used. In addition to the refinement of the search, the Gray coding was applied (Bäck&Fogel, 2000).

5.4 Results

The results were obtained after roughly 1000 iterations in each of the presented examples. The evolution was conducted on a PC with the AMD Athlon 64 X2 5000+ processor. For the sake of further research, object-based dynamic-link libraries were designed in C# (MS .NET Framework 2.0) to provide a generic multi-objective solver for various engineering optimization models. The results of the optimal truss structure on the following three figures (Fig. 3-5) present the course of evolution under different initial conditions. The number of free nodes is increased as the load is increased in the force node.

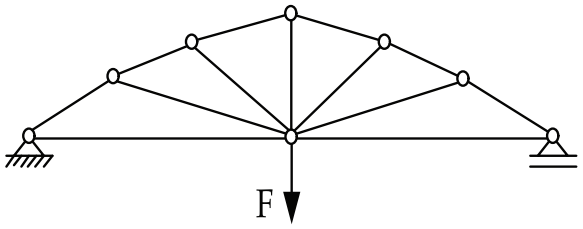


Fig. 3. Optimal structure with 5 free nodes

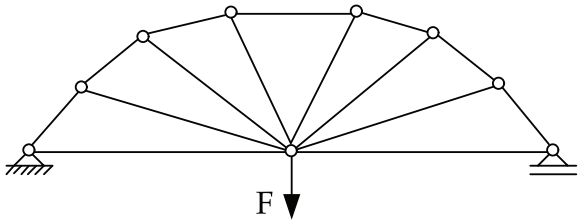


Fig. 4. Optimal structure with 6 free nodes

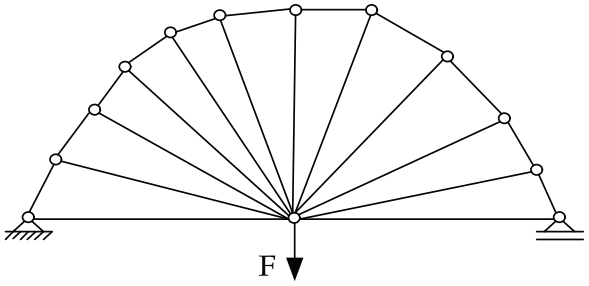


Fig. 5. Optimal structure with 10 free nodes

Of course, this is a subjective approach. There is an obvious correlation between the number of nodes, the amount of load imposed and the cross-section of the respective rod. At this point of research it was impossible to tackle all of the possible influences since the research focus was on a new type of encoding.

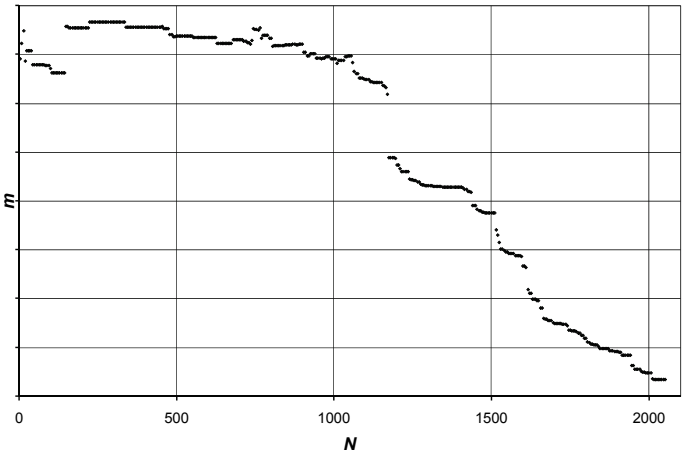


Fig. 6. Graph of the algorithm performance shown in terms of objective function m put against evolution step N

Graph of the algorithm performance shown in terms of objective function *m* and number of evolution steps *N* is presented in *Fig. 6.* This particular graph corresponds to the initial

condition of five free nodes. Every fifth solution is drawn in graph. Because of its stochastic nature algorithm cannot produce same performance graph in every run, however some key features remain. In Fig. 6 it is clearly visible that after the step *N=1100* the value of the objective function stepwise diminishes. With sufficient certainty it could be said that the algorithm moved search to the feasible space. Before the *N=1100* the aim of the algorithm was the minimization of the constraint violation expressed in equation (6) resulting in dispersion of the values of the objective function. After the step *N=2000* the evolution slows down and no significant improvements were noted.

6. Evolution of stiffened panels

The simplified ship hull structure in this case study, see for example a traditional boat in Fig. 7, is modeled as a transversely framed shell of isotropic material under lateral outer pressure p, and longitudinal in-plane stress σ_L (Hughes, 1972), Fig. 8, also considering the state of the art rules and regulations of classification societies based on experience of shipbuilding and shipping (CRS, 2006) (DNV, 1978) that evolved during a long period of development of theory and practice of shipbuilding. The material properties are the elastic modulus E, the Poisson's ratio ν, the allowable normal σ_a and shear τ_a stresses in shell and in framing (CRS, 2006).

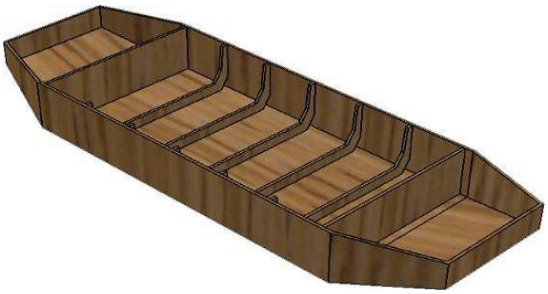


Fig. 7. Boat hull structure

The small deflection elastic plate bending theory (Hughes, 1972) defines the maximal local stress under lateral pressure p in the middle of the longer edge ℓ in the direction of the shorter edge s in the plating of thickness t clamped at stiffeners, Fig. 8. Using the semi-empirical plate side aspect ratio (Hughes, 1972) $k_s = 1 - 0,4\left(\frac{s}{\ell}\right)^2$, the stress in the shell under lateral load p can be assessed as:

$$\sigma_p = 0,5 \cdot p \cdot k_s \cdot \left(\frac{s}{t}\right)^2 \tag{1}$$

The simple elastic beam bending theory (Hughes, 1972) defines the normal stresses in frames, Fig. 8:

$$\sigma_f = p \cdot k_m \cdot \frac{s\ell^2}{W_{f,e}} \tag{2}$$

The end connection factor for clamped frame ends is k_m=1/12. The elastic section modulus $W_{f,e}$ of a single frame accounts for the width of the effective plate flange. The shear stress at supporting ends of the frame web (Hughes, 1972), taking the correction factor c_w=3/2 for rectangular cross sectional area A_f of a flat bar (CRS, 2006) (DNV, 1978) is as shown:

$$\tau_f = \frac{c_w \cdot p \cdot s \cdot \ell}{2A_f} \tag{3}$$

The orthotropic plate elastic bending theory (Hughes, 1972) defines the stresses in the edges of the longer side in the direction of the shorter edge, Fig. 7, of the whole transversely stiffened plate as:

$$\sigma_s = K \cdot p \cdot \frac{s \cdot \ell^2 \cdot e}{I_f} \tag{4}$$

In (4), I_f is the frame moment of inertia including effective plating width and e is the distance from the neutral axes to the plating. From Shade's diagrams (Hughes, 1972) is K=0.0916 for the edges of the longer side in the direction of the shorter edge and K=0.0627 for the edges of the shorter side.

The critical buckling stress of plating under in-plane compression of plates between frames (Hughes, 1972), (CRS, 2006), (DNV, 1978) using the term $\sigma_{p,e} = \frac{\pi^2 E}{12(1-\nu^2)}$ is:

$$\sigma_{p,c} = \sigma_{p,e} \cdot \left(\frac{t_p}{s}\right)^2 \cdot k_p \cdot k_\sigma \tag{5}$$

For transversely stiffened panels is $k_p = \left[1 + \left(\frac{s}{\ell}\right)^2\right]^2$ and for longitudinally stiffened panels is $k_p = 4$. For elastic buckling is $k_\sigma = 1$ and for plastic buckling is $k_\sigma = 1 - \left(\frac{\sigma_{c,e} - \sigma_y / 2}{\sigma_{c,e}}\right)^2$ when $\sigma_{c,e} \geq \sigma_y / 2$.

The torsional buckling of flat bar stiffeners prevents the empirical ratio of height to thickness (DNV, 1978) that is normally < 20.

The ultimate bending strength with respect to multimodal plastic failure modes of plates at the mid of the longer edge of unit plate plastic section modulus $W_{p,p} = \frac{t^2}{4}$ between frames under bending moment $M = k_m \cdot p \cdot s^2$ acting due to lateral pressures p combined with in-plane load σ_L, may be expressed by the following interaction formula (DNV, 1978) $\frac{M}{\sigma_y \cdot W_{p,p}} + \frac{1}{\beta}\left(\frac{\sigma_L}{\sigma_y}\right)^2 = \beta$.

The usage factor β relates the maximal permissible load to the collapse load. Using the factor $k_{L,p} = \left[\beta - \frac{1}{\beta}\left(\frac{\sigma_L}{\sigma_y}\right)^2\right]$ to represent the influence of the in-plane stress, the ultimate lateral pressure on plating accounting for the yield stress σ_y (DNV, 1978) is:

$$p_{u,p,\sigma} = 3 \cdot \left(\frac{t}{s}\right)^2 \cdot \frac{k_{L,p}}{k_s} \cdot \sigma_y \tag{6}$$

The ultimate bending strength of frames under lateral pressure and axial stress is the capability to prevent the plastic failure defined as a three-hinged mechanism (DNV, 1978). For frames with plastic section modulus $W_{f,p}$ including the effective plate flange under bending moment $M = k_m \cdot p \cdot s \cdot \ell^2$ due to lateral pressure p and for small axial stresses σ_x (the shear is usually small) the relation derived from (2) holds (DNV, 1978):

$$p_{f,u,\sigma} = 12 \cdot \frac{W_{f,p}}{s\ell^2} \cdot \varepsilon \cdot \sigma_y \tag{7}$$

where ε is the permissible usage factor.
The ultimate lateral pressure on the whole panel viewed as the orthotropic plate (4), is as shown:

$$p_{b,p,\sigma} = \frac{\sigma_y}{K} \cdot \frac{I_f}{s \cdot \ell^2 \cdot e} \tag{8}$$

Since the transverse in-plane compression of bottom plating is normally small, Fig. 7, it is not likely that buckling of plating occurs at all (DNV, 1978).
The model is a ship hull panel of thickness t, length ℓ, width b which is transversely stiffened by n flat bars of thickness t_w and height h_w at spacing s, Fig. 8. The plate is laterally loaded by pressure p and with in-plane stress σ_L.

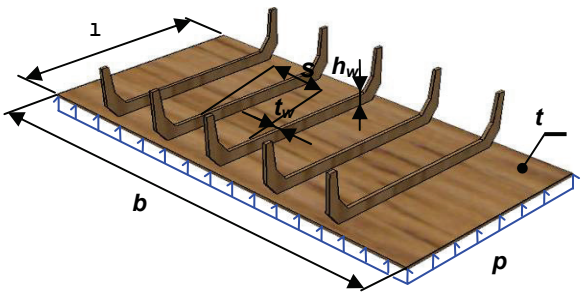


Fig. 8. Panel structural model

The evolutionary design in this example uses the engineering model in order to demonstrate the technical development by employing genetic algorithms aspired with achievement of appropriate safety level as well as with reduction of weight, expenses and production efforts using different materials.

Therefore the stiffened panel design of the case study is basically defined as a general non-linear mathematical programming model of the appropriate ship structure built of the material characterized by material coefficient k and l density ρ following section 4 as follows:

- parameters: $p, \ell, b, k, \rho, \sigma_y, \sigma_f, \sigma_s, \tau_s$
- variables: n, t, t_w, h_w

Design goals are the minimization of panel mass m, the minimization of number of transversely stiffening flat bars n which expresses in a simple way the complexity of design or workmanship expenses and finally the minimization of standard deviation of ultimate load carrying capacity taken as a measures of robustness of a structure (Žiha, 2000) st.dev.($p_{u,p,\sigma}$, $p_{f,p,\sigma}$, $p_{b,p,\sigma}$).

The later encapsulates the robustness of design by leveling out the safety apprehended as the maximum lateral pressure that the whole panel and its structural members – plate and stiffeners can withstand (Žiha, 2000) that means avoidance of week links in the structure.

Finally the design problem is formulated as:

$$\left.\begin{aligned}
&\min\left[m(n,k,t,t_w,h_w,\ell,\rho)\right],\\
&\min\left[n\right],\\
&\min\left[st.dev.(p_{u,p,\sigma},p_{f,p,\sigma},p_{b,p,\sigma})\right],\\
&\text{subject to:}\\
&\quad W \geq W_{\min}(p,k_m,n,\ell,b,\sigma_{f,a}),\\
&\quad t \geq t_{\min}(p,k_u,n,p_{u,p,\sigma},k,\sigma_{p,a}),\\
&\quad t \geq 2\ mm,\\
&\quad \sigma \leq \frac{\sigma_y}{k} \equiv \frac{235}{k},\\
&\quad t_w \cdot h_w \geq A_f(p,n,k,\tau_{f,a}),\\
&\quad \min\left[p_{u,p,\sigma},p_{f,p,\sigma},p_{b,p,\sigma}\right] \geq p,\\
&\quad t_w / h_w \leq 20,\\
&\quad t_w / h_w \geq 10.
\end{aligned}\right\} \qquad (9)$$

At the beginning hard constraints in (9) can hinder the evolutionary process since the majority of the early solutions are infeasible. Consequently, by measuring constraint violations one can rank infeasible solutions.

Later that ranking is added to Pareto frontier of feasible population (Deb, 2001). Constraint violation measure $\Omega(x^{(i)})$ of $i-th$ solution $x^{(i)}$ is derived as summation of normalized violations $\omega_j(x^{(i)})$ (10) (Deb, 2001):

$$\Omega(x^{(i)}) = \sum_{j=1}^{8} R_j \omega_j(x^{(i)}) \qquad (10)$$

No violations were favored so the weighting factor used is $R = 1$ for all j .

For encoding of chromosomes binary strings were used. Every chromosome consists of four genes which comprise four design variables of the ship hull panel. In addition for the refinement of search the Gray coding was applied (Pahl, 1998).

Design variable	t	n	h_w	t_w
The gene number	*1*	*2*	*3*	*4*
Available strings per gene	*10*	*10*	*10*	*10*
Maximum value attainable after mapping [mm]	*130*	*200*	*430*	*20*

Table 3. The chromosome structure

The emergence of new genes 2, 3, and 4, Table 3, for number of frames, thickness and height of the frame web opens potentials for development of plates stiffened by flat bars. These four characteristics together with the problem parameters define all the other panel properties.
Since the evolution was carried through fixed length chromosomes then the length of the individual genes is also a limitation - constraint put upon the search space, that guide evolution towards reasonable solutions and hopefully speed up the overall search process, Table 3.
Control parameters of the applied NSGA-II algorithm (Deb, 2001) were as follows:

- population size $\lambda = 60$,
- offspring population $\mu = \lambda$,
- uniform crossover (Bäck&Fogel, 2000) - probability $p_c = 1$.
- bit flip mutation probability $p_m = 1 / l \equiv 0.026$ (Bäck et. al., 2000).

The genetic algorithm tackles the design of the stiffened plate of a contemporary steel ship transversely stiffened panel structure, Fig. 9, of breadth b=28,8 m, length ℓ =5,17 m under lateral pressure of p=0.1 N/m^2 according to design loads defined by classification rules (CRS, 2006) using potentials of all the genes, Table 3.

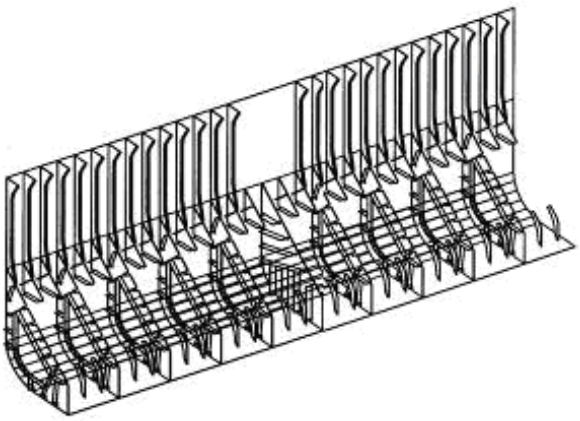


Fig. 9. The modern ship hull transversely framed side structure

The computation results of one out of many iterative trials with repeatable outcomes on standard personal computers are presented as the 3-D Pareto frontier plot *n-m-st.dev.*, Fig. 10.

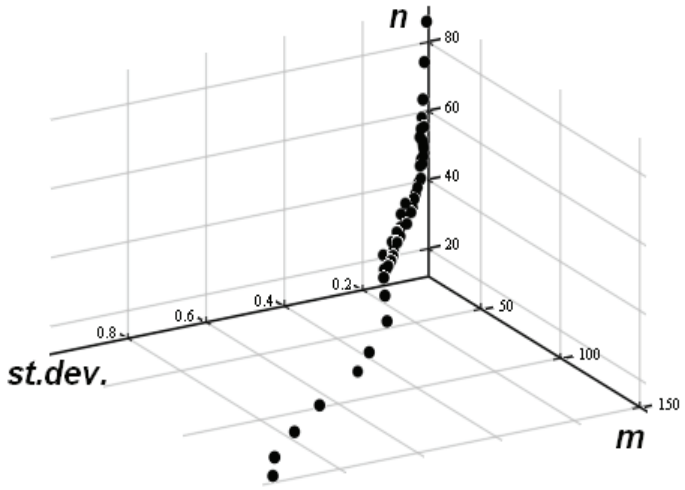


Fig. 10. 3-D Pareto frontier plot

After the full gene potential of chromosome, Table 3, is being unleashed more up to date solutions evolved. The obtained results after 8000 iterations are plotted on Figs. 10. – 14.

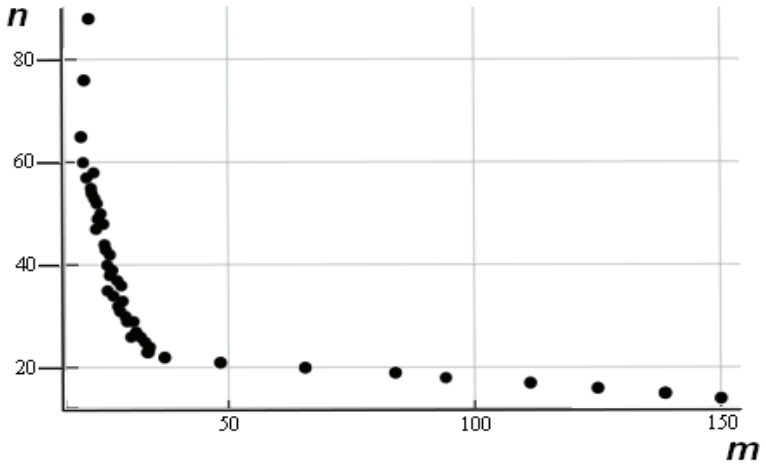


Fig. 11. n-m plot

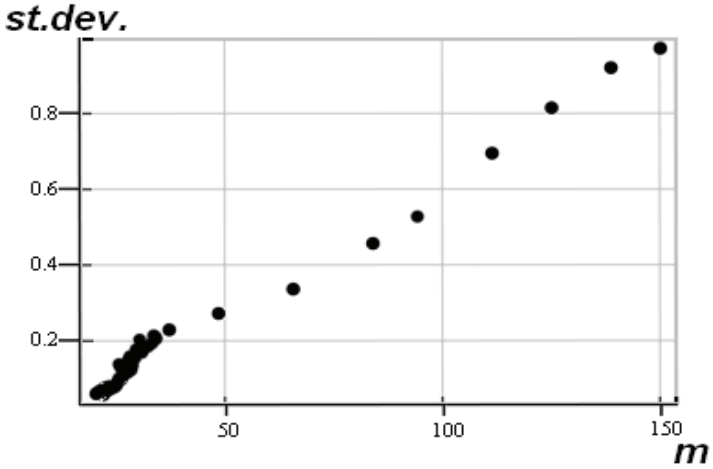


Fig. 12. st.dev.-m plot

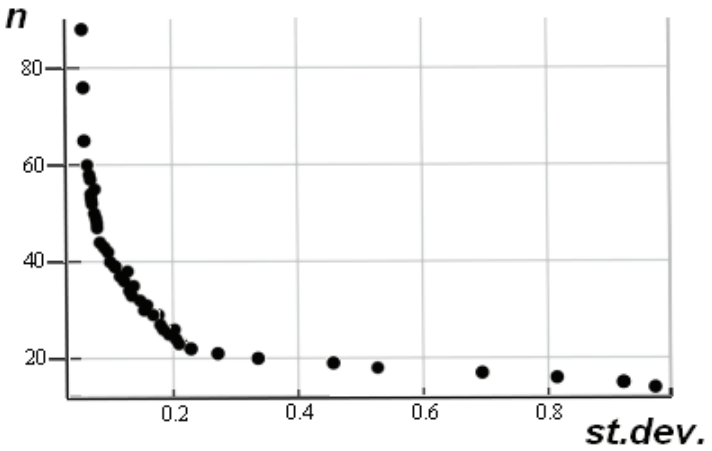


Fig. 13. st.dev.-n plot

The aim of the illustrative example is to interpret the optimization results obtained by evolutionary algorithm as the effects of social and environmental conditions on the development of technical structures. It is comprehensible on one hand, Fig. 11, how the expensive workmanship related to the number of stiffeners irrespective to the material expenses and other technical requirements may yield to preferable solutions of thicker plates with smaller number of stiffeners, even simple plates without stiffeners, regardless of the overall mass of the panel. On the other hand, the socio-environmental condition of expensive material or technical request for light structures irrespective to the workmanship expenses leads to solution of thinner plates with greater number of stiffeners. For highly efficient light-weight structures when the material and workmanship expenses are irrelevant, just the minimal mass, thinner plates with a greater number of stiffeners of

higher class material are preferable. The mathematical model incorporates the assumption of the importance of robustness when the environmental conditions imply uncertainties. The robustness is considered as the minimal variation among safety measures of different failure modes (Žiha, 2000) (inter frame plate bending (6), frame bending (7), overall panel yield (8) and effect of shear stresses (3)). In Fig. 12 it is shown how the request for maximum robustness (minimal standard deviation of safety measures) in this example leads to solution of minimal mass panel that satisfies the prescribed safety level. Moreover the increase of robustness followed by diminution of mass is affordable only by significant increase in workmanship efforts due to large number of built-in stiffeners, Fig. 13.
Implementing the ancient conditions of expensive (unavailable) material (except for example wood) and tough workmanship (no experience and tools available) into the mathematical model the solutions points to least expensive plane plate, Fig. 11, without stiffening as the primitive carved-out logs, Fig. 14.

Fig. 14. The primitive boat structure

Finally, the contemporary engineering model resulting in four genes, Table 1, in the last run degenerates to the one single primitive gene number 1, having the plate thickness for the only property. The design model is used in its most degenerative form appropriate to early days of shipbuilding and lack of engineering knowledge and experience. As a final consequence, the mathematical model points to un-stiffened 125 millimeter thick plating, Fig. 14, as the least workmanship demanding solution although inappropriate for now days practice. The only affordable outcome of one primitive gene is the simple un-stiffened plate of minimal thickness appropriate to ancient conditions for carved-out logs that satisfies the past and modern safety requirements, Fig 15.

Fig. 15. Carved-out log

7. Conclusion

The first case study in this chapter brings a special way of genotype encoding well suited for mechanisms of genetic algorithm operators. Furthermore, it considered how the structural model is decoded from such a genotype. The aim of this study of the new "mesh-like" approach was a consequence of formerly detected deficiencies of the existing methods. The presented advanced methods for the truss structure optimization work perfectly, but are specialized either for the structure properties optimization (Hasançeb, 2007), (Coello&Christiansen, 2000), or the structure topology optimization (Jakiela at all, 2000), (Hamdam, Schoenauer, 2002) and (Kim&Weck, 2004). Shape annealing and shape grammars applied for the structural optimization (Shea&Cagan, 1998) offered an alternative, but a GAs are found as suitable alternatives. For the reasons of simplicity, at this point of research the search space is reduced. The struggle genetic algorithm applied for single objective uses a fixed length chromosome, and the number of nodes is therefore user-defined. The cross-section area of trusses is fixed. Future work will include an introduction of these present parameters as variables with suitable coding and will aim at defining load vectors. To enhance speed the parallelization of algorithm is being considered in future too. With all which has been accomplished, moving away from the single to the multi-objective optimization makes a natural step ahead in evolving truss structures.

The evolutionary design supports normally the contemporary progressive engineering reasoning aspired with achievement of highly efficient products providing socially acceptable safety levels and appropriately lower costs by employing genetic algorithms. However, the second case study in this chapter indicates how the reverse process to the technical progress can reconstruct the origins of contemporary products using evolutionary algorithms on engineering models in two manners. The simplest way is the replication of primitive conditions, such as for example lack of experience, unavailability of appropriate material and technology. Introduction of past conditions into up to date mathematical models corroborates early solutions based on past engineering practice. Reconstruction of past social and environmental conditions may lead to primitive solutions appropriate to early human's engineering but it does not characterize only the evolutionary algorithms. Another way is the simplification or degeneration of the design model that is in terms of genetic algorithms, deactivating or removing more complex genes from the chromosomes that might be viewed as a particular feature of evolutionary algorithms.

Evolutionary design approach upholds that the technical progress goes on if the existing gene potentials are activated or the new evolutionary potentials based on additional knowledge are introduced.

The optimization search by genetic algorithms may be viewed as time-condensed best-practice that in reverse order can back-trace the engineering development either by replicating past condition or by omission of chromosomes introduced into evolutionary models by growth of engineering experience.

The chapter demonstrated that the evolution of structural systems is successfully driven forward towards the fittest structures by the synthesizing criteria of most uniform responsiveness. The changing of external circumstances that provoke uniform structural response is interpretable as the robustness criteria. The research takes up the most commonly used statistical measures of data dispersion to define the structural system robustness for practical purposes. The most conveniently measure appears the minimal coefficient of variations of internal forces under displaced external loads. Shortly, the thesis is that the fittest structure is the robust one.

8. Acknowledgment

This research was supported by two projects of the Ministry of science, education and sports of the Republic of Croatia.

9. References

Bäck T.; Fogel D.B., Michalewicz Z. (2000). Evolutionary Computation 1, Basic Algorithms and Operators Institute of Physics Publishing, Bristol and Philadelphia.

Bäck T., Fogel D. B., Michalewicz Z. Evolutionary Computation, Advanced Algorithms and Operators, Institute of Physics Publishing, Bristol and Philadelphia; 2000.

Bentley, P. (1999). Evolutionary Design by Computers, Morgan Kaufmann.

Chomsky, N. (1957). Syntatic Sructures, Mouton and Co.

Coello, C.A.C.; Christiansen, A.D. (2000). Multiobjective Optimization of Trusses using Genetic Algorithms, Computers and Structures, Elsevier.

CRS (2006). Rules for the Classification of Sea-Going Ships-Part 2 Hull, Croatian Register of Shipping, Split.

Deb, K. (2001). Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons Ltd.

Gero, J.S.; Louis, S.L. (1995). Improving Paretto optimal designs using genetic algorithms, Microcomputers in Civil Engineering, , pp 241–249.

DNV (1978). Ship' Load and Strength Manual, Det Norske Veritas, Hovik.

Goldberg, D.E. (1989). Genetic Algorithms in Search Optimization and Machine Learning, Addison Wesley Longman Inc.

Goldberg, D. E. (2002). Design of Innovatinon, Kluwer academic Publishers.

Golinski, J. (1970). Optimal Synthesis Problems Solved by Means of Nonlinear Programming and Random Methods, Journal of Mechanisms, pp 287-309.

Hamda, H.; Schoenauer, M. (2002). Topological optimum Design with Evolutionary Algorithms, Journal of Convex Analysis, pp 503-517,

Hasançeb, Q. (2007). Optimization of truss bridges within a specified design domain using evolution strategies, Engineering Optimization, Taylor and Francis.

Hubka, V.; Eder, W.E. (1992). Engineering Design: General Procedural Model of engineering Design, Springer-Verlag Berlin Heidelberg.

Hughes, O. (1972.) Rational Ship Structural Design, SNAME, New Jersey.

Jakiela, M.J.; Chapman, C.; Duda, J.; Adewuya, A.; Saitou, K. (2000). Continuum structural topology design with genetic algorithms, Computational. Methods in Applied Mechanical Engineering, Elsevier, pp 339-356

Kim, Y.I.; De Weck, O. (2004). Progressive structural topology optimization by variable chromosome length genetic algorithm, CJK-OSM3.

Pahl, G.; Beitz, W. (1988). Engineering Design – A Systematic Approach, Springer Verlag;

Shea, K.; Cagan, J. (1998). Generating Structural Essays from Languages of Discrete Structures, Artificial Intelligence in Desgin, Kluwer Academic Publishers, pp 365-384.

Tan K.C.; Lee T.H.; Khor E.F. (2002). Evolutionary Algorithms for Multi-Objective Optimization: Performance Assessments and Comparisons, Artificial Intelligence Review, Kluwer Academic Publishers,; pp 253–290.

Yockney, H. (2005). Information theory, Evolution and the Origin of Life, Cambridge University Press.

Zienkiewicz, O. C. (1971). The Finite Element Method in Engineering Science, Mc Graw-Hill.

Wood, K.L.; Otto, K.N. (1999). Product Design Techniques in Reverse Engineering, Systematic design, and New Product Development, Pren-tice-Hall, NY.

Žiha, K. (2000). Redundancy and Robustness of Systems of Events, Probabilistic Eng. Mechanics, Vol. 15, pp. 347-357.

Part 2

Applications

14

Evaluating the α-Dominance Operator in Multiobjective Optimization for the Probabilistic Traveling Salesman Problem with Profits

Bingchun Zhu[1], Junichi Suzuki[2] and Pruet Boonma[3]
[1,2]*University of Massachusetts, Boston*
[3]*Chiang Mai University*
[1,2]*USA*
[3]*Thailand*

1. Introduction

This chapter investigates a noise-aware dominance operator for evolutionary multiobjective optimization algorithms (EMOAs). An EMOA uses a population of individuals, each of which represents a solution candidate. It evolves individuals through generations and seeks the optimal solution(s) in a multiobjective optimization problem (MOP), which is formalized as follows.

$$\left.\begin{array}{r} \min\ F(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \cdots, f_m(\vec{x})]^T \in \mathcal{O} \\ \text{subject to } \vec{x} = [x_1, x_2, \cdots, x_n]^T \in \mathcal{S} \end{array}\right\} \quad (1)$$

$\mathcal{S}$ denotes the decision variable space. $\vec{x}$ denotes a decision variable vector (or solution candidate) with respect to $\mathcal{S}$. It is called an individual in EMOAs. A function vector, $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, consists of m real-value objective functions, each of which produces an objective value with respect to the objective space $\mathcal{O}$. An MOP is to find an individual(s) that minimizes objective values with subject to $\mathcal{O}$.
In an MOP, objective functions (i.e., $f_1(\vec{x}), \cdots, f_m(\vec{x})$ in Equation 1) often conflict with each other; there exist rarely a single individual that is optimum with respect to all objectives. Therefore, an MOP often aims to find the optimal trade-off solutions, or Pareto-optimal solutions, by balancing conflicting objectives simultaneously. The notion of *dominance* plays an important role to seek Pareto optimality in MOPs (Srinivas & Deb, 1995). An individual $\vec{x} \in \mathcal{S}$ is said to *dominate* an individual $\vec{y} \in \mathcal{S}$ (denoted by $\vec{x} \succ \vec{y}$) iif the both of the following conditions are hold.

- $f_i(\vec{x}) \leq f_i(\vec{y}) \ \forall\, i\ = 1, \cdots, m$
- $f_i(\vec{x}) < f_i(\vec{y}) \ \exists\, i\ =\ 1, \cdots, m$

In real-world MOPs, objective functions tend to contain noise (Beyer, 2000; Bianchi et al., 2009). Thus, objective functions can yield different objective values from the same individual from time to time. For considering this noise, Equation 1 is revised as follows.

$$\left.\begin{array}{r} \min\ F(\vec{x}) = [f_1(\vec{x}) + \epsilon_1, f_2(\vec{x}) + \epsilon_2, \cdots, f_m(\vec{x}) + \epsilon_m]^T \in \mathcal{O} \\ \text{subject to } \vec{x} = [x_1, x_2, \cdots, x_n]^T \in \mathcal{S} \end{array}\right\} \quad (2)$$

ϵ_m represents noise in the m-th objective function. Noise in objective functions can interfere with a dominance operator, which determines dominance relationships among individuals. For example, a dominance operator may mistakenly judge that an inferior individual dominates an superior one. Defects in a dominance operator significantly degrades the performance (e.g., convergence velocity) to solve MOPs (Arnold, 2000; Beyer, 2000; Beyer & Sendhoff, 2007; Bianchi et al., 2009; Carroll et al., 2006; Diwekar & Kalagnanam, 1997).

In order to address this issue, this chapter proposes a notion of noise-aware dominance, called *α-dominance*, and studies the α-dominance operator for EMOAs. This operator takes objective value samples of given two individuals, estimates the impacts of noise on the samples and determines whether it is confident enough to judge a dominance relationship between the two individuals. Unlike existing noise-aware dominance operators, the α-dominance operator assume no noise distributions a priori. (See Section 5. for more details.) Thus, it is well applicable to a variety of real-world MOPs whose objective functions follow unknown noise distributions.

This chapter describes the design of the α-dominance operator and evaluates it with the probabilistic traveling salesman problem with profits (pTSPP), which can derive a number of real-world noisy MOPs. pTSPP is a combination of existing two variants of the traveling salesman problem (TSP): the probabilistic TSP (pTSP) (Jaillet, 1985) and the TSP with profits (TSPP) (Feillet et al., 2005). In experimental evaluation, the α-dominance operator is integrated with NSGA-II (Deb et al., 2000), a well-known EMOA, and compared with existing noise-aware dominance operators. Experimental results demonstrate that the α-dominance operator reliably performs dominance operation in pTSPP and outperforms existing noise-aware operators in terms of the optimality, convergence velocity and diversity of individuals.

The remaining part of this chapter is structured as follows. A further related work, particularly in the areas of probabilistic Traveling Salesman Problems(TSPP) and noise handling techniques of evolutionary algorithms, is surveyed in Section 5. Section 2.proposes a new problem, probabilistic Traveling Salesman Problem with Profit (pTSPP) and describes its objectives and objective function. Section 3.introduces background of MOEAs and describes a variant of NSGA-II, a well-known EMOA. To handling the uncertainties whose distribution is unknown a priori, a noise-aware dominance operator is proposed in Section **??**. This section also presents the integration of proposed noise-aware dominance operator and NSGA-II. Then, Section **??** reports computational results of proposed noise-aware dominance operator on some test pTSPP problems with comparison to some other noise-aware dominance operators. Section 6.concludes this research with a summary.

2. Probabilistic Traveling Salesman Problem with Profits (pTSPP)

This paper uses the following notations to define pTSPP. pTSPP is defined on a fully-connected graph $G = (V, E)$.

- $V = \{v_0, v_1, v_2, ..., v_n\}$ is a set of vertices in G, where v_0 is the depot. $V' = V - \{V_0\}$ is a set of n vertices. This paper assumes that vertices are stationary, and $|V|$ does not change dynamically.

- $E = \{v_i, v_j | v_i, v_j \in V; i \neq j\}$ is the set of edges. Each edge $\{v_i, v_j\} \in E$ has an associated cost c_{v_i,v_j}.
- Each vertex $v_i \in V'$ maintains a visiting probability p_{v_i}, which represents the probability that v_i is visited. $p_{v_i} \in [0.0, 1.0]$. The visiting probability of the depot $p_{v_0} = 1.0$.
- Each vertex $v_i \in V'$ has an associated profit $\rho_{v_i} \geq 0.0$. The depot's profit $\rho_{v_0} = 0.0$.
- R is a sequence of vertices, starting and ending with v_0. R may not contain all the vertices in V': $|R| \leq |V'| + 2$. No redundant vertices exist in R. (A node is never visited more than once.) R is an *a posteriori* route; the salesman uses it to decide a posteriori which vertices he actually visits based on the visiting probabilities associated with vertices in R.

pTSPP is to find the Pareto-optimal routes with respect to the following two objectives.

- *Cost*: The total traveling cost that the salesman incurs by visiting vertices in a route. This objective is to be minimized. It is computed as:

$$f_{cost} = \sum_{v_n, v_{n'} \in R} p_{v_n} p_{v_{n'}} c_{v_n, v_{n'}} \tag{3}$$

  where $v_{n'}$ is the next vertex of v_n in R.
- *Profit*: The total profit that the salesman gains by visiting vertices in a route. This objective is to be maximized. It is computed as:

$$f_{profit} = \sum_{v_n \in R} p_{v_n} \rho_{v_n} \tag{4}$$

Two objectives in pTSPP conflict with each other. For example, a shorter route (i.e., a route containing a smaller number of vertices) yields a lower cost and a lower profit. On the contrary, a longer route (i.e., a route containing a larger number of vertices) yields a higher cost and a higher profit.
pTSPP inherently considers noise in its objective functions. Following the notations in Equation 2, pTSPP is formulated as follows.

$$\left.\begin{array}{r} \min\ F(R) = [f_{cost}(R) + \epsilon_{cost}, \frac{1}{f_{profit}(R)} + \epsilon_{profit}]^T \in \mathcal{O} \\ \text{subject to } R = [v_0, \cdots, v_n, v_{n'} \cdots, v_0] \in \mathcal{S} \end{array}\right\} \tag{5}$$

As mentioned in Section 1, pTSPP is a combination of pTSP (Bertsimas & Howell, 1993; Jaillet, 1985) and TSPP (Feillet et al., 2005). pTSP is to find an optimal *a priori* route with the minimum cost in which each vertex requires a visit of the salesman with a given visiting probability. TSPP is to find the optimal route, with respect to profit as well as cost, with which the salesman visit a subset of given vertices. pTSPP extends pTSP in a sense that pTSPP computes the total profit of a route based on the profit and visiting probability associated with each vertex in the route (Equation 4). Unlike TSPP, pTSPP considers a visiting probability for each vertex.
A number of real-world noisy MOPs can be reduced to pTSPP as various real-world optimization problems can be reduced to pTSP and TSPP (Bertsimas & Howell, 1993; Feillet et al., 2005; Jaillet, 1985; Jozefowiez et al., 2008a). For example, pTSPP can represent noisy MOPs in transportation planning, supply chain networks, data routing/gathering in computer networks.

3. The proposed evolutionary multiobjective optimization algorithm for pTSPP

This section describes the proposed noise-aware evolutionary multiobjective optimization algorithm (EMOA) to solve pTSPP. It is designed to evolve individuals (i.e., solution candidates) toward the Pareto-optima through generations with various operators such as parent selection, crossover, mutation, selection, individual ranking and diversity preservation operators. The α-dominance operator is used in the parent selection and individual ranking operators. Section 3.1explains the representation of individuals in the proposed algorithm. Section 3.2overviews the algorithmic structure of the proposed algorithm. Sections 3.3to 3.5 describe key operators in the proposed algorithm.

3.1 Individual representation

In the proposed EMOA, each individual represents a solution candidate for pTSPP: an a posteriori route R that contains a sequence of vertices. (See Section 2.) Every individual has the depot (v_0) as its first and last element. Figure 1 shows an example individual. Given this route, the salesman starts with v_0, visits v_3 and its subsequent 7 nodes, and returns back to v_0.

0	3	5	7	2	10	4	9	1	0

Fig. 1. The Structure of an Example Individual

Different individuals have different lengths, depending on the number of nodes to be visited.

3.2 Algorithmic structure

Listing 1 shows the algorithmic structure of evolutionary optimization in the proposed EMOA. It follows the evolutionary optimization process in NSGA-II, a well-known existing EMOA (Deb et al., 2000).

At the *0*-th generation, N individuals are randomly generated as the initial population $\mathcal{P}_0$ (Line 2). Each of them contains randomly-selected vertices in a random order. At each generation (g), a pair of individuals, called parents (p1 and p2), are chosen from the current population $\mathcal{P}_g$ using a binary tournament (Lines 6 and 7). A binary tournament randomly takes two individuals from $\mathcal{P}_g$, compares them based on the α-dominance relationship between them, and chooses a superior one as a parent.

With the crossover rate P_c, two parents reproduce two offspring with a crossover operator (Lines 8 and 9). Each offspring performs mutation with the mutation rate P_m (Lines 10 to 15). The binary tournament, crossover and mutation operators are executed repeatedly on $\mathcal{P}_g$ to reproduce N offspring. The offspring ($\mathcal{O}_g$) are combined with the parent population $\mathcal{P}_g$ to form $\mathcal{R}_g$ (Line 19).

The selection process follows the reproduction process. N individuals are selected from $2N$ individuals in $\mathcal{R}_g$ as the next generation's population ($\mathcal{P}_{g+1}$). First, the individuals in $\mathcal{R}_g$ are ranked based on their α-dominance relationships. Non-dominated individuals are on the first rank. The i-th rank consists of the individuals dominated only by the individuals on the $(i-1)$-th rank. Ranked individuals are stored in $\mathcal{F}$ (Line 20). $\mathcal{F}_i$ contains the i-th rank individuals.

Then, the individuals in $\mathcal{F}$ move to $\mathcal{P}_{g+1}$ on a rank by rank basis, starting with $\mathcal{F}_1$ (Lines 23 to 26). If the number of individuals in $\mathcal{P}_{g+1} \cup \mathcal{F}_i$ is less than N, $\mathcal{F}_i$ moves to $\mathcal{P}_{g+1}$. Otherwise, a subset of $\mathcal{F}_i$ moves to $\mathcal{P}_{g+1}$. The subset is selected based on the crowding distance metric, which measures the distribution (or diversity) of individuals in the objective space (Deb et al.,

2000) (Lines 27 to 29). The metric computes the distance between two closest neighbors of an individual in each objective and sums up the distances associated with all objectives. A higher crowding distance means that an individual in question is more distant from its neighboring individuals in the objective space. In Line 28, the individuals in $\mathcal{F}_i$ are sorted based on their crowding distance measures, from the one with the highest crowding distance to the one with the lowest crowding distance. The individuals with higher crowding distance measures have higher chances to be selected to $\mathcal{P}_{g+1}$ (Line 29).

```
 1  g = 0
 2  P_g = Randomly generated N individuals
 3  while g < MAX-GENERATION do
 4    O_g = ∅
 5    while |O_g| < N do
 6      p_1 = tournament(P_g)
 7      p_2 = tournament(P_g)
 8      if random() ≤ P_c then
 9        {o_1,o_2} = crossover(p_1, p_2)
10        if random() ≤ P_m then
11          o_1 = mutation(o_1)
12        end if
13        if random() ≤ P_m then
14          o_2 = mutation(o_2)
15        end if
16        O_g = {o_1,o_2} ∪ O_g
17      end if
18    end for
19    R_g = P_g ∪ O_g
20    F = sortByDominationRanking(R_g)
21    P_{g+1} = {∅}
22    i = 1
23    while |P_{g+1}| + |F_i| ≤ N do
24      P_{g+1} = P_{g+1} ∪ F_i
25      i = i + 1
26    end while
27    assignCrowdingDistance(F_i)
28    sortByCrowdingDistance(F_i)
29    P_{g+1} = P_{g+1} ∪ F_i[1 : (N − |P_{g+1}|)]
30    g = g + 1
31  end while
```

Listing 1. Optimization Process in the Proposed EMOA

3.3 Crossover

The proposed EMOA adopts partially-mapped crossover (PMX) as its crossover operator. PMX was originally proposed to solve TSP (Goldbert & Lingle, 1985). It is known that PMX effectively works for TSP and its variants (Goldbert & Lingle, 1985; Kellegőz et al., 2008).

PMX first selects two crossover points on parent individuals at random. A sub-route surrounded by the two crossover points is called a mapping section. In an example in Figure 2, parent 1's mapping section is [3, 9, 4, 13], and parent 2's mapping section is [2, 8, 7, 3]. Given two mapping sections, mapping relationships are formed by paring elements in the mapping sections on a position by position basis. In Figure 2, the first elements in two mapping sections, 2 and 3, are paired; 2–3 is the first mapping relationship. Similarly, three extra mapping relationships, 8–9, 7–4 and 3–13, are formed. In order to reproduce two offspring from two

parents, mapping sections are exchanged between parents. In Figure 2, parent 1's mapping section is replaced with parent 2's; therefore, one of proto-offspring is [0, 2, 7, 6, 2, 8, 7, 3, 10, 5, 1, 0]. (Note that Figure 2 does not show the other proto-offspring.) If proto-offspring has redundant vertices across its mapping section and the other section, PMX replaces each redundant vertex with its counterpart shown in mapping relationships. In Figure 2, 7 and 2 are redundant vertices. Given a mapping relationship of 7–4, 7 is replaced with 4 in the non-mapping section. (Replacements always occur in the non-mapping section.) 2 is replaced with 13 by referencing two mapping relationships (2–3 and 3–13) recursively.

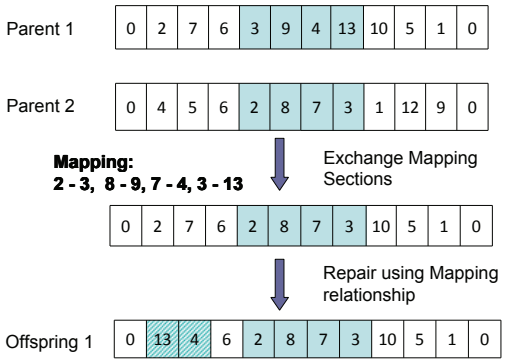


Fig. 2. An Example Crossover (PMX) Process

3.4 Mutation

The proposed EMOA provides a multi-mode mutation operator to alter reproduced offspring. The operator has the following four modes and selects one of them at a time randomly.

1. *Add*: randomly chooses a vertex from unvisited vertices and inserts it to a randomly-selected position in a route (Figure 3(a)). This mode gives the salesman a higher chance to visit more vertices.
2. *Delete*: removes a randomly-selected vertex from a route(Figure 3(b)). This mode reduces the number of vertices that the salesman visits.
3. *Exchange*: randomly chooses a vertex in a route and replaces it with one of unvisited vertices (Figure 3(c)). The unvisited vertex is also selected at random. This mode is intended to change a set of vertices that the salesman visits.
4. *Swap*: exchanges the positions of two randomly-selected nodes in a route (Figure 3(a)). This mode is intended to change a visiting sequence of vertices.

3.5 α-Dominance

This section describes the notion of α-dominance and the design of the α-dominance operator. α-dominance is a new dominance relationship that extends a classical dominance relationship described in Section 1. It takes objective value samples of given two individuals, estimates the impacts of noise on the samples, and determines whether it is confident enough to judge which one is superior/inferior between the two individuals.

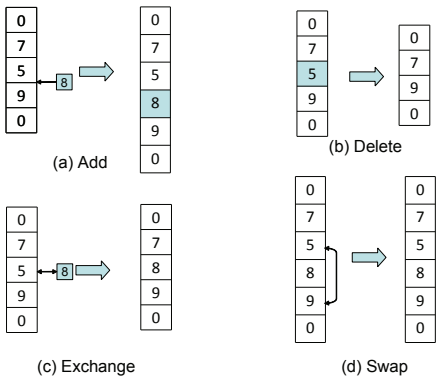


Fig. 3. Example Mutation Processes

3.5.1 The α-dominance operator

The α-dominance operator determines the α-dominance relationship between given two individuals by statistically processing their objective value samples. With this operator, individual A is said to *α-dominate* individual B (denoted by $A \succ_\alpha B$), iif:

- A's and B's objective value samples are classifiable with a statistical confidence level of α, and
- $\mathcal{C}(A, B) = 1 \wedge \mathcal{C}(B, A) < 1$.

In order to examine the first condition, the α-dominance operator classifies A's and B's objective value samples with Support Vector Machine (SVM), and measures a classification error. (See Step 1 in an example shown in Figure 4.) The error (e) is computed as the ratio of the number of miss-classified samples to the total number of samples. For evaluating the confidence level (α) in a classification error, the α-dominance operator computes the classification error's confidence interval (e_{int}):

$$e_{int} = e \pm t_{\alpha,n-1}\sigma \tag{6}$$

$t_{\alpha,n-1}$ denotes a single-tail t-distribution with α confidence level and $n-1$ degrees of freedom. n denotes the total number of samples. σ is the standard deviation of e. It is approximated as follows.

$$\sigma \cong \sqrt{\frac{e}{n}} \tag{7}$$

If e_{int} is significant (i.e., if e_{int} does not span zero), the α-dominance operator cannot classify A's and B's samples with the confidence level of α. Thus, the operator determines that A and B do not α-dominate each other. (See Step 2 in Figure 4.)
If e_{int} is not significant (i.e., if e_{int} spans zero), the α-dominance operator can classify A's and B's samples with the confidence level of α. Thus, the operator examine the aforementioned second condition. (See Step 2 in an example shown in Figure 4.) It measures $\mathcal{C}$-metric (Zitzler & Thiele, 1999) with a classical notion of dominance ($\succ$) described in Section 1.$\mathcal{C}(A, B)$ denotes the fraction of individual B's samples that at least one sample of individual A dominates:

$$\mathcal{C}(A,B) = \frac{|\{b \in B \mid \exists a \in A : a \succ b\}|}{|B|} \tag{8}$$

If $\mathcal{C}(A,B) = 1$, all of B's samples are dominated by at least one sample of A. If $\mathcal{C}(B,A) < 1$, not all of A's samples are dominated by at least one sample of B. The α-dominance operator determines $A \succ_\alpha B$ if $\mathcal{C}(A,B) = 1$ and $\mathcal{C}(B,A) < 1$. If $\mathcal{C}(A,B) < 1$ and $\mathcal{C}(B,A) < 1$, the operator determines neither $A \succ_\alpha B$ nor $B \succ_\alpha A$. See Figure 4 as well.

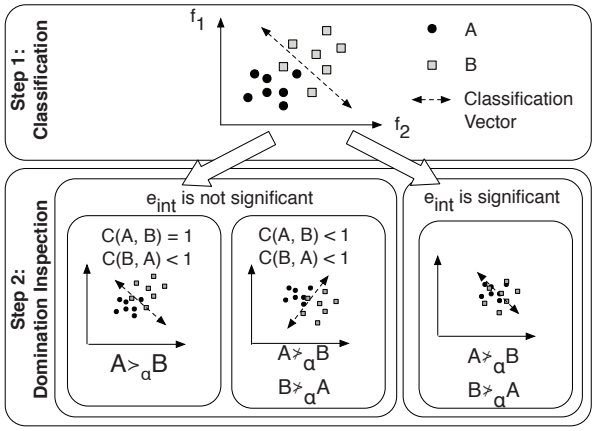


Fig. 4. An Example Process to determine the α-Dominance Relationship between two individuals (A and B)

Figure 4 shows an example to determine the α-dominance relationship between two individuals, A and B, with two objectives, f_1 and f_2, to be minimized. Individual A and B have seven samples each. First, the α-dominance operator classifies these 14 samples in the objective space with SVM and computes e_{int}. Suppose SVM produces a classification vector as shown in Figure 4. Two samples of B are miss-classified; $e = \frac{2}{14}$ (0.143). Thus, $\sigma \cong \sqrt{\frac{0.143}{14}} = 0.1$. Assuming the confidence level α of 95%, $e_{int} = 0.143 \pm 1.771 * 0.1 = 0.143 \pm 0.1771$. Since e_{int} spans zero, A's and B's samples are classifiable with the confidence level of 95%. This means that the aforementioned first condition is hold. In order to examine the second condition, the α-dominance operator measures $\mathcal{C}(A,B)$ and $\mathcal{C}(B,A)$. In Figure 4, $\mathcal{C}(A,B) = 1$ and $\mathcal{C}(B,A) = 2/14 < 1$. This means that the second condition is hold. As a re result, the α-dominance operator concludes $A \succ_\alpha B$.

Listing 2 shows pseudo code of the α-dominance operator. $\mathcal{A}$ and $\mathcal{B}$ denote individual A's and B's samples, respectively. $\mathcal{A}'$ and $\mathcal{B}'$ denote two clusters of samples classified by SVM.

```
1  function alphaDominance(𝒜, ℬ, α)
2    {𝒜', ℬ'} = SVMClassifier(𝒜, ℬ)
3    e = 0 // classification error
4    for each x ∈ 𝒜' do
5      if x ∉ 𝒜 then
6        e = e + 1
7      end if
8    end for
9
10   for each x ∈ ℬ' do
```

```
11      if x ∉ B then
12        e = e + 1
13      end if
14    end for
15
16    if e = 0 then
17      if C(A,B) = 1 then
18        return 1 // A α-dominates B.
19      else if C(B,A) = 1 then
20        return -1 // B α-dominates A.
21      else
22        return 0 // A & B are non-α-dominated.
23      end if
24    else
25      t = t-test(α, sqrt(e,|A| + |B|))
26      if  e − t < 0 then // e_int spans zero.
27        return 0 // A & B are non-α-dominated.
28      else
29        if C(A,B) = 1 then
30          return 1 // A α-dominates B.
31        else if C(B,A) = 1 then
32          return -1 // B α-dominates A.
33        else
34          return 0 // A & B are non-α-dominated.
35        end if
36      end if
37    end if
38  end function
```

Listing 2. Pseudocode of the α-Dominance Operator

3.5.2 Dynamic adjustment of confidence level

The α-dominance operator dynamically adjusts its confidence level (α) by estimating how close individuals have converged to the Pareto-optimal front. The convergence of individuals is estimated based on their disorderliness in the objective space. When individuals are disordered in the objective space, it indicates that they have not converged enough to the Pareto-optimal front. Therefore, the α-dominance operator maintains a low confidence level to determine the α-dominance relationships among individuals in a less strict manner and have diverse individuals explore the decision space and seek the Pareto front. Conversely, when individuals are ordered in the objective space, which indicates that individuals have converged close to the Pareto front, the α-dominance operator increases its confidence level to perform dominance operation in a more strict manner.

The α-dominance operator measures the disorderliness of individuals as their entropy in the objective space. To this end, a hypercube is created in the objective space. Its size is bounded by the maximum and minimum objective values yielded by individuals. (Note that all samples of all individuals, including dominated or non-dominated ones, are plotted in the objective space.) The hypercube is divided to sub-cubes. For example, Figure 5 shows six individuals plotted in a three dimensional hypercube contains eight sub-cubes.

The entropy of individuals (H) is computed as:

$$\left.\begin{aligned} H &= -\textstyle\sum_{i\in C} P(i)\log_2(P(i)) \\ P(i) &= \frac{n_i}{\sum_{i\in C} n_i} \end{aligned}\right\} \tag{9}$$

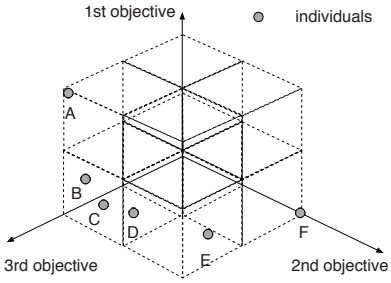


Fig. 5. An Example Hypercube in the Objective Space

C denotes a set of sub-cubes in a hypercube, and $P(i)$ denotes the probability that individuals exist in the i-th sub-cube. n_i denotes the number of individuals in the i-th sub-cube. Entropy (H) is normalized as follows:

$$H_o = \frac{H}{H_{\max}} = \frac{H}{\log_2 n} \tag{10}$$

H_{max} is the maximum entropy: the entropy in the case that all sub-cubes have the same number of individuals. n denotes the total number of sub-cubes. Given normalized entropy (H_o), the confidence level α is adjusted as follows:

$$\alpha = ((\alpha_{\max} - \alpha_{\min})\sqrt{1 - (1 - H_o)^2}) + \alpha_{\min} \tag{11}$$

$\alpha_{\max}$ and $\alpha_{\min}$ denote the predefined maximum and minimum confidence levels, respectively. α is adjusted in a non-linear fashion; a unit circle function is used to map H_o to α.

3.5.3 Integration of the α-dominance operator with parent selection and individual ranking operators

This section describes how α-dominance operator is integrated with the parent selection operator (`tournament()`; Lines 6 and 7 in Listing 1) and the individual ranking operator (
`sortByDominatinoRanking()`; Line 20 in Listing 1).
Listing 3 shows pseudo code of a noise-aware parent selection (binary tournament) operator that leverages the α-dominance operator. $\mathcal{P}$ is the current population of individuals.

```
1  function tournament(P)
2    a = randomSelection(P)
3    b = randomSelection(P)
4    A = samplesOf(a)
5    B = samplesOf(b)
6    r = alphaDominance(A, B, α)
7    if r = 1 then
8      return a
9    else if r = -1 then
10     return b
11   else if r = 0 then
12     if random() > 0.5 then
13       return a
14     else
15       return b
```

```
16      end if
17    end if
18  end function
```

Listing 3. Pseudocode of a Noise-aware Binary Tournament Operator using the α-Dominance Operator

First, two individuals (a and b) are randomly drawn from the current population $\mathcal{P}$ (Lines 2 and 3). Then, in Lines 4 and 5, their samples are obtained to execute the α-dominance operator in Line 6. Depending on the operator's return value (r), one of two individuals (a or b) is returned as a parent individual (Line 7 to 15). If the α-dominance operator cannot determine the α-dominance relationship between a and b (i.e., if $r = 0$), one of them is randomly selected and returned.
Listing 4 shows pseudo code of a noise-aware individual ranking operator that leverages the α-dominance operator. sortByDominatinoRanking() calls `findNonDominatedFront()`, which identifies non-α-dominated individuals in a given population using theα-dominance operator (Lines 11 to 27).

```
1   function sortByDominationRanking(P)
2     i = 1
3     while P ≠ ∅ do
4       F_i = findNonDominatedIndividuals(P)
5       P = P \ F_i
6       i = i + 1
7     end while
8     return F
9   end function
10
11  function findNonDominatedIndividuals(P)
12    P' = ∅
13    for each p ∈ P and p ∉ P' do
14      P' = P' ∪ {p}
15      for each q ∈ P' and q ≠ p do
16      P' = P' ∪ {p}
17      for each q ∈ P' and q ≠ p do
18        A = samplesOf(p)
19        B = samplesOf(q)
20        r = alphaDominance(A, B, α)
21        if r = 1 then
22          P' = P' \ {q}
23        else if r = −1 then
24          P' = P' \ {p}
25        end if
26      end for
27    end for
28    return P'
29  end function
```

Listing 4. Pseudocode of a Noise-aware Individual Ranking Operator using the α-Dominance Operator

4. Experimental evaluation

This section evaluates the proposed EMOA, particularly its α-dominance operator, through a series of computational experiments.

4.1 Test problems

This evaluation study uses three test problems that are built based on three TSP instances: ch130, pr226 and lin318. The TSP instances are obtained from TSPLIB*(Reinelt, 1991). ch130, pr226 and lin318 contain 130, 226 and 318 vertices, respectively. They are customized in this evaluation study so that each vertex maintains a profit and a visiting probability. The value ranges of a profit and a visiting probability are [1.0, 100.0] and [0.0,1.0], respectively. Both values are assigned to each vertex at a uniformly random.

A certain noise is generated and injected to each of two objective functions every time it is evaluated, as shown in Equation 5. Two types of noise are generated: random noise, which follows continuous uniform distributions, and Gaussian noise, which follow normal distributions. Each noise type has three levels of noise: low, medium and high. Table 1 illustrates noise configurations. For random noise, each cell of the table shows a pair of the lower and upper bounds of noise values. For Gaussian noise, each cell of the table shows a pair of the mean and variance of noise values.

		Random noise (Uniform distribution)			Gaussian noise (Normal distribution)		
		ch130	pr226	lin318	ch130	pr226	lin318
Cost	Low	[-20,20]	[-320,320]	[-96,96]	(0,40)	(0,740)	(0,192)
	Medium	[-80,80]	[-1280,1280]	[-384,384]	(0,100)	(0,1600)	(0,480)
	High	[-140,140]	[-2240,2240]	[-672,672]	(0,160)	(0,2560)	(0,768)
Profit	Low	[-2,2]	[-2,2]	[-2,2]	(0,4)	(0,4)	(0,4)
	Medium	[-8,8]	[-8,8]	[-8,8]	(0,10)	(0,10)	(0,10)
	High	[-14,14]	[-14,14]	[-14,14]	(0,16)	(0,16)	(0,16)

Table 1. Noise Configurations for Costs and Profits

4.2 Algorithmic and experimental configurations

The proposed EMOA is configured with a set of parameters shown in Table 2. It is called NSGA-II-A, or simply A, in this evaluation study because it follows NSGA-II's algorithmic structure and customizes the structure with the α-dominance operator, the PMX crossover operator and a mutation operator described in Section 3.4 In order to evaluate the α-dominance operator, NSGA-II-A is compared with the following three variants of NSGA-II:

- NSGA-II (or simply R): the original NSGA-II (Deb et al., 2000) with its crossover and mutation operators replaced by PMX and a mutation operator described in Section 3.4 . Its classical dominance operator does not consider noise in objective functions.
- NSGA-II-U (or simply U): NSGA-II with its classical dominance operator by a noise-aware dominance operator that assumes uniform distribution noise (Teich, 2001).
- NSGA-II-N (or simply N): NSGA-II with its classical dominance operator by a noise-aware dominance operator that assumes normal distribution noise (Eskandari et al., 2007).

All experiments have been implemented and carried out with jMetal (Durillo et al., 2006). Every experimental result is obtained and shown based on 20 independent experiments.

*http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

Parameter	Value	Parameter	Value
Population size	100	# of samples per individual	30
Max # of generations	500	SVM type	C-support vector
Crossover rate	0.9		classification
Mutation rate	0.2	SVM kernel	Linear
$\alpha_{\min}$ in Equation 11	0.90	C parameter for SVM	1
$\alpha_{\max}$ in Equation 11	0.99	SVM termination criteria	$1e^{-3}$

Table 2. Parameter Configurations

4.3 Metrics for performance evaluation

This evaluation study uses the following five performance metrics to compare individual algorithms.

- *The number of non-dominated individuals*: counts the number of non-dominated individuals in the population at the last (i.e., the 500th) generation. The higher this number is, the more successfully an algorithm in question has evolved and converged individuals by eliminating dominated ones. This metric evaluates the degree of convergence/evolution pressure on individuals.
- *Hypervolume* (Zitzler & Thiele, 1999): measures the volume that non-dominated individuals cover in the objective space. The higher a hypervolume measure is, the closer non-dominated individuals are to the Pareto-optima. This metric evaluates the optimality of individuals.
- $D1_R$ (Knowles & Corne, 2002): is computed as follows.

$$D1_R(A) = \frac{1}{|R|} \sum_{\mathbf{r} \in R} min_{\mathbf{z} \in A} d(\mathbf{r}, \mathbf{z}) \tag{12}$$

  A denotes a set of non-dominated individuals. R denotes a set of reference individuals. $d(\mathbf{r}, \mathbf{z}) = max_k \frac{(r_k - z_k)}{R_k}$ where $k = 1, ..., K$ indexes objectives and R_k is the range of objective values that individuals in R yield with respect to the k-th objective. r_k and z_k denote the objective values that individuals r and z yield with respect to the k-th objective. In this evaluation study, R contains a set of non-dominated individuals that NSGA-II produces when no noise is given to objective functions. Therefore, the lower a $D1_R$ measure is, the more effectively an algorithm in question cancels the existence of noise to yield a more similar performance as NSGA-II's. This metric evaluates the optimality of individuals as well as their degree of noise canceling.
- *U-metric* (Leung & Wang, 2003): is computed as follows.

$$U = \frac{1}{D} \sum_{i=1}^{D} \left| \frac{d_i}{\bar{d}} - 1 \right| \tag{13}$$

  d_i denotes the euclidean distance between the i-th individual and its nearest neighbor in the objective space. D denotes the total number of pairs of the nearest neighbors among non-dominated individuals. $\bar{d} = \frac{1}{D} \sum_{i=1}^{D} d_i$ is the average of d_i. The lower a U-metric measure is, the more uniformly individuals are distributed in the objective space. This metric evaluates the distribution (or diversity) of individuals.

- *C-Metric* (Zitzler & Thiele, 1999): uses Equation 8 to compare two different algorithms by examining the two sets of non-dominated individuals they produce.

In addition to the above metrics, this evaluation study examines the objective values that the non-dominated individuals of each algorithm yield at the last generation.

4.4 The number of non-dominated individuals in the population

Tables 3 and 4 show the number of non-dominated individuals that individual algorithms produces at the last generation. Its average and standard deviation results are obtained based on 20 independent experiments. A bold font face is used to indicate the best result(s) in each noise level case of each problem.

Tables 3 and 4 demonstrate that NSGA-II-A and NSGA-II consistently yield the best and worst results, respectively, in both cases with normal and uniform distribution noises. Under normal distribution noise, NSGA-II-A produces 95 or more non-dominated individuals, while NSGA-II produces only 32.2 in the high noise case of lin318. Under uniform distribution noise, NSGA-II-A evolves all individuals to be non-dominated in all cases, while NSGA-II produces only 49.7 non-dominated individuals in the high noise case of lin318. Tables 3 and 4 illustrate that the α-dominance operator successfully retains a high pressure to evolve and converge

Problem	Noise level	NSGA-II-A		NSGA-II		NSGA-II-N		NSGA-II-U	
		Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd
ch130	Low	**100**	0	74.7	9.89	**100**	0	**100**	0
	Medium	**99.2**	2.8	52.5	9.1	98.8	2.7	93.2	3.9
	High	**98.9**	0	34.1	9.3	90.1	8.9	89.5	2.1
pr226	Low	**100**	0	68.2	6.9	**100**	0	94.2	5.1
	Medium	**99.0**	12.5	42.3	4.7	93.2	8.2	86.5	7.9
	High	**97.1**	5.87	30.1	7.6	89.5	5.9	80.4	13.5
lin318	Low	**100**	0	66.3	9.5	99.5	7.8	96.5	3.5
	Medium	**99.5**	1.8	49.35	8.6	90.4	5.2	83.8	16.5
	High	**95.0**	12.5	32.2	7.2	85.3	4.6	80.1	12.4

Table 3. The Number of Non-dominated Individuals at the last Generation when Normal Distribution Noise is injected to Objective Functions

Problem	Noise level	NSGA-II-A		NSGA-II		NSGA-II-N		NSGA-II-U	
		Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd
ch130	Low	**100**	0	96.9	5.7	**100**	0	**100**	0
	Medium	**100**	0	56.2	10.4	98.1	2.1	**100**	0
	High	**100**	0	43.3	9.9	92.3	4.5	98.4	2.1
pr226	Low	**100**	0	83.8	11.1	**100**	0	100	0
	Medium	**100**	0	56.5	9.9	86.5	5.4	**100**	0
	High	**100**	0	46.3	7.9	80.6	10.2	90.0	4.8
lin318	Low	**100**	0	73.4	9.1	**100**	0	**100**	0
	Medium	**100**	0	58.4	6.9	89.9	8.7	98.4	1.1
	High	**100**	0	49.7	8.7	69.6	11.6	89.3	8.1

Table 4. The Number of Non-dominated Individuals at the last Generation when Uniform Distribution Noise is injected to Objective Functions

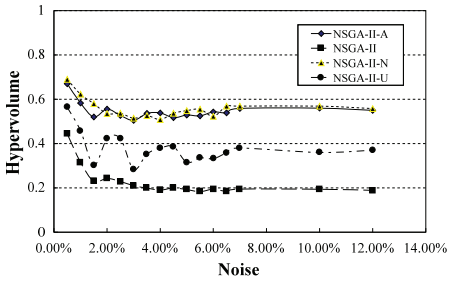


Fig. 6. Hypervolume (ch130, Normal Distribution Noise)

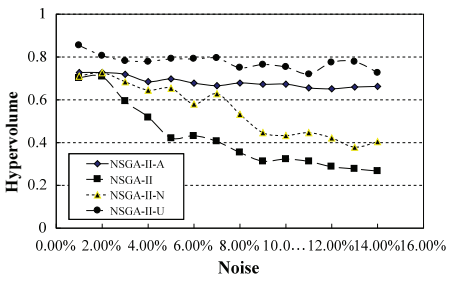


Fig. 7. Hypervolume (ch130, Uniform Distribution Noise)

individuals even in a harder problem (lin318) with a high level of noise. On the contrary, NSGA-II significantly loses the pressure as a given problem becomes harder and a given noise level becomes higher because its dominance operator does not handle noise at all.

In Table 3, NSGA-II-N produces 100 non-dominated individuals in two cases and yields better results than NSGA-II and NSGA-II-U because it assumes normal distribution noise beforehand. However, NSGA-II-A outperforms NSGA-II-N as a given problem becomes harder and a given noise level becomes higher. A similar observation is made in Figure 4; NSGA-II-A outperforms NSGA-II-U even when uniform distribution noise is injected to objective functions, as a given problem becomes harder and a given noise level becomes higher.

4.5 Optimality evaluation with hypervolume

Figures 6 to 11 show the hypervolume measures that individual algorithms yield in each problem with different noise distributions. NSGA-II-A yields the best hypervolume results in most cases (except in ch130 with uniform distribution; Figure 7). NSGA-II yields the worst results in most cases (except in pr226 with normal distribution; Figure 9), which is reasonable because its dominance operator does not handle noise in objective functions.

In Figure 6 (ch130 with normal distribution noise), NSGA-II-A performs similarly to NSGA-II-N, which outperforms NSGA-II and NSGA-II-U, because it designed to handle normal distribution noise. As a given problem becomes harder with normal distribution noise, NSGA-II-A yields better hypervolume measures than NSGA-II-N. (See Figures 8 and 10.) A similar observation is made in Figures 7, 9 and 11. In Figure 7, NSGA-II-A is outperformed by NSGA-II-U, which is designed to handle uniform distribution noise. However, it outperforms NSGA-II-U in harder problems (Figures 9 and 11). These results demonstrate that the α-dominance operator allows NSGA-II-A to successfully seek quality solutions toward the Pareto-optima regardless of problems and noise distributions.

In general, all algorithms yield smaller hypervolume measures as the amount of noise increases. However, the hypervolume measures of NSGA-II-A do not vary largely under different noise levels. It can maintain the hypervolume of 0.6 or higher in all the cases. NSGA-II, NSGA-II-N (under uniform distribution noise) and NSGA-II-U (under normal distribution noise) significantly decrease their hypervolume measures as given noise levels increases. In the pr226 problem with the highest normal distribution noise, NSGA-II-A performs 32.8% better than NSGA-II-U, 50.8% better than NSGA-II and 67.2% better than

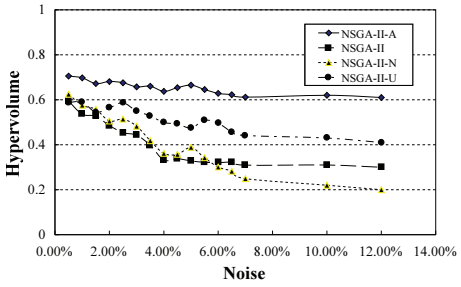


Fig. 8. Hypervolume (pr226, Normal Distribution Noise)

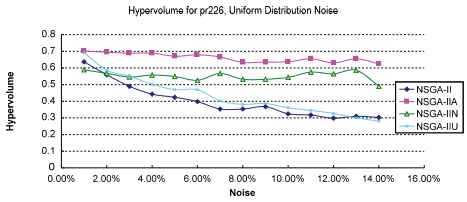


Fig. 9. Hypervolume (pr226, Uniform Distribution Noise)

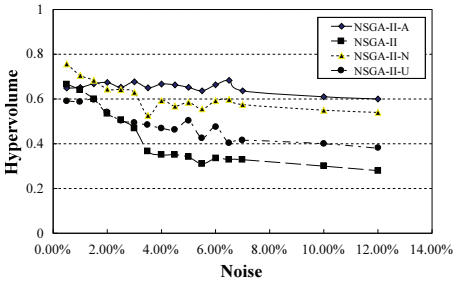


Fig. 10. Hypervolume (lin318, Normal Distribution Noise)

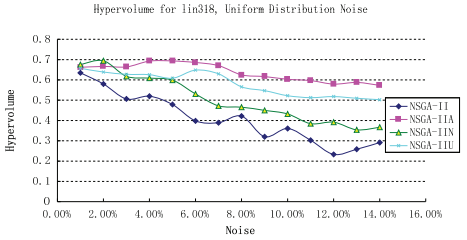


Fig. 11. Hypervolume (lin318, Uniform Distribution Noise)

NSGA-II-N. These results demonstrate that the α-dominance operator is less sensitive against noise levels than existing noise-aware dominance operators in NSGA-II-U and NSGA-II-N.

4.6 Evaluation of optimality and noise canceling with $D1_R$

Figures 12 to 17 show the $D1_R$ measures that individual algorithms yield in each problem with different noise distributions. As Figures 12, 14 and 16 depict, when normal distribution noise is injected to objective functions, NSGA-II-U performs poorly. This is understandable because it does not expect normal distribution noise at all. The other three algorithms perform similarly. Although NSGA-II-A is outperformed by NSGA-II-N and NSGA-II in the cs130 problem (Figure 12), it outperforms them in the other harder problems (the pr226 and lin318 problems; Figures 14 and 16). In the lin318 problem, which is hardest in the three test problems in this evaluation study, NSGA-II-A exhibits its superiority over NSGA-II-N and other algorithms.

When uniform distribution noise is injected to objective functions, NSGA-II-A outperforms the other three algorithms in all problems, although all algorithms perform similarly, particularly in the ch130 problem (Figures 13, 15 and 17). NSGA-II-A exhibits its superiority in the lin318 problem than the other two problems. These results illustrates that the α-dominance operator allows NSGA-II-A to successfully suppress the impacts of noise on the evolution/convergence of individuals and seek quality solutions toward the Pareto-optima.

As Figures 12 to 17 show, $D1_R$ measures grow in all algorithms when the amount of injected noise increases. This means that the evolution/convergence of individuals suffers a higher

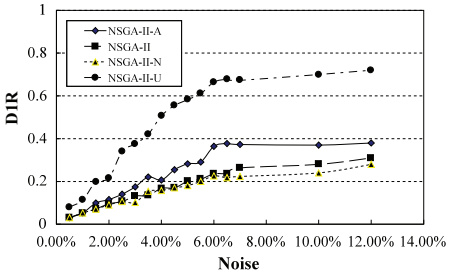


Fig. 12. $D1_R$ (ch130, Normal Dist. Noise)

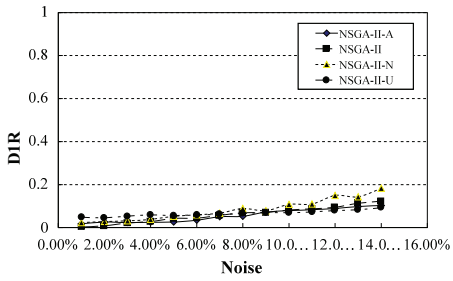


Fig. 13. $D1_R$ (ch130, Uniform Dist. Noise)

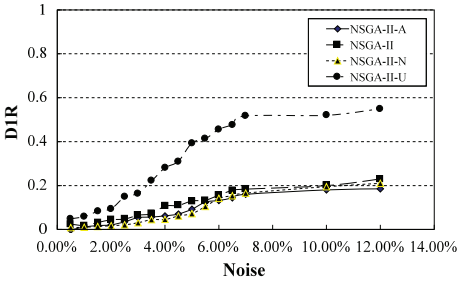


Fig. 14. $D1_R$ (pr226, Normal Dist. Noise)

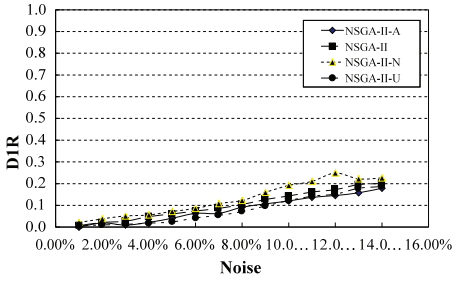


Fig. 15. $D1_R$ (pr226, Uniform Dist. Noise)

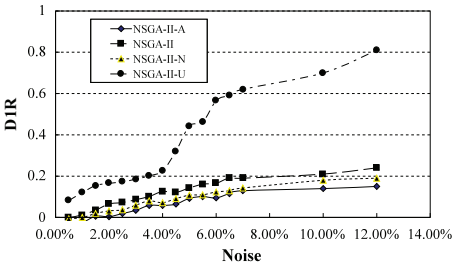


Fig. 16. $D1_R$ (lin318, Normal Dist. Noise)

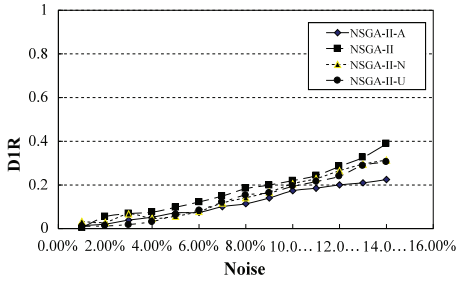


Fig. 17. $D1_R$ (lin318, Uniform Dist. Noise)

interference by a higher noise level. This trend is consistent with the results of hypervolume measures (Section 4.5). However, the $D1_R$ measures of NSGA-II-A do not vary largely under different noise levels. NSGA-II-A can maintain the $D1_R$ measure of approximately 0.2 or lower in all the cases except the ch130 problem with normal distribution noise (Figures12). This contrasts with, for example, NSGA-II-U that significantly increases $D1_R$ under higher noise levels in all problems. In the lin 318 problem with normal distribution noise, NSGA-II-A's $D1_R$ is under 0.2 under the highest noise level while NSGA-II-U's $D1_R$ exceeds 0.8. These results demonstrate that the α-dominance operator is less sensitive against noise levels than existing noise-aware dominance operators in NSGA-II-U and NSGA-II-N.

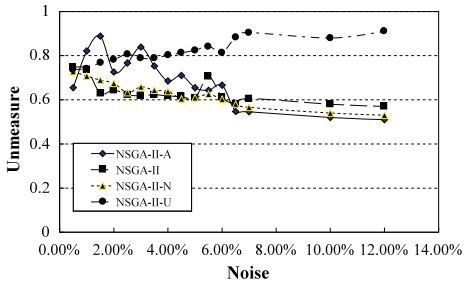


Fig. 18. U-metric (ch130, Normal Dist. Noise)

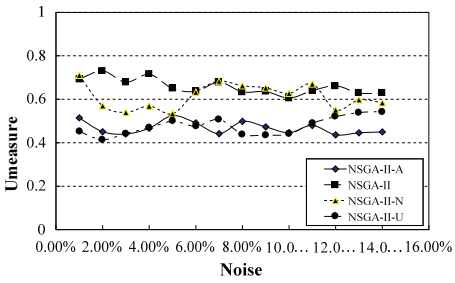


Fig. 19. U-metric (ch130, Uniform Dist. Noise)

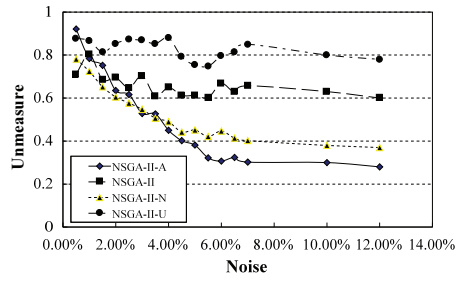


Fig. 20. U-metric (pr226, Normal Dist. Noise)

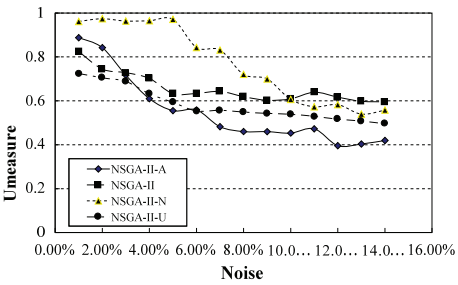


Fig. 21. U-metric (pr226, Uniform Dist. Noise)

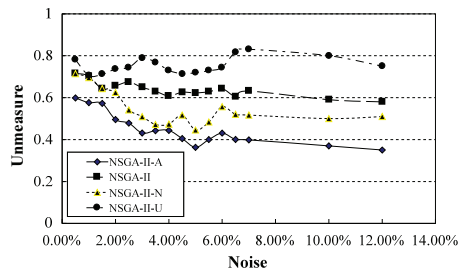


Fig. 22. U-metric (lin318, Normal Dist. Noise)

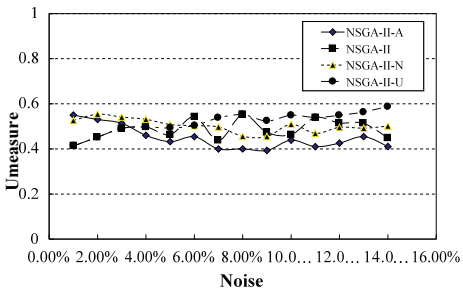


Fig. 23. U-metric (lin318, Uniform Dist. Noise)

4.7 Diversity evaluation with U-metric

Figures 18 to 23 show the U-metric measures that individual algorithms yield in each problem with different noise distributions. The four algorithms perform more similarly to each other in the problems with uniform distribution noise than normal distribution noise. In all cases, NSGA-II-A yields the best U-metric measures under the highest noise level. Considering that all algorithms perform the same (NSGA-II's) crowding distance operator for diversity preservation among individuals, the α-dominance operator produces the lowest interfere to diversity preservation based on crowding distance.

	Noise Level	$\mathcal{C}$(NSGA-II-A, NSGA-II)	$\mathcal{C}$(NSGA-II, NSGA-II-A)	$\mathcal{C}$(NSGA-II-A, NSGA-II-U)	$\mathcal{C}$(NSGA-II-U, NSGA-II-A)	$\mathcal{C}$(NSGA-II-A, NSGA-II-N)	$\mathcal{C}$(NSGA-II-N, NSGA-II-A)
ch130	Low	**4.23e − 03**	1.05e-03	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	Medium	**2.28e − 03**	0.00e+00	**3.21e − 03**	1.25e-03	0.00e+00	0.00e+00
	High	**1.20e − 03**	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
pr226	Low	**5.01e − 03**	0.00e+00	**5.23e − 03**	1.11e-03	0.00e+00	0.00e+00
	Medium	**0.21e − 03**	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	High	**3.10e − 03**	1.01e-03	0.00e+00	0.00e+00	0.00e+00	0.00e+00
lin318	Low	**8.27e − 03**	1.10e-03	**2.01e − 03**	0.00e+00	0.00e+00	0.00e+00
	Medium	**2.22e − 03**	0.00e+00	**1.00e − 03**	0.00e+00	0.00e+00	0.00e+00
	High	**9.36e − 03**	2.01e-03	0.00e+00	0.00e+00	0.00e+00	0.00e+00

Table 5. $\mathcal{C}$-metric Measures under Normal Distribution Noise

4.8 Algorithm comparison with C-metric

Tables 5 and 6 show the $\mathcal{C}$-metric measures to compare individual algorithms with each other under normal distribution noise and uniform distribution noise, respectively.

With normal distribution noise injected to objective functions (Tables 5), $\mathcal{C}$(NSGA-II-A, NSGA-II) > $\mathcal{C}$(NSGA-II, NSGA-II-A) in all of nine cases. (A bold font face is used to indicate a higher $\mathcal{C}$-metric measure between $\mathcal{C}$(NSGA-II-A, NSGA-II) and $\mathcal{C}$(NSGA-II, NSGA-II-A).) This means that NSGA-II-A outperforms NSGA-II in all the cases. In five of nine cases, NSGA-II produces no individuals that dominate the ones produced by NSGA-II-A. In comparison between NSGA-II-A and NSGA-II-U, $\mathcal{C}$(NSGA-II-A, NSGA-II-U) > $\mathcal{C}$(NSGA-II-U, NSGA-II-A) in three of nine cases. This means that NSGA-II-A outperforms NSGA-II-U in the three cases and the two algorithms tie in the other six cases. In seven of nine cases, NSGA-II-U produces no individuals that dominate the ones produced by NSGA-II-A. In comparison between NSGA-II-A and NSGA-II-N, the two algorithm tie in all nine cases. Even though NSGA-II-N is designed to handle normal distribution noise, it produces no individuals that dominate the ones produced by NSGA-II-A. Note that NSGA-II-A often outperforms NSGA-II-N in harder problems with higher normal distribution noise in terms of the number of non-dominated individuals (Section 4.4), hypervolume (Section 4.5) and $D1_R$ (Section 4.6).

	Noise Level	$\mathcal{C}$(NSGA-II-A, NSGA-II)	$\mathcal{C}$(NSGA-II, NSGA-II-A)	$\mathcal{C}$(NSGA-II-A, NSGA-II-U)	$\mathcal{C}$(NSGA-II-U, NSGA-II-A)	$\mathcal{C}$(NSGA-II-A, NSGA-II-N)	$\mathcal{C}$(NSGA-II-N, NSGA-II-A)
ch130	Low	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.01e-03	**2.01e − 03**
	Medium	**1.11e − 03**	0.00e+00	**4.01e − 03**	0.00e+00	**2.10e − 03**	1.30e-03
	High	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**2.58e − 03**	0.00e+00
pr226	Low	**1.22e − 03**	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	Medium	**4.29e − 03**	0.00e+00	**1.01e − 03**	0.00e+00	**6.07e − 03**	3.12e-03
	High	**5.18e − 03**	2.14e-03	**0.89e − 03**	0.00e+00	**1.00e − 03**	0.00e+00
lin318	Low	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**2.01e − 03**	0.00e+00
	Medium	**12.28e − 03**	0.00e+00	0.00e+00	0.00e+00	**6.31e − 03**	0.00e+00
	High	**2.33e − 03**	0.00e+00	**5.14e − 03**	1.02e-03	**1.08e − 03**	0.00e+00

Table 6. $\mathcal{C}$-Metric Measures under Uniform Distribution Noise

With uniform distribution noise injected to objective functions (Tables 6), NSGA-II-A outperforms NSGA-II in six of nine cases. In eight cases, NSGA-II produces no individuals that dominate the ones produced by NSGA-II-A. In comparison between NSGA-II-A and NSGA-II-U, which assumes uniform distribution noise in advance, NSGA-II-A outperforms NSGA-II-U in four of nine cases, and the two algorithms tie in the other five cases. In eight

cases, NSGA-II-U produces no individuals that dominate the ones produced by NSGA-II-A. A general observation is that NSGA-II-A outperforms NSGA-II-U in harder problems with higher uniform distribution noise. It is consistent with the observations made with the number of non-dominated individuals (Section 4.4), hypervolume (Section 4.5) and $D1_R$ (Section 4.6). In comparison between NSGA-II-A and NSGA-II-N, NSGA-II-A outperforms NSGA-II-N in seven of nine cases. In six cases, NSGA-II-N produces no individuals that dominate the ones produced by NSGA-II-A.

As Tables 5 and 6 illustrate, NSGA-II-N performs better under normal distribution noise than uniform distribution noise. This is reasonable because it anticipates normal distribution noise in advance. However, it never outperforms NSGA-II-A in Table- 5). On the contrary, NSGA-II-U performs poorly under both normal and uniform distribution noise. It never outperforms NSGA-II-A in Tables 5 and 6. These results demonstrate that, although the α-dominance operator assumes no noise distribution in advance, it performs under both normal and uniform distribution noise. Moreover, it exhibits higher superiority under higher noise levels.

4.9 Optimality evaluation with objective values

Tables 7 and 8 show the objective values that each algorithm's individuals yield at the last (the 500th) generation under normal and uniform distribution noise, respectively. L, M and H mean that low, medium and high levels of noise. Each table shows the average and standard deviation results that are obtained from 20 independent experiments. A bold font face is used to indicate the best average result among four algorithms on an objective by objective basis. An asterisk (*) is placed for an average result when the result is significantly different (worse) than the best average result based on a t-test with 19 degrees of freedom and 95% confidence level.

Table 7 depicts that, in the ch130 problem, NSGA-II-A is the best among four algorithms in cost under the low noise level, in profit under the medium noise level, and in both objectives under the high noise level. Under the high noise level, NSGA-II-A exhibits statistical significance over NSGA-II and NSGA-II-N in cost and over NSGA-II and NSGA-II-U in profit. In the pr220 problem, NSGA-II-A is the best among four algorithms in cost under all noise levels. It is also the best in profit under the low and high levels of noise. Under the high noise level, NSGA-II-A exhibits statistical significance over NSGA-II-N and NSGA-II-U in cost and over NSGA-II-R and NSGA-II-U in profit. Moreover, NSGA-II-A yields the lowest standard deviation in both objectives in all noise levels. This means that the variance of it's objective values is the lowest among different experiments. In the lin318 problem, NSGA-II-A yields the best objective values and the best standard deviation values in both objectives under all noise levels. Under the high noise level, it exhibits statistical significance over NSGA-II and NSGA-II-U in cost and over all the other three algorithms in profit. Table 7 demonstrates that the α-dominance operator allows NSGA-II-A to outperform the other algorithms more often in more metrics as a given problem becomes harder and a given noise level becomes higher.

Table 8 shows qualitatively similar results to the ones in Table 7. In the lin318 problem with the highest noise level, NSGA-II-A yields the best objective values and the best standard deviation values in both objectives under all noise levels. It exhibits statistical significance over NSGA-II and NSGA-II-N in cost and over all the other three algorithms in profit.

		ch130				pr220				lin318			
		Cost		profit		Cost		profit		Cost		profit	
		Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd
L	A	**2533.8**	174.6	3129.0⋆	34.2	**92236.2**	181.2	**6828.3**	82.4	**45353.2**	204.3	**8852.7**	111.0
	R	2890.5⋆	167.9	3124.3	46.4	116225.3⋆	192.5	6826.9	95.3	47927.6	248.2	8748.9	118.3
	N	2646.2⋆	186.7	**3208.8**	49.4	104295.2⋆	188.7	6806.4	107.4	48918.3⋆	212.2	8398.4⋆	152.2
	U	2760.1⋆	214.3	2958.6⋆	39.2	114219.1⋆	214.0	6485.0⋆	111.0	50688.4⋆	302.3	8539.0⋆	185.8
M	A	3471.6	244.8	**3855.1**	79.2	**132637.8**	242.4	7077.6	93.1	**45894.5**	402.3	**8740.5**	185.4
	R	**3280.9**	272.2	3768.2	76.4	136017.3	290.4	**7153.1**	142.6	56162.1⋆	499.8	8520.8⋆	321.0
	N	3971.8⋆	266.1	3685.7⋆	76.9	135070.4	345.7	6944.9⋆	112.6	56649.0⋆	461.0	8522.2⋆	201.8
	U	3698.3⋆	291.9	3602.3⋆	85.8	139797.3⋆	293.2	6920.4⋆	150.2	61204.3⋆	481.4	8622.2	342.4
H	A	**4564.2**	312.5	**3734.6**	84.1	**148905.8**	243.2	**6752.2**	101.0	**56227.3**	527.4	**8504.8**	385.1
	R	4740.3⋆	248.5	3534.6⋆	93.1	150162.1	277.8	6557.8⋆	161.4	70204.6⋆	599.4	8361.9⋆	414.4
	N	4702.5⋆	324.2	3656.2	89.4	171468.0⋆	267.3	6733.6	143.3	56450.7	582.2	8292.8⋆	442.1
	U	4615.1	352.8	3593.6⋆	98.7	171936.6⋆	391.0	6436.6⋆	165.0	66661.1⋆	701.3	8258.3⋆	499.5

Table 7. Objective Values at the Last Generation under Normal Distribution Noise

		ch130				pr220				lin318			
		Cost		profit		Cost		profit		Cost		profit	
		Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd
L	A	**2368.5**	242.4	3139.9	183.1	**99782.9**	144.2	**6949.3**	131.0	**41937.1**	309.7	**9010.6**	119.9
	R	2777.2⋆	223.6	3167.6	184.1	105385.1⋆	214.0	6566.0⋆	152.1	43865.8	351.9	9000.9	134.7
	N	2466.0	265.9	2895.2⋆	100.3	122022.7⋆	183.5	6779.7⋆	141.1	53194.4⋆	408.6	8937.6⋆	177.4
	U	2860.1⋆	281.2	**3252.0**	78.6	110096.8⋆	188.3	6920.4	158.8	50819.3⋆	469.5	8805.6⋆	102.4
M	A	**2976.3**	203.2	3117.2	146.3	**97879.5**	248.3	**7137.8**	136.8	**48109.6**	677.9	**8829.7**	87.7
	R	3705.6⋆	296.2	2666.7⋆	145.6	110639.5⋆	321.0	6925.2⋆	148.5	53918.6⋆	686.5	8374.6⋆	123.6
	N	3573.3⋆	210.3	2709.3⋆	163.5	119108.1⋆	295.4	6680.0⋆	146.1	53346.2⋆	707.5	8442.0⋆	118.4
	U	3368.7	189.3	**3344.5**	135.3	122815.9⋆	274.1	6816.6⋆	148.5	53640.5⋆	706.9	8519.1⋆	171.0
H	A	4384.6	404.1	**3838.0**	137.7	**151578.5**	381.1	**7022.4**	132.2	**61174.5**	439.8	**8887.9**	114.8
	R	4463.5⋆	426.7	2798.0⋆	142.6	164282.2⋆	458.4	6523.2⋆	139.8	66035.3⋆	647.3	8152.2⋆	267.4
	N	4992.5⋆	445.8	2730.0⋆	152.8	178198.2⋆	422.2	6830.6⋆	136.9	62668.5⋆	553.8	8566.8⋆	168.0
	U	**4299.3**	499.1	3330.0⋆	163.6	166907.1⋆	461.9	6711.4⋆	142.5	61701.4	520.3	8285.4⋆	189.1

Table 8. Objective Values at the Last Generation under Uniform Distribution Noise

5. Related work

pTSP and TSPP have been studied extensively and used to model many real-world applications in different fields (Feillet et al., 2005). Early pTSP studies adopted heuristics that were modified from the heuristics to solve TSP (e.g., nearest neighbor, savings heuristic, k-opt exchanges, 1-shift) (Bertsimas, 1988; Birattari et al., 2007). Recent studies often focus on meta-heuristics, such as ant colony optimization algorithms (Branke & Guntsch, 2004) and evolutionary algorithms (Liu, 2008; Liu et al., 2007), in favor of their global search capabilities. However, these algorithms are not applicable for pTSPP because pTSP is a single objective optimization problem and pTSPP is a multiobjective optimization problem as described in Section 2.

TSPP is a multiobjective optimization algorithm; however, a number of existing work have attempted to solve it as a single objective optimization problem by aggregating multiple objectives into a single fitness function as, for example, a weighted sum of objective values or considering extra objectives as constraints with given bounds (Awerbuch et al., 1999; Laporte & Martello, 1990). These algorithms are not designed to seek the optimal tradeoff (i.e., Pareto-optimal) solutions among conflicting objectives. Moreover, it is not always straightforward to manually tune weight values in a fitness function that aggregates multiple objective values.

A very limited number of existing work have attempted to solve TSPP with multiobjective optimization algorithms (Jozefowiez et al., 2008b). These algorithms better address the characteristics of pTSPP; however, they never consider noise in objective functions.

The α-dominance operator is designed to aid seeking the Pareto-optimality of solution candidates in multiobjective optimization problems with noisy objective functions. In the area of evolutionary multiobjective optimization, there exist several existing work to handle uncertainties in objective functions by modifying NSGA-II's classical dominance operator

(Beyer, 2000; Jin & Branke, 2005). All of them assume particular noise distributions in advance. For example,Babbar et al. (2003); Eskandari et al. (2007); Goh & Tan (2006) assume normal distribution noise. Teich (2001) assume uniform distribution noise. Delibrasis et al. (1996); Wormington et al. (1999) assume Poisson distribution noise. Given a noise distribution, each of existing noise-aware dominance operators statistically estimates each individual's objective value by collecting its samples. In contrast, the α-dominance operator assumes no noise distributions a priori because, in general, it is hard to predict and model them in most (particularly, real-world) multiobjective optimization problems. Instead of estimating each individual's objective values, the α-dominance operator estimates the impacts of noise on objective value samples and determines whether it is confident enough to compare individuals.

Another line of relevant research is to handle uncertainties in decision variables (Deb & Gupta, 2006; Deb et al., 2009; 2006). These work proposes the notion of robust individuals, and the robustness quantifies the sensitivity of noise in the decision space on the objective space. They also assume normal distribution noise in advance. Unlike these work, α-dominance focuses on uncertainties in the objective space and assumes no noise distributions in advance.

6. Conclusions

This chapter formulates a noisy multiobjective optimization problem, the Probabilistic Traveling Salesman Problem with Profits (pTSPP), which contains noise in its objective functions. In order to solve pTSPP, this chapter proposes an evolutionary multiobjective optimization algorithm (EMOA) that leverages a novel noise-aware dominance operator, called the α-dominance operator. The operator takes objective value samples of given two individuals, estimates the impacts of noise on the samples and determines whether it is statistically confident enough to judge which individual is superior/inferior to the other. Experimental results demonstrate that the α-dominance operator allows the proposed EMOA to effectively obtain quality solutions to pTSPP and it outperforms existing noise-aware dominance operators.

7. References

Arnold, D. (2000). Evolution strategies in noisy environments – a survey of existing work, *in* L. Kallel, B. Naudts & A. Rogers (eds), *Theoretical Aspects of Evolutionary Computing*, Springer, chapter 11.

Awerbuch, B., Azar, Y., Blum, A. & Vempala, S. (1999). New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen, *SIAM J. Comput.* 28(1): 254–262.

Babbar, M., Lakshmikantha, A. & Goldberg, D. E. (2003). A modified NSGA-II to solve noisy multiobjective problems, *Proc. of ACM Genetic and Evol. Comp. Conf.*

Bertsimas, D. (1988). Probabilistic combinatorial optimization problems, *Ph.D dissertation, MIT, USA* .

Bertsimas, D. & Howell, L. H. (1993). Further results on the probabilistic traveling salesman problem, *European Journal of Operational Research* 65(1): 68–95.

Beyer, H.-G. (2000). Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice, *Elsevier Computer Methods in Applied Mechanics and Engineering* 186(2–4).

Beyer, H.-G. & Sendhoff, B. (2007). Robust Optimization – A Comprehensive Survey, *Computer Methods in Applied Mechanics and Engineering* 196(33-34): 3190–3218.

Bianchi, L., Dorigo, M., Gambardella, L. & Gutjahr, W. (2009). A Survey on Metaheuristics for Stochastic Combinatorial Optimization, *Natural Computing* 8(2): 239–287.

Birattari, M., Balaprakash, P., Stï£¡ï£¡tzle, T. & Dorigo, M. (2007). Estimation-based local search for the probabilistic traveling salesman problem, *MIC07: The seventh Metaheuristics International Conference* .

Branke, J. & Guntsch, M. (2004). Solving the probabilistic tsp with ant colony optimization, *Journal of Mathematical Modelling and Algorithms* 3: 403–425.

Carroll, R. J., Ruppert, D., Stefanski, L. A. & Crainiceanu, C. (2006). *Measurement Error in Nonlinear Models: A Modern Perspective, Second Edition*, CRC Press.

Deb, K., Agrawal, S., Pratab, A. & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, *Proc. of Int'l Parallel Problem Solving from Nature Conf.*

Deb, K. & Gupta, H. (2006). Introducing robustness in multi-objective optimization, *Evol. Comput.* 14: 463–494.

Deb, K., Gupta, S., Daum, D., Branke, J., Mall, A. K. & Padmanabhan, D. (2009). Reliability-based optimization using evolutionary algorithms, *Trans. Evol. Comp* 13: 1054–1074.

Deb, K., Padmanabhan, D., Gupta, S. & Mall, A. K. (2006). Handling uncertainties through reliability-based optimization using evolutionary algorithms, *Technical Report KanGAL 2006009*, Indian Institute of Technology Kanpur.

Delibrasis, K., Undrill, P. & Cameron, G. (1996). Genetic algorithm implementation of stack filter design for image restoration, *Proc. of Vision, image and signal proc.*

Diwekar, U. & Kalagnanam, J. (1997). Efficient Sampling Technique for Optimization under Uncertainty, *Wiley AIChE J.* 43(2).

Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro, B. & Alba, E. (2006). jMetal: A java framework for developing multi-objective optimization metaheuristics, *Technical Report ITI-2006-10*, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga.

Eskandari, H., Geiger, C. D. & Bird, R. (2007). Handling uncertainty in evolutionary multiobjective optimization: SPGA, *Proc. of IEEE Congress on Evol. Comp.*

Feillet, D., Dejax, P. & Gendreau, M. (2005). Traveling salesman problems with profits, *INFORMS Transportation Science* 39(2): 188–205.

Goh, C. K. & Tan, K. C. (2006). Noise handling in evolutionary multi-objective optimization, *Proc. of IEEE Congress on Evol. Comp.*

Goldbert, D. E. & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem, *Proc. of the Second Int. Conf. on Genetic Algorithms and Their Applications*, pp. 154–159.

Jaillet, P. (1985). *Probabilistic Traveling Salesman Problem*, PhD thesis, Massachusetts Institution of Technology.

Jin, Y. & Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey, *IEEE Trans. on Evol. Comp.* 9.

Jozefowiez, N., Glover, F. & Laguna, M. (2008a). Multi-objective meta-heuristics for the traveling salesman problem with profits, *Springer Journal of Mathematical Modelling and Algorithms* 7(2): 177–195.

Jozefowiez, N., Glover, F. & Laguna, M. (2008b). Multi-objective meta-heuristics for the traveling salesman problem with profits, *Journal of Mathematical Modelling and Algorithms* 7: 177–195.

Kellegőz, T., Toklu, B. & Wilson, J. (2008). Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem, *Applied Mathematics and Computation* 199(2): 590 – 598.

Knowles, J. & Corne, D. (2002). On metrics for comparing nondominated sets, *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, Vol. 1, pp. 711 –716.

Laporte, G. & Martello, S. (1990). The selective travelling salesman problem, *Discrete Appl. Math.* 26(2-3): 193–207.

Leung, Y.-W. & Wang, Y.-P. (2003). U-measure: A quality measure for multiobjective programming, *Technical Report COMP-03-011*, Hong kong Baptist University.

Liu, Y.-H. (2008). *Solving the Probabilistic Traveling Salesman Problem Based on Genetic Algorithm with Queen Selection Scheme*, InTech, chapter Traveling Salesman Problem.

Liu, Y.-H., Jou, R.-C., Wang, C.-C. & Chiu, C.-S. (2007). An evolutionary algorithm with diversified crossover operator for the heterogeneous probabilistic tsp, *MDAI '07: Proceedings of the 4th international conference on Modeling Decisions for Artificial Intelligence*, pp. 351–360.

Reinelt, G. (1991). Tsplib – a traveling salesman problem library, *ORSA Journal on Computing* 3: 376.

Srinivas, N. & Deb, K. (1995). Multiobjective function optimization using nondominated sorting genetic algorithms, *MIT Evol. Comp. J.* 2(3).

Teich, J. (2001). Pareto-front exploration with uncertain objectives, *Proc. of Int'l Conf. on Evol. Multi-Criterion Optimization*.

Wormington, M., Panaccione, C., Matney, K. M. & Bowen, D. K. (1999). Characterization of structures from x-ray scattering data using genetic algorithms, *JSTOR Philosophical Transactions* 357(1761).

Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, *IEEE Trans. Evol. Comp.* 3(4).

15

Scheduling of Construction Projects with a Hybrid Evolutionary Algorithm's Application

Wojciech Bożejko[1], Zdzisław Hejducki[1],
Magdalena Rogalska[2] and Mieczysław Wodecki[3]
[1]*Wrocław University of Technology, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27,*
[2]*Lublin University of Technology,20-950 Lublin, Nadbystrzycka 40,*
[3]*Wrocław University, 50-383 Wrocław, Joliot-Curie 15,*
Poland

1. Introduction

In the works [5,16] there is presented a possibility of application of critical chain scheduling /buffer management methodology (CCS/BM) in investment enterprise and construction planning. The completion of a civil structure or a building complex is an undertaking consisting of the following factors: fulfilling the requirements (quality), cost, time of execution, range and resources [7]. In the article there are presented the outcomes of the tests concerning an improvement of methods used for investment scheduling and construction project with the implementation of mementic algorithm (i.e. hybrid evolutionary, HEA, [1]). The aim of the work was finding an optimal (for a taken goal function) level of workers' employment that is minimization of a divergence from the average level of employment with the implementation of CCS/BM methodology [5].
Scheduling of investment and construction project is connected to an optimization task. It is related to finding the best solution fulfilling the constraining conditions and taking into consideration the goal function. There are many known methods of optimization applied in specific cases.
Among others, there can be mentioned, for instance, for continuous tasks – methods of linear simplex, tasks of global optimization – when a goal function in the field of accepted solutions has more than one local minimum, discrete tasks with a greater complexity of calculations based most commonly on division or constraint methods, non – determined methods using random generating of solutions, a simulated annealing method as a modification of random walk with the improvement of quality of goal function, tabu search, that is with a list of revised variants and others.
There are also techniques applied with the use of biological systems – evolutionary algorithms, genetic, evolutionary strategies, evolutionary programming and genetic programming ([3],[18]). The general scheme of evolutionary algorithm's operation resides in creating a loop embracing reproduction, genetic operations, evaluation and succession. The classic scheme of operation of evolutionary algorithm is presented below according to [3].
This paper is a continuation of the topic presented in work [15], concerning the application of genetic algorithms [7] to steering of a level of an employment in investment and construction projects. Treating the evolutionary algorithm as a typical method of proceeding concerning

searching for better solutions, taking into consideration the closest environment, there was a slightly different approach proposed than the one applied in modeling of evolution with a application of a binary genetic code. The new one consists in replicating of randomly chosen variants (individuals) with a possibility of multiple copying of the same solution, in which the random choice takes into account a better adaptation of a variant (individual).
Solving practical issues of operational tasks, including scheduling of investment and construction project requires a selection of such an optimization method which leads to the best solution with the minimal cost. It is assumed [3], that one of the most efficient optimization algorithms is the evolutionary algorithm with strategy, constituting frameworks for an idea of a mementic algorithm HEA, [1].

2. Defining the problem

An optimization task refers to planning of regularity of workers' employment level [8,17,14,11,10] during realization of construction undertaking with regard towards CCM/BM methodology [5,16]. Time buffers introduced to scheduling task increase space of accepted variations of realizing the project and an extent of a task.
In order to solve an optimization task a hybrid evolution algorithm HEA created by Bożejko and Wodecki [1] was applied. The idea of a algorithm HEA consists of creation of a start-up population in which for every individual there is a permutation applied in order to find a local minimum. Then, there is a passing towards separate populations with defining a number and position in a set of individuals.
Theory of Constrains (TOC) by Goldratt [2,3,4,5,6] and its practical application in managing of projects known as Critical Chain Scheduling (CCS) and Buffer Management (BM), in short defined as CCS/BM Method [7], stays in the center of interest of many scientific goups. Precursors of this methods were Giffler and Thomson [8] together with Wiest [17] who introduced the concept of critical sequences determined not only by technological sequencing and adjusting the time of tasks' completion but also by constraints of resources in creation of schedules. Creating and steering of schedules is one of the aims of managing building projects. High level of simplification of the two systems: Critical Path Method (CPM) together with Program Evaluation and Review Technique (PERT) led to their popularizing and common use all over the world. Changing standards, growing expectations, cost cutting, minimizing fines and the extent of investment projects, clearness of expressions led to undertaking of many works aiming at increasing the effectiveness of scheduling treated in relation to traditional CPM/PERT methods.

2.1 Theory of constrains framework

Theory of Constraints (TOC) can be applied in all projects which aim at reaching revenue. TOC is based on five basic steps:

1. Identification system constraints.
2. Decision of maximum utilization of constrained resources.
3. Subordination of all processes to above decision.
4. Increasing number of constraints created as a result of a liquidation of a constraint defined in the first step as a bottleneck of the system.
5. Identification of new constraints of resources created as a result of bottleneck elimination in point 4; if constraint in point 4 is eliminated then return to step one in order not to allow any internal factors to constraint the whole system.

The proposed system aims at ongoing production improvement in order to gain bigger profit in current and future undertakings through identifying and eliminating bottlenecks in production. TOC can be used in realization of building projects. Let us assume that in the first step a tower crane is identified as a constrained resource. Thus, in the second step we take a decision of maximum utilization of crane's capacity shifting work schedule from 8 to 16 working hours [14]. In the third step, as a consequence of the taken decision, we must subordinate all teams and machines which cooperate with the crane. In the fourth step we analyze whether the resources pose a threat of becoming bottlenecks of the system. In case when internal factors constrain the whole system, we must return to step one and identify constraints one more time. While implementing a production improvement in accordance with TOC we can expect an increase of profits. The example of practical application of Theory of Constraints is a project management method – scheduling method using critical chain and buffer management CCS/BM.

2.2 CCS/BM methodology

Chronic problems appearing in realization of construction tasks which cannot be removed even with the use of advanced technologies became a reason for development of CCS/BM methods [11]. Classic scheduling methods used in construction projects CPM/PERT are characterized by utilization of time used for completing separate activities with regard to time limits resulting from valid, approximate, standard data. Goldratt perceives such reasoning as inappropriate and in his work [2] even calls this process 'a thief of time', whereas Turner [16] analyzes the influence of time reserve dispersion on revenue gained from undertaking. He assumes that a project consists of n tasks and each of them is afflicted with a contingency (randomness). Moreover, on the assumption that there is an even scheduling of time needed for task completion, it has a standard σ deviation. Thus, an overall contingent reserve equals $n\sigma$. The rights to manage it are dispersed. Every participant of the project can manage their time reserve. From the point of view of a project manager an optimal situation would be if he could manage the time himself for the following reasons: reserves not utilized by separate participants of activities would not get wasted and total time reserve could be smaller than in the first case. From the calculations presented by Turner it results that a total time reserve should equal $(n)^{1/2}\sigma$ which is a much smaller value than $n\sigma$.

Implementation of a method of time reserve aggregation results in a significant reduction of project costs. Moreover, lack of time reserve for a project manager often leads to a failure in meeting the deadlines. Goldratt recalls some known examples: a tunnel under British Channel or drilling towers in the North Sea. The size of time reserves and duration of separate activities (not taking into consideration its internal reserves) is one of the basic problems of CCS/BM. During developing of CCS/BM method Goldratt was basing on the following psychological assumptions:

- Student Syndrome – 'do not begin work before all possibilities and time limits are used';
- Parkinson's Law – 'every work will be done in a assigned time or longer';
- roadrunner mentality – 'real race against time', [7];
- Conkling's Roadunner – *Geococcyx californianus* is the quickest runner among its species reaching the speed of 30 km/h, it never moves slowly;
- Murphy's Law – 'anything that can go wrong, will go wrong'.

Goldratt proposes time reduction for a completion of an individual processes together with informing tasks executors only about due dates. A schedule including time buffers is accessible exclusively for a project manager. Participants, not possessing any time reserves, try to complete their tasks as quick as possible (roadrunner mentality), start their job with a full capacity (Student Syndrome), in case of threat of falling behind the deadline (Murphy's Law & Parkinson's Law) the project manager has time reserves to modify and steer the course of works.

Goldratt solves the problem of time reserves in an arbitrary manner. Namely, he introduces reduction of an activity duration by half (50%) and creates time buffers of a 50% value of a new, shortened activity duration. In reference to sub-critical chains Goldratt calls it a feeding buffer FB, whereas behind a critical chain he places a project buffer PB. According to the following assumptions:

- **Project buffer PB** is a time reserve placed at the end of critical chain, staying to project managers disposal, introduced in order to protect completion of a project, calculated on the basis of critical chain time duration (according to Goldratt a project buffer is 25% of critical chain duration), there is only one project buffer in scheduling.
- **Feeding Buffer FB** is a time reserve placed at the end of non-critical chain (feeding chain) staying to managers disposal in order to protect tasks placed in a critical chain, introduced as a protection of deadline of critical tasks, calculated on the basis of non-critical chain duration (in Goldratt's method a feeding buffer equals 25% of non-critical chain duration), there are as many feeding buffers as the number of non-critical chains (FB appears always when a non-critical chain links with a critical chain).
- **Resource buffer RB** is a time reserve placed in schedule before entering of a new resource into a critical chain, calculated on the basis of logistic dependences and possibilities, introduced in order to ensure initiation of a process in a critical chain, in a planned due date.
- **Critical chain CC** is a set of processes appearing in front of a project buffer, determined by: activities' duration time,, their technical sequence of completion, accessibility and resource requirements; processes in a critical chain are deprived of time reserves taken into consideration in CPM/PERT (according to Goldratt the length of a critical chain is calculated as 50% of critical path CP); in a critical chain cannot appear any processes not having a direct influence on due task time (owing to time, technical or resource constraints); in case of appearing of more than one critical path one must choose one of them and transform a critical chain (there can be only one critical chain).

Figure 1 presents a graph of activities 1-2-3-4-5 lasting adequately 16,4,8,8,4 time units. There are dependences between activities shown and their sequence has been marked in figure 1.

According to CCS/BM methodology and applying Goldratt's assumption concerning a duration of time of processes and length of feeding (FB) and project buffer (PB) we obtain a graph depicted in figure 2.

An individual assessment, based on system analysis of shortening separate processes, creation of critical chain and size of feeding and project buffers allows for creation of rational schedule including not only technical but also resources, organizational, financial constraints. Dependencies between process duration time, critical chain buffer and optimal utilization of resources with application of mementic algorithm (HEA) to calculations have been thoroughly examined. Expansion of implemented in financial calculations method of contingency onto area of scheduling of construction projects with application of TOC and CCS/BM methods accompanied by HEA might lead to shortening of time and costs of a

building process. Works concerning this problem will be continued in the future by the authors.

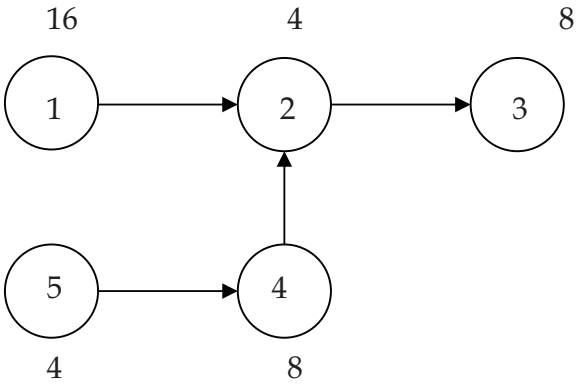


Fig. 1. Graph of activities and times of their duration in a classic formulation.

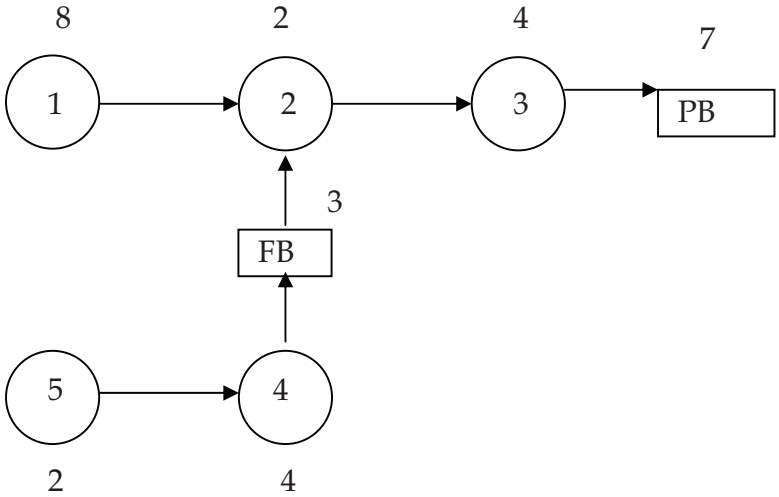


Fig. 2. Graph of activities and time of their duration according to CCS/BM including project (PB) and feeding buffers (FB).

3. Hybrid Evolutionary Algorithm

Hybrid evolutionary algorithm was proposed by Bożejko and Wodecki [1] and it is a general method of solving discreet optimization problems. Therefore some elements of the algorithm have to be addressed in detail to use it for solving automation problems in construction, especially the method of the problem's code for HEA, determining of set of fixed elements and local optimization approach.

The algorithm starts by forming residual population P^0 (which can be randomly formed). The best element of population P^0 is adopted as suboptimum solution π^*. Let i be the algorithm iteration number. New population i+1 (i.e. set P^{i+1}) is generated as follows. For

current population P^i a set of local minima LM^i is fixed (by carrying out procedure $LocalOpt(\pi)$ for each element $\pi \in P^i$). Elements occupying the same positions in the local minima are fixed (procedure $FixeSet(LM^i, FS^i)$), forming a set of fixed elements and positions FS^{i+1}. Each permutation of new population P^{i+1} has fixed elements (in fixed positions) from set FS^{i+1}. Free elements are randomly assigned to the remaining (unoccupied) positions. If there is a permutation $\beta \in LM^i$ and $F(\beta) < F(\pi^*)$, then β is adopted as permutation π^*. The algorithm stops when it has generated a predetermined number of generations.
We apply following notation:

π^*	:	sub-optimal solution determined by the algorithm,
η	:	number of elements in population (the same in each generation),
P^i	:	population in the iteration *i* of algorithm, $P^i = \{\pi_1, \pi_2, \dots, \pi_\eta\}$,
$LocalOpt(\pi)$	:	local optimization algorithm to determining local minimum, where π is a starting solution of the algorithm,
LM^i	:	a set of local minima in iteration *i*, $LM^i = \{\hat{\pi}_1, \hat{\pi}_2, \dots, \hat{\pi}_\eta\}$ where $\hat{\pi}_j = LocalOpt(\pi_j), \quad \pi_j \in P^i, \; j = 1, 2, \dots, \eta.$
FS^i	:	a set of fixed elements and position in permutations of population P^i,
$FixSet(LM^i, FS^i)$	:	a procedure which determines a set of fixed elements and positions in next iteration of evolutionary algorithm, ,
$NewPopulation(FS^i)$	:	a procedure which generates a new population in next iteration of the algorithm, $P^{i+1} = NewPopulation(FS^i)$.

The code of the proposed hybrid evolutionary algorithm is given below and figure 3.

Algorithm (HEA)
Initialization:
randomly formed population $P^0 = \{\pi_1, \pi_2, \dots, \pi_\eta\}$;
π^* = the best element of population P^0;
Iteration number i=0;
$FS^0 = \varnothing$;
repeat
Determine a set of local minima $LM^i = \{\hat{\pi}_1, \hat{\pi}_2, \dots, \hat{\pi}_\eta\}$, where
$\hat{\pi}_j = LocalOpt(\pi_j), \quad \pi_j \in P^i$;
for j:=1 **to** η **do**
if $F(\hat{\pi}_j) < F(\pi^*)$ **then** $\pi^* \leftarrow \hat{\pi}_j$;
$FS^{i+1} = FixSet(LM^i, FS^i)$; {*fix set*}
$P^{i+1} := NewPopulation(FS^i)$; {*generate new population*}
i=i+1;
not until **S**top **C**riterion (exceeding time or a number of iterations).

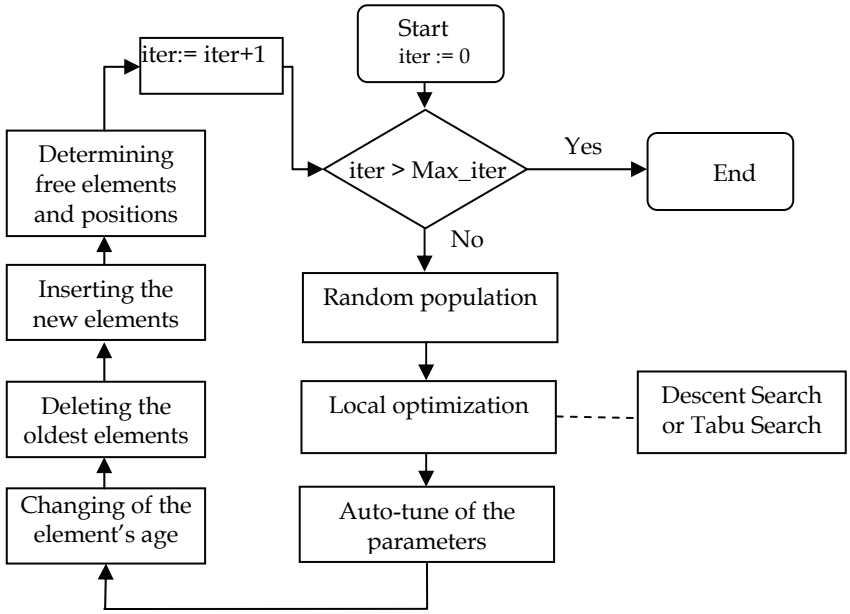


Fig. 3. Hybrid evolutionary algorithm.

4. Problem coding and notation

The problem can be defined as follows. There are: a set of n jobs $J=\{1,2,\ldots,n\}$, a set of m machines $M=\{1,2,\ldots,m\}$. Job $j\in J$, consists of a sequence of m operations O_{j1}, O_{j2},…, O_{jm}. Operation O_{jk} corresponds to the processing of job j on machine k during an uninterrupted processing time p_{jk}. We want to find a schedule such that the maximum completion times is minimal. Such a problem is known as a flow shop problem in literature.

Let $\pi=(\pi(1), \pi(1),\ldots,\pi(n))$ be a permutation of jobs $\{1,2,\ldots,n\}$ and Π be the set of all permutations. Each permutation $\pi\in\Pi$ defines a processing order of jobs on each machine. Completion time of job $\pi(j)$ on machine k can be found using the recursive formula:

$$C_{\pi(j)k}=\max\{C_{\pi(j-1)k}, C_{\pi(j)k-1}\}+p_{\pi(j)k}\},$$

where $\pi(0)=0$, $C_{0k}=0$, $k=1,2,\ldots,m$ and $C_{0j}=0$, $j=1,2,\ldots,n$.

Local optimization (**procedure** *LocalOpt*)

The local search (*LS*) method is a metaheuristic approach designed to find a near-optimal solution of combinatorial optimization problems. The basic version of *LS* starts from an *initial solution* x^0. The elementary step of the method performs, for a given solution x^i, a search through the *neighborhood* $N(x^i)$ of x^i. The neighborhood $N(x^i)$ is define by move performed from x^i. A move transforms a solution into another solution. The aim of this elementary search is to find in $N(x^i)$ a solution x^{i+1} with the lowest cost functions. Then the search repeats from the best found, as a new starting solution.

Local search algorithm

Select a starting point: x;

$x_{best} := x$;

repeat

Select a point $y \in N(x)$ according to the given criterion based on the value of the goal function $F(y)$;

$x := y$;

if $F(y) > F(x_{best})$ **then** $x_{best} := y$;

until some termination condition is satisfied.

A fundamental element of the algorithm, which has a crucial influence on quality and time of computation, is a neighborhood. A neighborhood is generated by the insert moves in the best local search algorithms with the permutation representation of the solution.

A set of fixed elements and position (procedure *FixSet*)

A set of fixed elements and positions FS^i (in *i*-th iteration of the algorithm) consists of quads (a,l,α,φ), where a is an element of the set N ($a \in N$), l is a positions in a solution ($1 \le l \le n$) and α, φ are attributes of a pair (a,l). Parameter α means „*fitness*" and decides on belonging to the set, φ is an „*age*" – element is removed from the set after exceeding some number of iterations (here: 3 iterations).

In every iteration of the algorithm, after determining the local minima (procedure *LocalOpt*), a new set $FS^{i+1} = FS^i$ is established. Next, a *FixSet*(*LMi, FS*^{i}) procedure is called, in which there are executed the following operations :

a. changing of the age of each element,
b. deleting the oldest elements,
c. inserting the new elements.

Inserting elements

Let $P^i = \{\pi_1, \pi_2, ..., \pi_\eta\}$ be a population of η elements in iteration *i*. For each permutation π_j from P^i, applying the local search algorithm *(LocalOpt*(π_j*)* procedure), a set of local minima $LM^i = \{\hat{\pi}_1, \hat{\pi}_2, ..., \hat{\pi}_\eta\}$ is determined. Each permutation

$$\hat{\pi}_j = (\hat{\pi}_j(1), \hat{\pi}_j(2), ..., \hat{\pi}_j(n)), \ j = 1,2,... \eta .$$

Let

$$nr(a,l) \ge \left|\{\hat{\pi}_j \in LM^i : \ \hat{\pi}_j(l) = a\}\right| .$$

It is a number of permutations from the set *LM*^{i}, in which there is an element a in the position l. If $a \in N$ is a free element and

$$\alpha = \frac{nr(a,l)}{\eta} \ge \Phi(i)$$

then the element a is fixed in the position l.

A new population (**procedure** *NewPopul*)

To generate a new population P^{i+1}, randomly drawn free elements are inserted in remaining free positions of the elements of population P^i.

NewPopulation(FL^i)
$P^{i+1} \leftarrow \emptyset$;
Determine a set of free elements:
$FE = \{a \in N : \exists (a,l,\alpha,\varphi) \in FS^{i+1}\}$
and a set of free positions
$FP = \{l : \exists (a,l,\alpha,\varphi) \in FS^{i+1}\}$;
for j:=1 **to** η **do** {*Inserting of fixed elements*}
for each $(a,l,\alpha,\varphi) \in FS^{i+1}$ **do**
$\pi_j(l) := a$;
$W \leftarrow FE$;
for s:=1 **to** n **do** {Inserting of free elements}
if $s \notin FP$ **then**
$\pi_j(s) = w$, **where** $w = random(W)$ a**nd** $W \leftarrow W \setminus \{w\}$;
$P_{i+1} \leftarrow P_{i+1} \cup \{\pi_j\}$.

Function *random* generates from a uniform distribution an element of the set *W*. Computational complexity of the algorithm is $O(\eta\, n)$.

5. Case study

The subject matter of the analysis is a network model of an investment and construction project (ICP) according to [9,13], containing of n = 16 of linked building processes. There has been a computational test carried out in order to prove the possibility of time buffers influence on regularity of workers' employment in an enterprise schedule. With regard to taken constraints the values of a objective function F were adopted as follows:

$$f(x) = \sum_{j=1}^{T} \left(\left| q_j(x) - \frac{1}{T} \sum_{i=1}^{n} d_i r_i \right| \right) \tag{1}$$

where:
$x \in R^n$,
$x=(x_1, x_2, \ldots, x_n)$ – vector of moments of initiating tasks' execution,
$x_i \in [a_i, b_i]$,
a_i – earliest moment of task beginning,
b_i – latest moment initiating tasks' execution,
$q_j(x)$ – number of workers employed on a j day, j = 1,2,…,T,
T – time horizon,
d_i – time of process duration,
r_i – number of workers employed in order to carry out the process.

Below there is presented a network model of an exemplary enterprise taking into consideration time buffers; feeding FB [12] and project buffer PB. There is information concerning a number of activities, time of activities according to Goldratt method [4] and forecasted resources (number of employees).

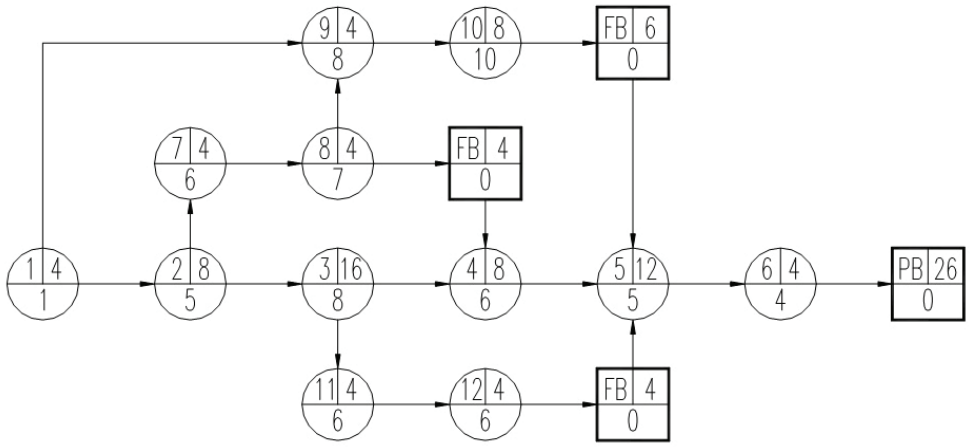


Fig. 4. The graph of ICP

Process number	Duration time	Earliest term of initializing	Earliest term of finishing	Latest term of begginning	Latest term of finishing	Number of workers
1	4	0	4	0	4	2
2	8	4	12	4	12	5
3	16	12	28	12	28	8
4	8	28	36	28	36	6
5	12	36	48	36	48	5
6	4	48	52	48	52	4
PB	26	52	78	52	78	0
7	4	12	16	16	20	6
8	4	16	20	20	24	7
FB	4	20	24	24	28	0
9	4	12	16	18	22	8
10	8	16	24	22	30	10
FB	6	24	30	30	36	0
11	4	28	32	40	44	6
12	4	32	36	44	48	6
FB	4	36	40	48	52	0

Table 1. The data of activities

Table 1 includes basic information concerning a construction project modeled by network. Basic time parameters forecasted to carry out construction works were counted. Numerical data necessary in order to carry out optimization calculations are presented below:

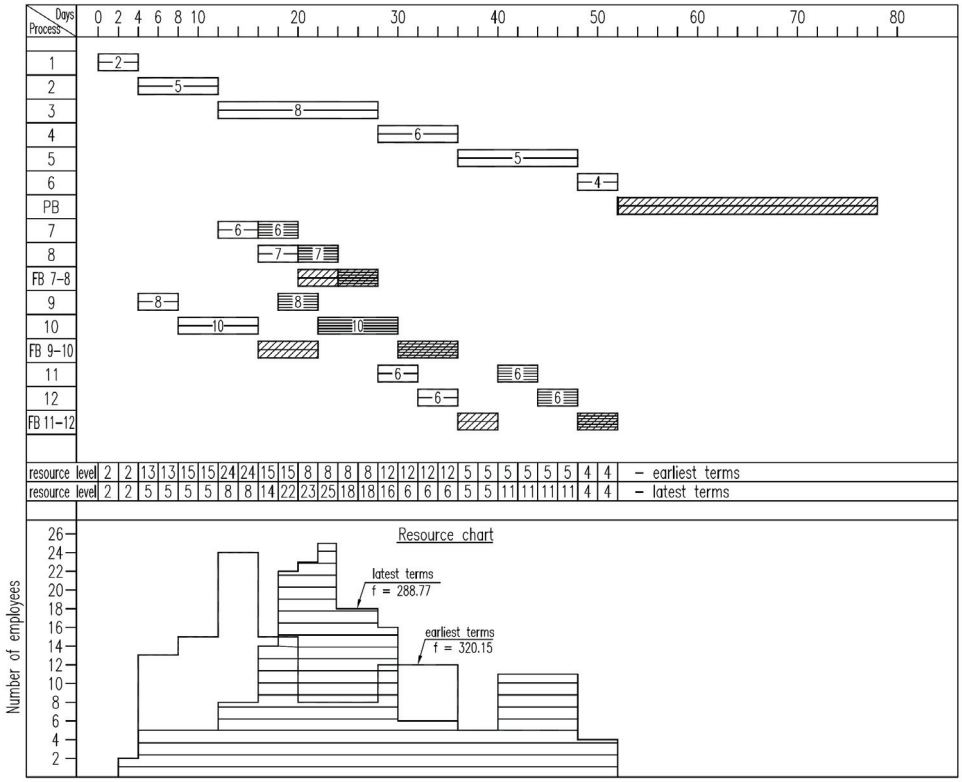


Fig. 5. The Gantt Chart, value of objective function for earliest terms f=320.15 for latest terms f=288.77.

Figure 5 depicts The Gantt Chart together with a graph of workers' employment for earliest and latest terms of works' beginning. For these extreme terms the values of goal function were calculated, i.e. f=320.15 and f=288.77 adequately.

Figure 6 shows Gantt's linear graph presenting an optimal schedule of a construction enterprise, for a taken goal function. Placement of a critical path does not change, whereas non-critical activities were put on a time scale taking into account time buffers (FB), in an optimal way with regard to a minimal value of a goal function.

Below Gantt's linear graph there is a graph of workers' employment corresponding to planned tasks. It shows the best solution concerning a minimal average divergence from an

average level of workers' employment. A striped line shows time buffers which are in fact a time reserve enabling its utilization for an optimal placement of tasks on a time axis of construction works, so that the possible level of workers' employment can be kept regular.

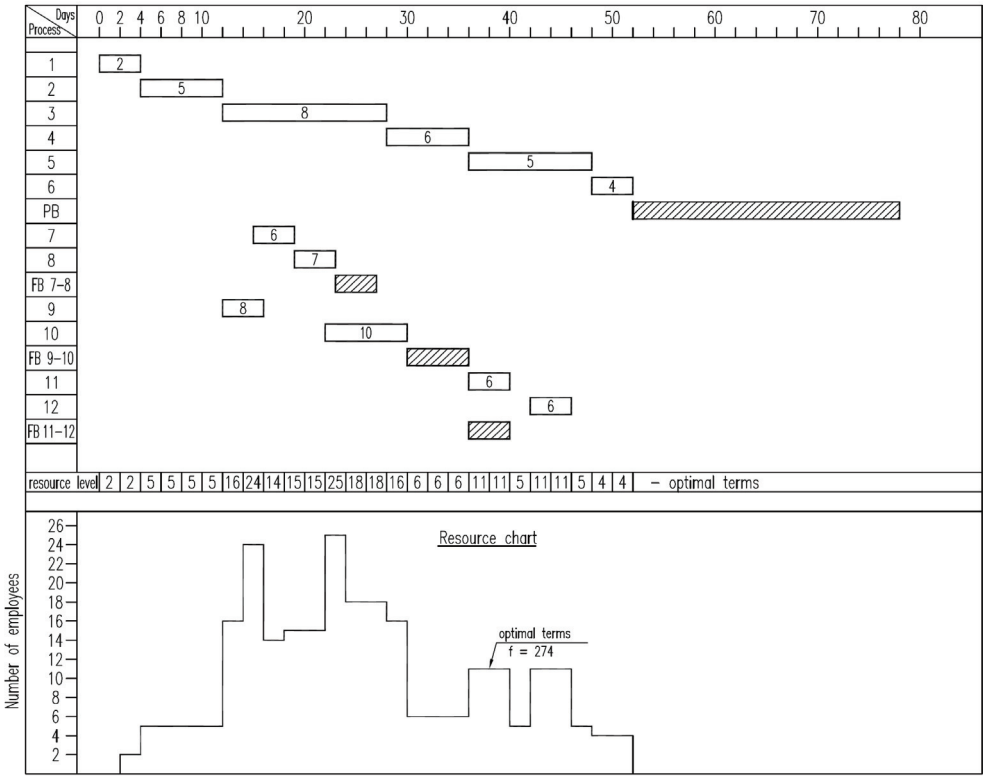


Fig. 6. The Gantt Chart after optimization, minimal value of objective function f=274.00.

In an analyzed example there has been a genetic algorithm (GA) applied ensuring after 1000000th iteration the result of f= 274.00, whereas after application of hybrid evolutionary algorithm (HEA) the result of f=274.00 just after 100 iterations. The calculations were carried out on a Pentium IV computer with a clock of 3 GHz. The time of calculations in the first case was 2 seconds whereas in the second – below 1 millisecond, i. e. over 2000 times faster.

6. Summary

The calculations have been based on an elaborated optimization programs with application of a genetic algorithm GA a hybrid evolutionary algorithm. For a presented optimization task of n = 16 size, achieved values of a goal function equal: in case of application of a

genetic algorithm (GA), f_{mi}=274.00 whereas for a hybrid evolutionary algorithm (HEA) also f_{min}=274.00. While analyzing the results in case a genetic algorithm (GA) from an example [13], hybrid evolutionary algorithm (HEA) and after introduction of time buffers according to CCS/BM methodology one can state that application of time buffers of a zero load of resources (team workers) increases the extend of a optimization task and ensures the smallest value of a total divergence from an average level of employment.

7. References

[1] Bożejko W., Wodecki M., A hybrid evolutionary algorithm for some discrete optimization problems, IEEE Computer Society, 2005, 326-331.

[2] Fox R., Goldratt E.M., The Race. [Croton-on-NY]: North River Press, 1986.

[3] Goldratt E.M., The Goal. [Great Barrington, MA]: North River Press, 1 st ed. 1984, 2nd revised ed, 1992.

[4] Goldratt E.M.,. Critical chain. [Great Barrington, MA]: North River Press, 1997.

[5] Goldratt E.M., The Haystack Syndrome: sifting information out of the data ocean, New York, North River Press, 1990.

[6] Goldratt E.M.. It is no luck. [Great Barrington, MA]: North River Press, 1994.

[7] Herroelen W., Leus R., On the merits and pitfalls of critical chain scheduling. Journal of Operations Management, 19, 2001, 559-577.

[8] Giffler,B., Thompson, G.L., Algorithms for solving production-scheduling problems. Operations Research, 8, 1960, 487-503.

[9] Leu S.S., Yang C.H., Huang J.C., Resource leveling in construction by genetic algorithm-based optimization and its decision support system application. Automation in Construction, 10, 2000, 27-41.

[10] Radovilsky Z.D., A quantitative approach to estimate the size of the time buffer in the theory of constraints. International Journal of Production Economics 55, 1998, 113-119.

[11] Rand G.K., Critical Chain: theory of constraints applied to project management. International Journal of Project Management, 18, 200, 173-177.

[12] Rogalska M., Hejducki Z., Shortening the realisation time of building project with application of theory of constraints and critical chain scheduling.Journal of Civil Engineering and Management, vol. X, Suppl 2, 2004, 99-105.

[13] Rogalska M., Bożejko W., Hejducki Z., Employment levelcontrol using genetic algorithms (in Polish). 51st KILiW PAN and KN PZITB Scientific Conference Gdańsk-Krynica, 2005, 185-192.

[14] Scavino N.J., Effect of Multiple Calendars on Total Float and Critical Path. Cost Engineering, vol.45, No.6, 2003.

[15] Steyn.H., Project management applications of the theory of constrains beyond critical chain scheduling. International Journal of Project Management, 20, 2002, 75-80.

[16] Turner JR., Controlling progress with planned cost or budgeted cost. International Journal of Project Management , 18(3), 2000, 153-4.

[17] Wiest, J.D., Some properties of schedules for large projects with limited resources. Operations Research, 12, 1964, 395-418.

[18] Yang J.B., Applying the theory of constraints to construction scheduling. Proceedings of Second International Structural Engineering and Construction Conference (ISEC 02), Vol. 1, 2003, 175-180.

16

A Memetic Algorithm for the Car Renter Salesman Problem

Marco Goldbarg[1], Paulo Asconavieta[2] and Elizabeth Goldbarg[1]
[1]*Universidade Federal do Rio Grande do Norte*
[2]*Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense*
Brazil

1. Introduction

The Traveling Salesman Problem (TSP) is a classic Combinatorial Optimization problem. Given a graph $G=(N,M)$, where $N=\{1,...,n\}$ is the set of nodes and $M=\{1,...,m\}$ is the set of edges, and costs, c_{ij}, associated with each edge connecting vertices i and j, the problem consists in finding the minimum length Hamiltonian cycle. The TSP is NP-hard (Garey & Johnson, 1979) and one of the combinatorial optimization problems more intensively investigated. The size of the larger non trivial TSP instance solved by an exact method evolved from 318 cities in the 80's (Crowder & Padberg, 1980), to 7397 cities in the 90's (Applegate *et al*., 1994) and 24978 cities in 2004. The best mark was reached in 2006 with the solution of an instance with 85900 cities (Applegate *et al*., 2006). The TSP has several important practical applications and a number of variants (Gutin & Punnen, 2002). Some of these variants are classic such as the Peripatetic Salesman (Krarup, 1975) and the M-tour TSP (Russel, 1977), other variants are more recent such as the Colorful TSP (Xiong *et al*., 2007) and the Robust TSP (Montemanni *et al*., 2007), among others.

A new TSP variant is introduced in this chapter named The Car Renter Salesman Problem (CaRS). It models important applications in tourism and transportation areas and represents a complex variant that challenges the state of the art. In this paper the new problem and some variations are presented, its complexity is analyzed and some related problems are briefly overviewed. A memetic algorithm is proposed for the problem and it is compared to a hybrid GRASP/VND algorithm.

CaRS Problem is introduced in Section 2, where several conditions under which this variant can be presented are introduced. Section 3 presents two metaheuristic methods for the investigated problem. In order to compare the performance of the proposed approaches, a set of instances introduced for the new problem, named CaRSLib. This set contains Euclidean and non-Euclidean symmetric instances with number of cities ranging from 14 to 300 and number of cars between 2 and 5. A set of 40 instances is used in the computational experiments. The heuristics proposed in Section 3 establish the first upper limits for solutions of CaRSLib of instances. The results of computational experiments comparing the performance of the proposed approaches are presented in Section 4. Statistical tests are applied to support conclusions on the behavior of the proposed algorithms. According to

those results, the Memetic Algorithm is pointed as the best approach for CaRS considering the tested cases. Final conclusions and future directions for the research of algorithms for CaRS and its variations are addressed in Section 5.

2. The car renter salesman problem

2.1 The rental car industry

Today over 90 significant economic size car rental companies exist in the world market (Car, 2008). The importance of the car rental business can be measured both by the enterprise turnover as by the size of the companies that provide the service. For example, Hertz is a company in the car rental segment with wide accessibility of providing the services at approximately 8,000 locations in approximately 145 countries (Hertz, 2009). The Enterprise has more than 878,000 vehicles in its rental and leasing fleet and operates across 6,900 local markets (Enterprise, 2009). Avis operates in more than 3,800 locations all over Europe, Africa, the Middle East and Asia. In December 2007, the company operated an average fleet of 118,000 vehicles (Avis, 2009). Avis Budget Group Inc. earned $ 5.1 billion dollars in 2009 (Avis, 2010). In 2009, the Enterprise Holdings Inc. which owns today the National Car Rental, Alamo Rent A Car and WeCar earned about 12.1 billion dollars (Conrad & Perlut 2006; Wikipedia, 2010). These numbers represent only part of the market that also has other major car rental networks such as Dollar and Hertz. The world market in 2012 is estimated at 52.6 billion dollars (Car Rental, 2008).

Besides being itself a major business, spending on car rentals may represent a significant portion of the activities involving tourism and business. Currently the rental options are becoming increasingly diversified with the expansion of the companies, justifying the search for rent schemes that minimize the total cost of this form of transport.

2.2 Models of combinatorial optimization in the car rental industry

Among the various logistical problems of this branch of activity, the literature describes specific studies of combinatorial optimization in the Fleet Assignment Problem (Lia &Tao, 2010), the Strategic Fleet Planning and Tactical Fleet Planning (Pachon *et al*., 2003), the Demand Forecast (Edelstein & Melnyk, 1977) and the car fleet management problem with maintenance constraints (Hertz *et al*. 2009). Logistic problems that occur in the car rental industry are reviewed by Yang *et al*. (2008). These studies focus on the viewpoint of the car rental industry, however, the customer's point of view has not yet been the subject of published research.

2.3 The car renter salesman problem

In general, under the viewpoint of a user of rented cars, the goal is to minimize the costs to move from a starting point to a destination. On the other hand, when someone rents a car, it is assumed that it meets the requirements of comfort and safety. During the travel, in addition to the costs of renting the car, at least the costs of fuel and the payment of fees to travel on the road should be considered. Let $G=(N, M, W)$ be a graph where $N=\{1,...,n\}$ represents the set of vertices, $M=\{1,...,m\}$ is the set of edges and $W=\{1,...,w\}$ is the set of distances between the vertices or the length of the edges of the set M. The problem described in this paper has the following features:

1. Several types of cars are available for rent, each of them has own characteristics, that is, specific operational costs. These costs include fuel consumption, fees that have to be paid to travel on roads and the value of the rent. The fees that have to be paid to travel on the roads may depend on the type of the car and on the specific roads chosen for the route. The value of the rent can also be associated with a cost per kilometer. Thus, without loss of generality, these costs can be considered as a function of each car on a value associated to the edges (i,j) of graph G. The operational cost of a given car k to traverse an edge (i,j) is denoted by c_{ij}^{k}.
2. A car rented in a given company can only be returned in a city where there is an agency of the same company. It is therefore not allowed to rent a car of a given company to travel on a certain segment of the route, if that car cannot be returned on the last city of the segment – there is not an agency of this company in the last city of the segment.
3. Whenever it is possible to rent a car in a city i and return it in city j, $i \neq j$, there is an extra for returning the car to its home city. The variable d_{ij}^{k} represents the expense to return car k to city j when it was rented in city i, $i \neq j$.
4. The tour begins and ends in the city where the first car is rented, the city that is the basis for the CaRS.
5. The return cost is null in case the tour is completed with a single car which is delivered in the same town it was rented. This case corresponds to the classic TSP considering the cost of other conditions associated only with the selected car.
6. Cars with the same characteristics rented in a single rental car company can be hired under different costs, depending on the city they where rented or on the contract negotiation. Therefore, without loss of generality, the designation of rent can be efficiently controlled by decisions related to cars, not considering the companies. The set $K=\{1,...,k\}$, $|K|=k$ is the set of different cars that can be in the solution.
7. The costs of returning the rented car may be strictly associated with the path between the city where the car is delivered and the city where the car was rented or these costs can be a result of independent calculation.

The objective of the proposed problem it to find the hamiltonian cycle that, starting on an initial vertex previously known, minimize the sum of total operating costs of cars in the tour. The total operating costs are composed of a parcel that unifies the rent and other expenses in a value associated to the edges, and a parcel associated to return the car to a city that is not its basis, calculated for each car and for each pair of cities origin/return in the cycle. The CaRS cycle may also be understood as obtained by the union of up to t Hamiltonian paths developed on up to t disjoint subsets of vertices of G. Each of the paths is accomplished with a different car or a car different from those used for the neighboring paths in the cycle. Therefore the cities that compose the cycle can be grouped into up to t different subsets of vertices of G that are covered by cars at least distinct from each other in the neighboring paths in the cycle.

Figure 1 illustrates, in a complete graph with six vertices, a typical instance of CaRS. In the example there are three different rental cars. Figures 1(a), (b) and (c) show the accounting of the costs involved in the displacement of each type of car. Note that, unlike the classical traveling salesman cycle, the solution of CaRS depends on the city chosen to be the starting point of the tour, the basis of the salesman. This fact is due to the rate of return is linked both to the starting city and the direction of devolution. In the example this city is represented by vertex F.

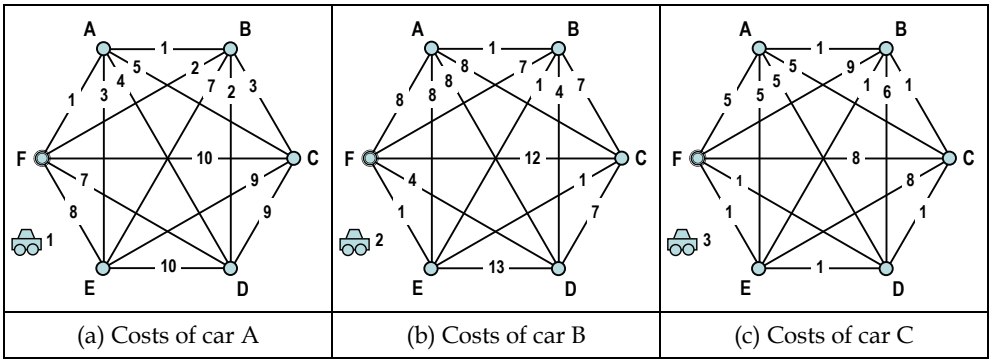


Fig. 1. Costs associated to each car

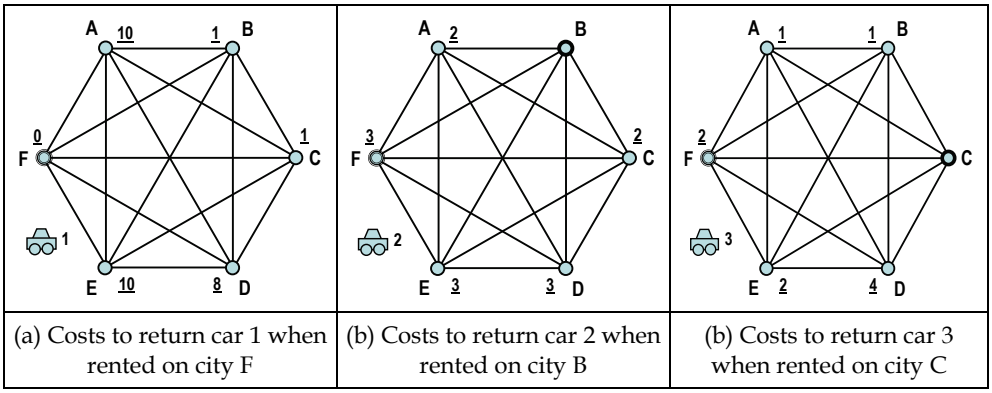
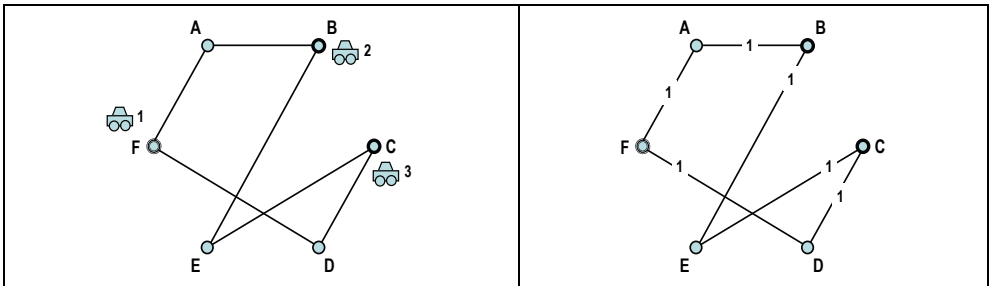


Fig. 2. Return costs

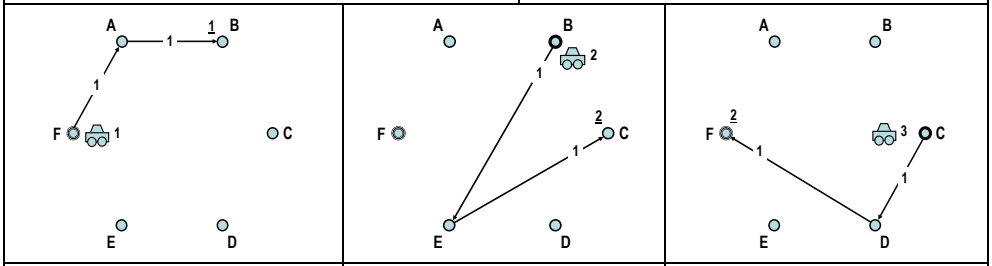
Figure 2 shows, for the example in Figure 1, some of the costs of returning the cars to their bases. Delivery costs appear as underlined numbers next to vertices. Figure 2(a) shows the graph of return of car 1 when rented on vertex F. Figure 2(b) shows the graph of return of car 2 when rented on vertex B and Figure 2(c) the return of car 3 when rented on vertex C. In the general case return costs are known of all cars when rented in any of the cities.

A solution of the problem exemplified in Figures 1 and 2 is exhibited in Figure 3. This solution considers a case where all available cars are rented and no car is rented more than once. The cost of the cycle, according to the solution shown in Figure 3, corresponds to the cost of path F-A-B for car 1, added to the cost of path B-E-C for car 2, added to the cost of path C-D-F of car 3, in a total of 6 unities. To this value it is necessary to add the cost of returning the car to their bases. For car 1, the cost of the return from B to F is one unity. For car 2, the cost of the return from C to B is two unities and, for car 3, the cost to return to C when the car is delivered in F is two unities. Thus, the cost of the final solution is 11 unities.

The CaRS Problem has several variants in accordance with the real conditions of the problem. The problem can be classified according to the availability of cars, the alternatives of return, the existence of symmetry of the cost matrix and the existent links between the cities, etc.



(a) Cars distributed in the route | (b) Route costs



(a) Costs of the route of car 1 | (b) Costs of the route of car 2 | (b) Costs of the route of car 3

Fig. 3. Costs of the route of the exemplified problem

2.4 Variants of the car renter salesman problem

The Renter Salesman admits several specific situations being classified according to:

1. Availability of rental cars

Since there is no guarantee that the rental companies are present in all cities of the tour, it may not necessarily be assumed that any car can be rented in any city. The case in which all cars can be rented in all cities is called *total*. In any other case the problem is called *partial*. In this study, when no observation is made otherwise, the problem is considered total.

2. Alternatives to return the car to its basis

Since there is no guarantee that the rental companies operate services for receiving cars in all cities, it may not necessarily be assumed that any rental car can be returned in any city. To distinguish these different situations, the case in which all cities can receive all cars is called *unrestricted*. In any other situation the problem is called *restricted*. In this paper, when no observation is made otherwise, the problem is considered unrestricted.

3. The integrity of the contract

When the problem does not allow the same type of car is rented more than once on the tour, the problem is called *without repetition*, in this case $k \geq t$. The case without repetition where all cars have to be rented is called *exact*, in this case $k=t$. In any other situation the problem is called *with repetition*. In this paper, when no observation is made otherwise, the problem is considered without repetition.

4. Calculation of the costs of returning a rental car

The costs of returning the cars may be made of values independent of the topology or network restrictions. In this case the problem is called *free*. In the event that the cost of returning a car is calculated taking into account the route used by the car to return to its

base, the problem is said to be *bonded*. In this paper, when no observation is made otherwise, the problem is considered free.

5. Symmetry of the distances between the cities

When $c_{ij}^k = c_{ji}^k$ for $1 \le i,j \le n$, $1 \le k \le ncars$, where *ncars* denotes the number of cars available in a given problem instance, the problem is said to be *symmetric*, otherwise the problem is said to be *asymmetric*.

6. Existence of links in the connection graph that models the problem

When the graph that models the problem is complete, the problem is called *complete*, otherwise the problem is called *incomplete*.

2.5 The difficulty of solving CaRS

The problem basically consists in determining a Hamiltonian cycle in a graph G by composition of paths developed on the vertices of G. Let $T=\{1,...,t\}$ denote the set of indices of up to t subgraphs H_r of G, $r \in T$. Calling $V(H_r)$ the vertices of H_r, the subgraphs H_r of CaRS have the following properties.

$$\bigcup_{r=2}^{t} V(H_r) = N \tag{1}$$

$$\left| V(H_s) \cap V(H_r) \le 1 \right| \quad \forall r,s \tag{2}$$

Constraint (1) determines that the union of all paths visits all vertices of G. Constraints (2) implies that two different subgraphs never have more than one vertex in common, a condition to prevent formation of subcycles. Note that the constraints (1) and (2) are not sufficient to guarantee the cycle of the CaRS. It is also necessary that the t subgraphs considered three to three, four to four, and so on, until t-1 to t-1, do not have more than one vertex in common.

Once this problem deals with Hamiltonian paths, each path done by a car in one subgraph H_r visits all vertices of H_r. The path of subgraph H_r has to be assigned to a car different from the cars assigned to neighbor paths during the construction of an Hamiltonian cycle in G. The costs of the edges of each subgraph correspond to the operation costs of the car traversing H_r. Furthermore, when $t \ge 2$ the total cost considers the return cost of each car rented in city i and returned in city j, $i \ne j$. Hamiltonian cycle and Hamiltonian path problems are well known NP-complete problems (Garey & Johnson, 1979). Due to what was previously exposed, the difficulty of solving CaRS is at least the same as the TSP. Nevertheless, although some solutions of the TSP are also solutions of CaRS, the latter has a number of feasible solutions greater than the former and incorporates all the requirements of the TSP, like other several classes of vehicle routing problems which are known to be more difficult than the TSP (Ralphs *et al.*, 2003).

The Traveling Salesman Problem is a particular case of CaRS in the situation where there is only one vehicle available for rent. Note that the solution space of CaRS is exponentially greater than the solution space of the Traveling Salesman Problem. Considering $G=(N,M)$ a complete graph and that CaRS is total, unrestricted and without repetition, any permutation of the vertices of G is a feasible solution for the rental car problem considering only one of the k possible cars. Once there are k cars available for rent, there are $k.n!$ different feasible configurations that meet the condition of use of only one different car in CaRS. From $k \ge 2$

each hamiltonian cycle of the car renter salesman can be partitioned in up to k-subpaths in sequence and to each possibility of composition of such partitions a distinct cost for the route can be assigned. The possibility of this single value is guaranteed due to the composition of the costs related to the return fee of the rented cars with the costs of the trajectories of independent costs of each car. The number of possible partitions associated to each hamiltonian cycle in G is equal to the number of different ways the set of n cities of the cycle can be divided in s groups of cities that are visited by s different cars, $k \geq s \geq 2$. The number that counts this process of division of a set in disjoint sets that go from 2 to k is the Stirling number of the second type. In this way, the number of configurations of the space of solutions of CaRS is at least $O(2^n)$ greater than the space of solutions of the Traveling Salesman Problem, since that for $k = 2$ the associated number of Stirling is $O(2^n)$ (Amdeberhan *et al*. 2008). For a general case of CaRS the dimension of the space of solutions is still greater once there is not a theoretical limit for the number of different cars to be considered in the problem.

3. Metaheuristic algorithms

This section presents two heuristics for the investigated problem. The first one is a Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1995) combined with Variable Neighborhood Descent (VND) (Mladenović & Hansen, 1997) in the local search phase. The second heuristics is a Memetic Algorithm (Moscato, 1989).

3.1 GRASP with VND

This algorithm has a pre-processing phase where *nCar* optimal TSP solutions are obtained with the Concorde TSP Solver (Applegate *et al.*, 2001), one for each available car, where *nCar* is the number of cars available for the instance being considered. The constructive phase of the GRASP hybridized with VND, named GVND, starts with a random selected car at the home city. A path is built each iteration between two cities: a known origin and a destination city randomly chosen among the cities yet not considered by the algorithm. A Restricted Candidate List (RCL), with size α, is built with the cities that have the cheapest return rates for the car being considered in the current iteration. The destination city is selected at random, based on a roulette wheel, from the RCL and a path between the origin and the destination city is built. Except for the last iteration, the path is built based on the tours built in the pre-processing phase. In the first iteration the path between the origin and destination cities is obtained in the optimal solution correspondent to the first selected car. The path of the *i-th* car, $1 < i < nCar$, is also obtained from the optimal solution that corresponds to the *i-th* car, but in this case a procedure to remove cities already considered in paths constructed in previous iterations may be necessary. Suppose that city b, between cities a and c in the path built in the *i-th* iteration, is already in a path built in iteration j, $j < i$. Then the procedure removes city b from the *i-th* path and includes a link between cities a and c. The initial starting city is the origin of the first iteration. The origin city of iteration i, $i > 1$, is the destination city of iteration i-1. The destination city of the last iteration is the initial starting city. In the last iteration, the nearest neighbor heuristic is used to build the path of the last considered car.

In the local search phase a VND metaheuristic was used to explore the search space of three neighborhood structures named *InvertSol*, *Insert&Saving* and *Shift*. *InvertSol* is a simple low

time consuming heuristics that reverses the sequences of cities of an input solution. With the reversal, though the same cars traverse the same sets of cities, the cost of rent and fees for returning the car to where they were rented change. The *Insert&Saving* procedure searches for a car insertion in a given solution that yields a decreasing of its cost. Let s be a solution in which there is at least one car that is not assigned to a path in s. *Insert&Saving* method randomly chooses a not assigned car and searches the best position to insert it in s. The procedure verifies the cost of the insertion of the new car in every point of s. If any of these insertions produces a solution with a cost lower than the cost of s, the new solution is set as the current solution in the local search. The procedure continues until all non-assigned cars have been considered for insertion. In the third procedure, *Shift*, a neighboring solution s' of a solution s is generated by exchanging two cities within the path of one car of s. The whole neighborhood of each solution is searched in the local search procedures.

3.2 Memetic algorithm

Algorithm 1 shows the pseudo-code of the Memetic Algorithm (MA) developed for CaRS. The input parameters are: number of generations (*nOffspring*), population size (*sizePop*), recombination rate (*txCros*) (the number of individuals that reproduce in each generation), mutation rate (*txMuta*) and renewal rate of the population (*txRenw*).

Algorithm 1 – Main Procedure of Memetic for CaRS

```
1.  main(nameInstance,sizePop,nOffspring,txCros,txMuta,txRenw)
2.    instanceRead(nameInstance)
3.    Pop[] ← generateInitPop(sizePop)
4.    VNDlocalSearchPhase(Pop)
5.    for i of 1 to nOffspring do
6.      for j of 1 to sizePop*txCros do
7.        dad,mom ←  parentsSelection()
8.        son1, son2 ← Crossover(dad,mom)
9.        son1, son2 ←  carsMutation(son1,son2,txMuta)
10.       VNDlocalSearchPhase(son1,son2)
11.       if son1,son2 < Pop[dad],Pop[Mom]
12.         Pop[dad] ← son1, Pop[mom] ← son2
13.   generateNewIndividuals(sizePop*txRenw)
14. return(Pop[0])
```

Chromosomes are represented in 2-dimensional arrays with n elements as illustrated in figure 4 for an instance with n = 11 and 5 cars. The second row corresponds to the sequence of cities visited in the tour. The elements in the first row correspond to cars. Let car c be assigned to the cities in the second row corresponding to indices i_1 to i_m, that is, car c goes from city i_1 to city i_m, then the elements of the first row corresponding to indices i_1 to i_{m-1} are equal to c. The last city visited by a car is not assigned to that car in the chromosome, since the car is returned on that city and another car is rented there to continue the tour. The starting city (city 0) is not represented in the chromosome as the final destination. In figure 4 four of the five available cars are used. The tour begins at city 0 with car 2 which passes through cities 6, 4, 3, 10 and is delivered in city 7 where car 1 is rented. Car 1 proceeds to

city 9 passing through city 1. Car 5 is rented in city 9, passes through cities 2 and 5 and is delivered in city 8 where the last car, 4, is rented. Car 4 is delivered in the starting city. The fitness of each chromosome is given by the inverse of the objective function, which means that the lower the value of the objective function the fittest the chromosome is.

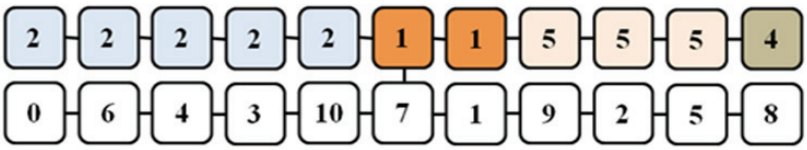


Fig. 4. Chromosome

The initial population is generated with a version of the nearest neighbor heuristics adapted for CaRS in procedure *generateInitPop*( ) which receives the size of the population as input parameter. Let *nCar* be the number of available cars of a given instance. The algorithm randomly selects a car c and a destination city j for c, $j \neq 0$. Then a path between cities 0 and j is built with the nearest neighbor heuristic. City j is set as the new origin and a new car and a new destination city are randomly selected. The procedure continues until all cities are added to the tour or until there is only one car available. In the latter case, the last available car is assigned to a path built with the same heuristics between the previous destination city and the starting city, closing the tour. In step 4, each individual of the initial population is subjected to the same VND procedure used in GVND.

Parents for recombination are selected with the roulette wheel method. A multi-point recombination operation adapted for CaRS is used to generate two children. The recombination operator is illustrated in Figure 5, considering an instance with n = 11 and 3 cars. Two parent chromosomes, A and B, generate offspring C and D. In Figure 5 a 2-point operator is used. The first and third parts of chromosomes A and B are inherited by chromosomes C and D, respectively. A restoration procedure may be necessary to restore feasibility regarding the routes and car assignments. For example, after recombination the route of chromosome C is [0 3 1 8 10 1 9 4 5 10 6] which is not feasible since cities 1 and 10 appear twice each and cities 2 and 7 are missing. Thus the route of chromosome C is replaced by [0 3 1 8 10 * 9 4 5 * 6] with asterisks replacing the second time cities 1 and 10 appears. Each asterisk is then replaced at random by cities 2 and 7. The row corresponding to the car assignment of chromosome C after recombination is [1 1 1 1 2 3 3 2 2 2 3] which is not feasible for the problem considered in this paper, since each car can be rented only once. Thus, the car assignment of chromosome C is replaced by [1 1 1 1 2 3 3 * * * * 3] and each asterisk is replaced by car 3. Chromosome D is analyzed similarly.

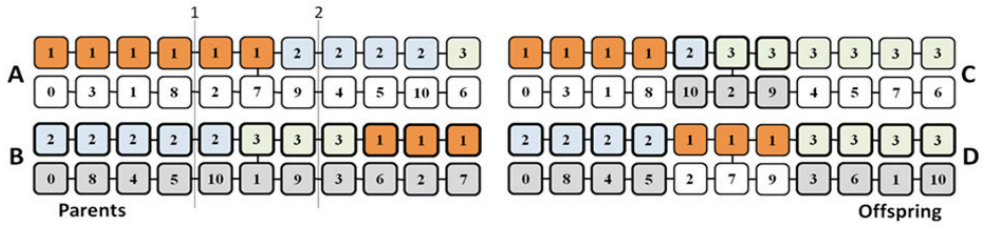


Fig. 5. Recombination operator

The solutions resulting from the recombination are subjected to mutation. The mutation operator verifies which vehicles are not in the solution represented in the chromosome. Each of these vehicles is inserted in the chromosome in a given segment defined in advance. The maximum size of each segment is defined as the mutation rate. The mutation operator is illustrated in figure 6 considering an instance with n= 11, 5 cars, and the mutation rate sets the size of the segments to 3. Cars 3 and 4 are not used in the solution shown in chromosome A of figure 6. The mutation operator inserts the missing cars in the solution, resulting on chromosome B. Cities 10 and 5 are chosen at random to be the starting cities of cars 3 and 4, respectively. Therefore, vehicle 5 is replaced by vehicle 3 in cities 10, 7 and 1 and car 4 replaces car 1 in cities 5 and 8, since these are the last two cities of the tour. The sequence of cars is the resulting chromosome is [2 2 2 5 3 3 3 5 1 4 4]. A repairing function has to be utilized to restore the feasibility of the resultant chromosome, since car 5 appears in two different paths. The second time car 5 appears in the solution is removed, resulting on [2 2 2 5 3 3 3 * 1 4 4]. The asterisks are replaced by the car that appears in the city immediately before to the ones which the asterisks are assigned. In this example, the asterisks are replaced by car 3.

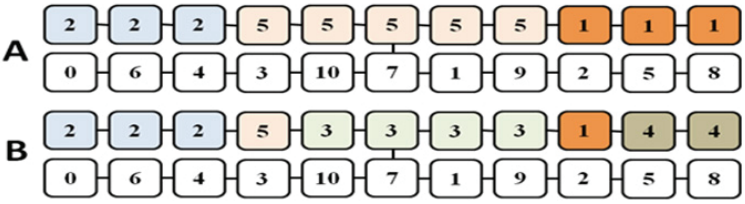


Fig. 6. Mutation operator

After crossover and mutation, the offspring is subjected to the VND method presented at GVND. The resulting solutions are compared with their parents and the best two individuals survive. Finally, part of the current population is replaced by new solutions generated with the constructive method used to create the initial population. The number of new individuals created with that procedure is given by the renewal rate and the individuals chosen to be replaced are those with the worst values of fitness. This renewal process promotes diversification and prevents premature convergence.

4. Computational experiments

This section presents the comparison between the performance of the GRASP/VND and the Memetic Algorithm, called GVND and MA, respectively. Since the problem introduced in this paper is new, a library of instances, named CaRSLib, was created with the purpose of testing the proposed algorithms. These instances have the following features: *total* (all cars can be rented in all cities), *unrestricted* (all cars can be delivered in any city), *without repetition* (each type of car can be rented at most once), *free* (the return costs are not correlated with instance topology), *symmetric* (costs to go from city i to city j and vice-versa are equal) and *complete* (the graph that models the instance is complete). The set consists of Euclidean and non-Euclidean instances. For each set, three groups of instances were created, the first is based on real maps, the second was formed with randomly generated data and the third one is based on the TSPLIB instances. The dataset, the description of each group of instances and file formats are available at http://www.dimap.ufrn.br/lae/en/projects/CaRS.php.

An exact backtracking algorithm was developed and adapted to CaRS. This method enumerates all possible configurations utilizing permutations of the available cars for each instance. The results were used to evaluate the solutions generated by the metaheuristic algorithms on the instances solved by the exact method. The algorithm was implemented in C++ and executed on an Intel Core Duo 1.67 GHz, 2GB RAM running Linux. The algorithm solved eighteen Euclidean and non Euclidean instances with n between 10 and 16 and 2 cars. Table 1 presents the results for the backtracking algorithm (column *Backtrack*) and the best results obtained by GVND and MA. Columns *gap* show the deviation of the best solution (*Best*) found by the metaheuristic algorithm from the optimum (#*Opt*).
Table 1 shows that the performance of the memetic algorithm is satisfactory for small instances, obtaining the optimum for all investigated instances. The success rate of GVND is 78% and the maximal deviation from the best solution is 3%.

INSTANCE			BACKTRACK		GVND			MA		
Name	City	Car	T(s)	#Opt	T(s)	Best	GAP	T(s)	Best	GAP
Mauritania10e	10	2	1	540	1	540	0.00	1	540	0.00
Mauritania10n	10	2	1	571	1	571	0.00	1	571	0.00
Colombia11e	11	2	19	620	1	620	0.00	1	620	0.00
Colombia11n	11	2	14	639	1	640	0.00	1	639	0.00
Angola12e	12	2	266	719	1	719	0.00	1	719	0.00
Angola12n	12	2	144	656	1	656	0.00	1	656	0.00
Peru13e	13	2	1953	672	1	672	0.00	1	672	0.00
Peru13n	13	2	1847	693	1	693	0.00	1	693	0.00
Libia14e	14	2	31273	730	1	730	0.00	1	730	0.00
Libia14n	14	2	28331	760	1	764	0.01	1	760	0.00
BrazilRJe	14	2	44104	294	1	297	0.01	1	294	0.00
BrazilRJn	14	2	35263	167	1	170	0.02	1	167	0.00
Congo15e	15	2	455788	756	1	756	0.00	1	756	0.00
Congo15n	15	2	412212	886	1	886	0.00	1	886	0.00
Argentina16e	16	2	7603200	955	1	955	0.00	1	955	0.00
Argentina16n	16	2	7612310	894	1	894	0.00	1	894	0.00
BrasilRN16e	16	2	7609203	375	1	375	0.00	1	375	0.00
BrasilRN16n	16	2	7613217	188	1	194	0.03	1	188	0.00

Table 1. Results of Backtrack, GVND, MA

The results shown in the following tables for GVND and MA were obtained on a PC Intel Xeon QuadCore W3520 2.8 GHz, 8G of RAM running Scientific Linux 5.5 64bits. The results refer to 40 CaRS instances, 20 of them are Euclidean and 20 are non Euclidean. Each group of instances is formed with 10 instances based on real maps, 5 random instances and 5 instances based on TSPLIB (Reinelt, 1995). Thirty independent executions of each algorithm were performed for each instance. Two groups of experiments were performed with fixed processing times. In the first group the average processing times spent by GVND to find its

best solution for each instance was given to both algorithms. In the second group the average processing times of the MA for each instance was fixed for both algorithms.
Preliminary tests to tune the parameters of the proposed algorithms were executed on a set of 20 CaRSLib instances, with number of cities ranging from 14 to 300 and 2 to 5 vehicles. Twenty independent executions were performed for each instance. Two groups of tests were performed with the GVND. First, the algorithm was executed without the VND method and then the VND was included. The parameter α was set to 0.25. The maximum number of re-starts was set to 300. An additional stopping criterion fixed 90 re-starts without improvement of the best current solution. The parameters chosen for the Memetic Algorithms are: *nOffspring* = 20, *sizePop* = 30, *txCros* = 0.60, *txMuta* = 0.40 and *txRenw* = 0.15. The stopping criterion was *maxGen* = 0.30*nOffspring* generations without improvement of the best current solution.

INSTANCE				GVND				MA				T(s)	p-value
Name	City	Car	#Best	Avg	SD	Best	Freq	Avg	SD	Best	Freq		
BrasilRJ14e	14	2	294	297	0	297	0	**294**	0	**294**	29	1	0
BrasilRN16e	16	2	375	**375**	0	**375**	30	**375**	2	**375**	29	1	0.85
BrasilPR25e	25	3	510	**510**	0	**510**	29	515	5	**510**	16	2	1
BrasilAM26e	26	3	467	495	1	495	0	**485**	9	**469**	0	3	0
BrasilMG30e	30	4	563	603	2	595	0	**599**	7	**575**	0	5	0
BrasilSP32e	32	4	611	633	5	626	0	**621**	4	**611**	2	8	0
BrasilRS32e	32	4	510	537	9	529	0	**522**	6	**510**	1	8	0
BrasilCO40e	40	5	779	**807**	2	805	0	822	10	**779**	1	18	1
BrasilNO45e	45	5	886	1008	0	1008	0	**978**	34	**886**	1	23	0
BrasilNE50e	50	5	822	963	5	940	0	**954**	27	**822**	1	43	0.08
Betim100e	100	3	1401	1723	8	1708	0	**1692**	89	**1410**	0	78	0.99
Vitoria100e	100	5	1598	**1802**	75	1642	0	1891	87	**1598**	1	155	1
PortoVelho200e	200	3	2827	**3142**	29	3041	0	3149	129	**2827**	1	466	0.98
Cuiaba200e	200	3	3052	**3379**	88	3212	0	3414	80	**3217**	0	686	1
Belem300e	300	4	4031	4635	121	4563	0	**4425**	76	**4031**	1	1804	0
berlin52eA	52	3	8948	**9020**	35	8991	0	9081	72	**8948**	4	20	1
eil76eB	76	4	1940	2228	42	2158	0	**2077**	43	**1940**	1	87	0
rat99eB	99	5	3339	**3439**	42	3351	0	3513	75	**3365**	0	194	1
rd100eB	100	4	9951	**10107**	81	**9951**	1	10364	172	10054	0	103	1
st70eB	70	4	2037	2201	44	2085	0	**2151**	46	**2042**	0	77	0

Table 2. Results with time determined by GVND for Euclidean instances

With the aim of comparing the algorithms on a fair basis, the same maximum processing time is given for both algorithms. These processing times were obtained in preliminary experiments with the stop conditions afore mentioned. The results are reported in Tables 2-5. These tables show the name of the instance (*Name*), the number of cities (*City*), the number of available cars (*Car*), the best solution found (#*Best*), the average (*Avg*) solution, the best solution obtained by one of the tested algorithms (*Best*), the standard deviation (*SD*), the number of times (*Freq*) the best known solution, reported in column #*Best*, was found by

each algorithm, the maximum processing time in seconds (T) and the p-value obtained in the statistical U-test (Conover, 2001). Considering level of significance 0.05, values less than 0.05 in the last column indicate that the performance of MA is significantly better than the performance of GVND and values greater than 0.95 indicate that GVND produced better results than MA.

Tables 2 and 3 refer to Euclidean instances and show the results with the maximum processing time fixed by the stop conditions of GVND and MA, respectively. Similarly, Tables 4 and 5 show results for non Euclidean instances with the maximum processing time fixed by the stop conditions of GVND and MA, respectively.

INSTANCE				MA				GVND				T(s)	p-value
Name	City	Car	#Best	Avg	SD	Best	Freq	Avg	SD	Best	Freq		
BrasilRJ14e	14	2	294	**294**	1	**294**	25	297	0	297	0	1	0
BrasilRN16e	16	2	375	376	4	**375**	27	**375**	0	**375**	30	1	0.96
BrasilPR25e	25	3	510	515	5	**510**	17	**510**	0	**510**	30	2	1
BrasilAM26e	26	3	467	**481**	10	**467**	3	495	0	495	0	4	0
BrasilMG30e	30	4	563	**596**	10	**563**	1	602	2	595	0	6	0
BrasilSP32e	32	4	611	**624**	5	**615**	0	632	4	626	0	8	0
BrasilRS32e	32	4	510	**523**	7	**512**	0	536	9	529	0	8	0
BrasilCO40e	40	5	779	824	7	**801**	0	**806**	2	806	0	17	1
BrasilNO45e	45	5	886	**993**	27	**897**	0	1008	0	1008	0	25	0
BrasilNE50e	50	5	822	963	2	953	0	**962**	11	**908**	0	31	0.43
Betim100e	100	3	1401	**1642**	110	**1401**	1	1720	7	1708	0	128	0.30
Vitoria100e	100	5	1598	1922	29	1814	0	**1859**	77	**1676**	0	98	1
PortoVelho200e	200	3	2827	3134	117	**2871**	0	**3128**	22	3041	0	766	0.61
Cuiaba200e	200	4	3052	3415	96	**3052**	1	**3365**	56	3334	0	701	1
Belem300e	300	4	4031	**4434**	30	**4282**	0	4621	125	4563	0	2016	0
berlin52eA	52	3	8948	9094	65	**8948**	4	**9013**	24	8991	0	27	1
eil76eB	76	4	1940	**2069**	43	**1986**	0	2226	56	2129	0	61	0
rat99eB	99	5	3339	3525	71	**3339**	1	**3468**	54	3348	0	128	1
rd100eB	100	4	9951	10385	209	9994	0	**10055**	54	**9951**	1	161	1
st70eB	70	4	2037	**2158**	67	**2037**	1	2212	31	2137	0	54	0

Table 3. Results with time determined by Memetic Algorithm for Euclidean instances

Provided that the same computational effort (processing time) is fixed, throughout this section a statistical test for proportions comparison is applied. The test proposed by Taillard *et al.* (2008) compare success rates between two methods. In this paper, given two methods A and B, success of method A is stated when A achieves a better result than B for the same problem instance. The values for this test presented here were calculated with the tool available at http://qualopt.eivd.ch/stats/?page=stats with the one-tailed Taillard Test.

Column *p-value* of Table 2 shows that MA and GVND outperforms one another on 9 instances each, considering level of significance 0.05. With the processing time of MA fixed for both algorithms, column *p-value* of Table 3 shows that MA presents the best performance

on 9 instances and the GVND on 8 instances. These results show that the algorithms present similar performance concerning the number of instances each of them is significantly better than the other. If hitting the lowest value solution is set as a target for the algorithms, then Table 2 and 3 show that MA hits this target 19 and 17 times, respectively, whilst GVND hits the target 3 and 5 times. For these results, the test to compare success rates between two methods (Taillard *et al*., 2008) shows that the confidence level of the hypothesis that MA has success rate higher than GVND is 1 and 0.999968 when the maximum processing times are fixed by the GVND and the MA, respectively.

INSTANCE				GVND				MA				T(s)	p-value
Name	City	Car	#Best	Avg	SD	Best	Freq	Avg	SD	Best	Freq		
BrasilRJ14n	14	2	167	171	0	171	0	**167**	0	**167**	4	1	0
BrasilRN16n	16	2	190	203	0	203	0	**194**	2	**192**	0	1	0
BrasilPR25n	25	3	235	311	9	305	0	**255**	7	**239**	0	5	0
BrasilAM26n	26	3	204	242	6	239	0	**213**	4	**206**	0	5	0
BrasilMG30n	30	4	279	375	11	352	0	**330**	14	**298**	0	11	0
BrasilSP32n	32	4	285	336	16	298	0	**295**	5	**285**	1	12	0
BrasilRS32n	32	4	297	372	15	344	0	**337**	14	**297**	1	15	0
BrasilCO40n	40	5	655	826	42	755	0	**718**	37	**655**	1	39	0
BrasilNO45n	45	5	664	889	42	770	0	**753**	39	**664**	1	55	0
BrasilNE50n	50	5	707	1044	60	874	0	**844**	43	**761**	0	81	0
Londrina100n	100	3	1450	1783	80	1629	0	**1564**	51	**1450**	1	192	0
Osasco100n	100	4	1150	2000	60	1910	0	**1443**	109	**1265**	0	191	0
Aracaju200n	200	3	2467	3686	212	3223	0	**2802**	136	**2588**	0	903	0
Teresina200n	200	5	2192	3793	144	3261	0	**2480**	143	**2192**	1	1407	0
Curitiba300n	300	5	3676	6125	202	5680	0	**4081**	202	**3749**	0	3388	0
berlin52nA	52	3	1480	1777	82	1661	0	**1640**	51	**1543**	0	41	0
ch130n	130	5	2487	4706	307	3855	0	**2940**	245	**2487**	1	478	0
d198n	198	4	4807	7138	333	6529	0	**5332**	269	**4807**	1	1330	0
kroB150n	150	3	3824	5368	434	4414	0	**4312**	194	**3824**	1	464	0
rd100nB	100	4	1890	2953	169	2623	0	**2274**	118	**2083**	0	205	0

Table 4. Results with time determined by GRASP/VND for non Euclidean instances

Column *Freq* of Table 2 shows that, on average, 15% and 10% of the best solutions generated by one of the tested algorithms on the two experiments with fixed processing times are found by MA and GVND, respectively. Data presented in column *Freq* of Table 3 show that, on average, 13.5% and 10% of the best solutions are found by MA and GVND, respectively.

INSTANCE				MA				GVND				T(s)	p-value
Name	City	Car	#Best	Avg	SD	Best	Freq	Avg	SD	Best	Freq		
BrasilRJ14n	14	2	167	**167**	0	**167**	2	171	0	171	0	1	0
BrasilRN16n	16	2	190	**195**	3	**190**	1	203	0	203	0	1	0
BrasilPR25n	25	3	235	**256**	10	**235**	1	316	12	305	0	4	0
BrasilAM26n	26	3	204	**212**	4	**204**	1	242	5	239	0	5	0
BrasilMG30n	30	4	279	**328**	15	**279**	1	378	14	352	0	8	0
BrasilSP32n	32	4	285	**296**	7	**287**	0	331	14	300	0	13	0
BrasilRS32n	32	4	297	**340**	16	**304**	0	378	18	344	0	9	0
BrasilCO40n	40	5	655	**743**	33	**668**	0	839	41	710	0	20	0
BrasilNO45n	45	5	664	**764**	39	**667**	0	919	43	814	0	32	0
BrasilNE50n	50	5	707	**861**	61	**707**	1	1068	57	924	0	46	0
Londrina100n	100	3	1450	**1592**	50	**1471**	0	1767	85	1629	0	146	0
Osasco100n	100	4	1150	**1442**	139	**1150**	1	2046	80	1817	0	125	0
Aracaju200n	200	3	2467	**2744**	135	**2467**	1	3594	236	3106	0	922	0
Teresina200n	200	5	2192	**2551**	182	**2233**	0	3866	126	3611	0	836	0
Curitiba300n	300	5	3676	**4076**	216	**3676**	1	6050	160	5647	0	2384	0
berlin52nA	52	3	1480	**1642**	78	**1480**	1	1748	63	1661	0	38	0
ch130n	130	5	2487	**3020**	228	**2493**	0	4863	345	3813	0	237	0
d198n	198	4	4807	**5449**	318	**4887**	0	7407	378	6250	0	823	0
kroB150n	150	3	3824	**4259**	208	**3845**	0	5313	272	4871	0	418	0
rd100nB	100	4	1890	**2271**	143	**1890**	1	2962	180	2685	0	140	0

Table 5. Results with time determined by Memetic Algorithm for non Euclidean instances

The analysis of columns *p-value* of Tables 4 and 5 shows MA outperforms GVND on all non Euclidean instances, regardless the maximum processing time fixed for the experiments. All best results are found by MA. On average, MA finds approximately 2% of the best solutions on the 30 executions of each one of the 40 tested instances.

5. Conclusion

This chapter presented the Car Renter Salesman Problem (CARS), a new generalization of the classic Traveling Salesman Problem. An experimental investigation was carried out to compare two metaheuristic approaches proposed for this new problem: GRASP (Greedy Randomized Search Procedure) hybridized with VND (Variable Neighborhood Descent) and Memetic Algorithms. The algorithms were applied to 40 Euclidean and non Euclidean

instances of the CaRSLib benchmark which is proposed for this problem. An exact procedure established the optimal solutions of 4 from the 40 instances, whilst the proposed heuristics established the first upper limits for the remaining 36 instances. Statistical tests are applied to the results generated by the proposed algorithms in order to support conclusions on their behaviors concerning quality of solution.

To establish a fair basis of comparison for the proposed algorithms, the effect of the computational effort demanded by each algorithm is neutralized by comparing the performance of each algorithm according to fixed processing times. These execution times are established in accordance to the requirements of each algorithm for its best performance. Therefore, the proposed algorithms are tested twice, first with the processing times fixed by the best performance of one algorithm and then with the processing times fixed by the best performance of the other. Thus, a superior qualitative behavior can be considered conclusive when it holds for both processing time conditions.

The results of the computational experiments showed that for Euclidean instances the proposed algorithms present similar behavior with some advantage for MA concerning the number of best solutions found. For the set of non Euclidean instances, MA outperformed GVND on the whole set regardless the maximum processing time fixed for both algorithms. The MA also presented the best solution values for all non Euclidean instances.

This chapter presented also other six variants for the introduced problem, opening up the topic for future research.

6. References

Amdeberhan, T.; Manna, D. & Moll, V. H. (2008). The 2-adic Valuation of Stirling Numbers, *Experiment. Math.* 17 (1), pp. 69-82

Applegate, D.L.; Bixby, R.; Chvátal, V. & Cook, W. (1994). Finding cuts in the TSP: a preliminary report distributed at The Mathematical Programming Symposium, Ann Arbor, Michigan

Applegate, D.L.; Bixby, R.; Chvátal, V. & Cook, W. (2001). TSP cuts which do not conform to the template paradigm, In : *Computational Combinatorial Optimization*, M. Junger and D. Naddef (editors), pp. 261-303

Applegate, D.L. Bixby, R.; Chvátal, V. & Cook, W. (2006). The Traveling Salesman Problem: A Computational Study, Princeton University Press

Avis (2009). Avis Europe plc. Financial and Strategic Analysis Review, *Global Markets Direct*, 17, available at http://www.researchandmarkets.com/reports/690999

Avis (2010). Avis Budget Rental Car Funding, available at http://www.dbrs.com/research/232065/avis-budget-rental-car-funding-aesop-llc-series-2010-2-3/rating-report-series-2010-2-3.pdf

Car Rental (2008). Web Page, available at http://thrifty4.com/article.cfm/id/284920

Car (2008). Car Rental Business, *Global Strategic Business Report Global Industry Analysts*, Inc., 278, available at http://www.researchandmarkets.com/reports/338373/

Conover, W. J. (2001). Practical Nonparametric Statistics, John Wiley & Sons, 3rd Ed.

Conrad, C. & Perlut, A. (2006). Enterprise Rent-A-Car Hits New Billion-Dollar Revenue Mark for 3rd Consecutive Year, *Enterprise rent-a-car*, available at: http//www.enterpriseholdings.com/NewsReleases/Enterprise_FYO6_Sept06.pdf

Crowder, H. & Padberg, M. W. (1980). Solving large scale symmetric traveling salesman problems to optimality, *Management Science* 26, pp. 495-509

Edelstein, M. & Melnyk, M. (1977). The pool control system, *Interfaces* 8(1), pp. 21-36

Enterprise (2009). Enterprise Rent-A-Car Company - Strategic Analysis Review, *Global Markets Direct* 9, available at http://www.researchandmarkets.com/reports/690685/

Feo, T.A. & Resende, M.G.C. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization* 6, pp. 109-133

Garey, M. R. & Johnson, D. S. (1979). Computers and Intractability : A Guide to the Theory of NP-completeness. W.H. Freeman and Company, New York

Gutin, G. & Punnen, A. P. (2002). The Traveling Salesman Problem and Its Variations, Series: *Combinatorial Optimization* 12, Springer

Hertz (2009). The Hertz Corporation - Strategic Analysis Review, *Global Markets Direct* 12, available at http://www.researchandmarkets.com/reports/690555/

Hertz, A.; Schindl, D. & Zufferey, N. (2009). A solution method for a car fleet management problem with maintenance constraints, *Journal of Heuristics* 5, pp. 425–450

Krarup, J. (1975). The peripatetic salesman and some related unsolved problems, In: *Combinatorial Programming Methods and Applications*, Reidel, Dordrecht, pp. 173–178

Lia, Z. & Tao, F. (2010). On determining optimal fleet size and vehicle transfer policy for a car rental company, *Computers & Operations Research* 37, pp. 341-350.

Mladenovic, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research* 24, pp. 1097-1100

Montemanni, R.; Barta, J.; Mastrolilli, M. & Gambardella, L. M. (2007). The robust traveling salesman problem with interval data, *Transportation Science* 41(3), pp. 366–381

Moscato, P. (1989). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithm. *Caltech Concurrent Computation Program*. California Institute of Technology, USA

Pachon. J. E.; Iakovou, B.; Ip, C. & Aboudi, R. (2003). A synthesis of tactical fleet planning models for the car rental industry, *IIE Transactions* 35(9), pp. 907-916

Ralphs, T. K.; Kopman, L.; Pulleyblank, W. R. & Trotter, L. E. (2003). On the capacitated vehicle routing problem, *Mathematical Programming*, Ser. B 94, pp. 343–359

Reinelt, G. (1995). TSPLIB95, available at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95

Russel, R. (1977). An effective heuristic for the m-tour traveling salesman problem with some side conditions, *Operations Research* 25, pp. 517-524

Taillard, E.; Waelti, P. & Zuber, J. (2008). Few statistical tests for proportions comparison, *European Journal of Operational Research*, vol. 185, pp. 1336–1350

Wikipedia (2010). Enterprise Rent-a-Car page, available at http://en.wikipedia.org/wiki/Enterprise_Rent-a-Car

Xiong, Y.; Golden, B. & Wasil, E. (2007). The Colorful Traveling Salesman Problem, Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies, *Operations Research/Computer Science Interfaces Series* 37, Springer US, pp. 115-123

Yang, Y.; Jin, W. & Hao, X. (2008). Car Rental Logistics Problem: A Review of Literature, *IEEE International Conference on Service Operations and Logistics, and Informatics.* IEEE/SOLI 2008. 2, pp. 2815–2819

17

Multi-Objective Scheduling on a Single Machine with Evolutionary Algorithm

A. S. Xanthopoulos, D. E. Koulouriotis and V. D. Tourassis
Democritus University of Thrace
Greece

1. Introduction

In this chapter, the single-machine scheduling problem (SMSP) is addressed by adopting a multi-objective perspective. A finite set of independent jobs has to be scheduled on a single server that is continuously available and has the ability to process only one job at a time. Once a job has started its processing it cannot be interrupted and no idle time between successive jobs is allowed. Each job is characterized by its processing time and due date, both integer numbers.

Job scheduling considering only one criterion can be thought of as an over-simplification of the problem, since there are several objectives related to this problem, namely sum of earliness/tardiness, max earliness/tardiness and so forth. There is a growing trend in the relevant literature to study scheduling problems under multiple objectives. Some indicative examples can be found in the works of (Behnamian et al., 2009); (Yagmahan et al., 2010); (Loukil et al., 2005); (Moslehi & Mahnam, 2010); (Choobineh et al, 2006) and (Gupta & Sivakumar, 2005).

In this chapter we consider two objectives; sum of earliness and sum of tardiness. In general, these objectives are conflicting meaning that a solution that improves one objective function will deteriorate the other. In the absence of optimization criteria preferences each one objective needs to be dealt with explicitly, giving rise to the concept of Pareto optimality. The solution to the bi-objective single-machine scheduling problem is a set of points in objective space known as Pareto front and all points along the Pareto front share the following property: the value of a certain objective function can be improved only by degrading the value of at least one of the remaining objective functions.

The derivation of the Pareto set using exhaustive search is limited to relatively small instances of the problem due to the combinatorial explosion that takes place as the dimensionality increases. Consequently, the application of meta-heuristics appears to be well-suited for this type of problems. The reader is referred to (Bagchi, 1999) for further information on multi-objective scheduling using evolutionary techniques. In this chapter, a multi-objective algorithm evolutionary algorithm (MOEA) for approximating the Pareto front in single-machine scheduling problems is described and analyzed. The MOEA is tested in a series of randomly generated SMSP instances of various configurations. In order to select appropriate parameters for the MOEA we apply techniques from the field of design of experiments (DoE), specifically general fractional factorial designs. The solutions found by the MOEA are

compared to the true Pareto front and to each other based on four metrics: generational distance, non-uniformity of solution distribution, hypervolume and maximum spread.
The structure of this chapter is presented hereafter. In Section 2 the multi-objective version of the SMSP is described. Section 3 is devoted to the presentation of the salient features of SPEA2, the MOEA which was applied to solve the underlying optimization problem. Four metrics for comparing Pareto sets are presented in Section 4. The numerical results from the series of experiments are presented and commented upon in Sections 5 to 5.5. Finally, Section 6 contains the concluding remarks and some directions for future research.

2. Problem description

The formal description of the problem under consideration is given in this section. Let $J = \{j_1, j_2, ..., j_n\}$ be a finite set of jobs. The assumptions pertaining to the set J are the following.

- each job constitutes an indivisible whole, i.e. it cannot be broken down to elementary operations.
- the release time of the *i*-th job is denoted as r_i and for all *i*, $r_i = 0$, meaning that all jobs are available for processing from $t = 0$.
- each job is associated with a non-negative integer processing time $p_i \in Z^+$.
- each job has a due date $d_i \in Z$. Due dates can assume negative values and the meaning of a negative due date is that the related job is already delayed.

All jobs in J have to be processed on a single machine to which, the following assumptions apply.

- the machine can only process one job at a time.
- once the machine has started processing a job it cannot be interrupted.
- the machine does not undergo failures nor does it suspend its operation for maintenance or other reasons.
- once a job has completed its processing it exits the system immediately, and as a consequence, the machine is never blocked.

The underlying decision problem is to determine the sequence according to which all jobs will be processed on the machine under two additional assumptions:

- the machine setup time for shifting from one job to another is always zero.
- no machine idle time between successive job is allowed.

The sequence of jobs is called a *schedule* and a possible schedule is a permutation of the elements that belong to the set $\{1,2,...,n\}$. Clearly, an admissible schedule is a *n*-dimensional vector **s** where s_i is the position of the *i*-th job in the schedule and the number of all plausible schedules is $n!$. For a given schedule, the completion time of the *i*-th job is $c_i = w_i + p_i$ where w_i is the waiting time related to that job. The waiting time of job j_i is the sum of the processing times of all jobs that were completed up to the point that j_i starts its processing:

$$w_i = \sum_{J_i^p} p_k \ , \ J_i^p = \{j_k \in J : s_k < s_i\} \tag{1}$$

The *earliness* of the *i*-th job is $e_i = \max\{0, d_i - c_i\}$ and the *tardiness* of job j_i is $t_i = \max\{0, c_i - d_i\}$.

The single-objective approach to compare two alternative schedules $\mathbf{s}$ and $\mathbf{s}'$ is to define an objective metric f that would assign a numerical value $f(\mathbf{s})$ to every $\mathbf{s} \in S$, where S is the set of possible schedules. However, as stated in the introduction, more that one objective functions can be associated with the problem under consideration. In this chapter two objective functions are considered simultaneously, forming the objective vector described in (2).

$$\mathbf{f} = [f_1, f_2] = [\sum_{i=1}^{n} e_i, \sum_{i=1}^{n} t_i] \tag{2}$$

The first element of the objective vector is the sum of earliness values of all jobs whereas the second objective element is the sum of all tardiness values. Both of these two quantities are to be minimized. The adoption of the two objective functions for quantifying the system's performance is compatible with the philosophy of Just In Time manufacturing, according to which an end-item should be ideally complete its processing exactly at the time when it is needed. In the total absence of objective function preferences each objective must be dealt with explicitly. The concept of Pareto dominance can be used to compare two candidate solutions in the multi-objective setting where each objective component is treated separately. In the minimization problem treated in this chapter, an objective vector $\mathbf{f}_a$ dominates another vector $\mathbf{f}_b$, *iff*

$$f_{a,i} \leq f_{b,i}, \forall i \in \{1,2\} \tag{3}$$

and

$$\exists j \in \{1,2\} \text{ such that } f_{a,j} < f_{b,j} \tag{4}$$

Pareto dominance in this case is denoted by $\mathbf{f}_a \prec \mathbf{f}_b$. Using the notion of Pareto dominance, the objective functions that constitute the objective vector are characterized as *partially conflicting*, meaning that there is at least one decision vector $\mathbf{s}$ dominated by some other vector that belongs to S.

The solution to the multi-objective optimization problem stated in this section is the global Pareto optimal set *P*, that is, the set of objective vectors which are not dominated by any other feasible objective vector.

3. SPEA2

Some examples of popular multi-objective evolutionary algorithms (MOEAs) are PESA (Corne et al. 2000), PESA-II (Corne et al., 2001), SPEA (Zitzler & Thiele, 1999), SPEA2 (Zitzler et al., 2001), NSGA-II (Deb et al., 2002), MOEA (Tan et al., 1999), ESPEA (Everson et al. 2002), DMOEA (Lu & Yen, 2002) and μGA2 (Pulido and Coello Coello, 2003). The performance of a MOEA is typically assessed on the basis of its ability to approximate effectively the true global Pareto front and to produce a uniformly distributed set of solutions in addition to its consistency and robustness. The SPEA2 algorithm has been shown in (Tan et al, 2005) and (Zitzler et al., 2002) to perform very well in comparison to other MOEAs in a variety of different test problems and under several MOEA-specific performance metrics, and was therefore adopted as the search algorithm for the purposes of this investigation. The key features of SPEA2 are outlined in the remaining of this section. For a detailed description of the SPEA2 algorithm the reader is referred to (Zitzler et al., 2002).

The SPEA2 algorithm evolves a population of candidate solutions while maintaining an external population called *archive* where non-dominated individuals found during the evolutionary process are stored. The population is initialized with randomly generated individuals and its size remains constant throughout the execution of the algorithm. Initially the archive is empty but after the first generation of candidate solutions is evaluated the archive is resized to contain a pre-specified number of individuals and its size is kept fixed hereinafter. In each iteration, all individuals in the population and the archive are assigned a fitness value. The SPEA2 algorithm employs a sophisticated fitness assignment scheme, where the fitness of an individual is given by the sum of the *strengths* of its dominators plus its *density*. The strength of an individual represents the number of solutions that it dominates, while the density of an individual is the inverse of the distance from its *k*-th nearest neighbour (another individual) in objective space. After all elements in the population and the archive have been assigned a fitness value, the non-dominated individuals are copied to the *temporary archive*. If the size of the temporary archive exceeds the pre-specified threshold then it is truncated, whereas if the size of the archive is smaller than the threshold then it is expanded by adding to it the best (in terms of fitness value) dominated individuals from the population and the archive. For the truncation of the temporary archive a sequential procedure is applied, where in each pass of the procedure the element with minimum distance (in objective space) to another individual is deleted. Ties are broken by considering the second smallest distance, the third etc. After the temporary archive reaches the specified size, all of its contents are stored in the archive and the temporary archive is deleted. Subsequently, the selection operator is applied on the archive and the selected individuals are subjected to standard genetic operations in order to produce the next population. The algorithm terminates when the stopping criterion is satisfied and returns the archived non-dominated set of solutions. The pseudo-code for the SPEA2 algorithm is presented below, where $|\cdot|$ denotes cardinality of set and S symbolizes the size of the archive:

1. generate initial population *Pop* and empty archive *Arc*
2. assign fitness values to all individuals in *Pop* and *Arc*
3. copy all non-dominated solutions in *Pop* and *Arc* in temporary archive *TempArc*
   a. IF $|TempArc| > S$
      apply truncation procedure on *TempArc*
   b. IF $|TempArc| < S$
      add dominated individuals from *Pop* and *Arc* to *TempArc* based on fitness
4. set $Arc \leftarrow TempArc$, delete *TempArc*
   a. IF $STOPPING\_CRITERION = TRUE$
      return *Arc*. Terminate
   b. ELSE
      go to Step 5
5. apply selection operator on *Arc*. Apply genetic operators on the selected individuals and store the offspring in *Pop*. Go to Step 2

In the remaining of this section we discuss the implementation of the key features of the SPEA2 algorithm that was applied to the problem addressed in this chapter, namely the encoding of candidate solutions and the selection strategy/genetic operators in Step 5 of the algorithm.

All individuals in the population are encoded as vectors of *random keys*. A random key is simply a real number which assumes values in the range $[0,1]$. In order to translate a vector of random keys to a schedule the elements of the chromosome are sorted in descending order. An example of this decoding scheme involving 4 jobs is depicted below.

jobs:	j_1	j_2	j_3	j_4
chromosome:	0.98	0.12	0.56	0.78
sorted genes:	0.98	0.78	0.56	0.12
schedule:	1	4	3	2

The major advantage of random key encoding is that every possible chromosome decodes to a feasible schedule and therefore, there is no need for customized initialization routines and genetic operators. *Tournament selection* is responsible for selecting individuals from the archive for recombination. According to this selection scheme *TourSize* individuals are chosen with the same probability from the archive and the fittest individual from that subset is chosen to constitute a parent which will be recombined with another individual to produce offspring. Parameter *TourSize* is commonly known as the *tournament size* and tournament selection mechanisms with $TourSize = 2$ are referred to as binary tournaments. The procedure is iterated until the required number of individuals has been selected. The selected individuals are recombined according to the *intermediate recombination* technique. The *i*-the element of the offspring is computed according to the following equation:

$$c_i^o = u_i c_i^{p1} + (1 - u_i) c_i^{p2} \tag{5}$$

where c_i^{p1} and c_i^{p2} are the *i*-th elements of the two parents and u_i is a random variable uniformly distributed in $[-\delta, 1+\delta]$. Parameter δ determines the hypercube to which the produced offspring is possible to fall in. The fraction of the new population which consists of individuals generated via recombination is determined by parameter *RecFrc*, whereas the remaining individuals are produced using *migration*, i.e. they are generated at randomly.

4. Performance metrics for non-dominated sets

A number of metrics for comparing non-dominated sets have been proposed in the relevant literature. These metrics largely fall into two categories: i) performance metrics that require the true Pareto front to be known and, ii) performance measures that do not involve the true Pareto front in the computation and can be used to compare two or more non-dominated sets directly. In this investigation we consider the following four measures: a) the *generational distance GD* (Tan et al., 2005), b) the metric of *non-uniform distribution of solutions U*, c) the *maximum spread MS* of the obtained solutions and, d) the *hyper-volume HV* (Tan et al., 2005). The rest of this section briefly describes these performance metrics.

The metric of generational distance reflects the "distance" between a Pareto optimal set *A* and the true Pareto front *P*. It is defined as:

$$GD = \left(\frac{1}{|A|} \sum_{i=1}^{|A|} d_i^2 \right)^{1/2} \tag{6}$$

where $|A|$ is the cardinality of A and d_i is the Euclidean distance between the i-th element of A and the nearest element that belongs to P.
The metric of non-uniform distribution of solutions $U \in \Re^+$ measures how uniformly the individual solutions are distributed in a Pareto optimal set A. It is defined as:

$$U = \sqrt{\frac{\sum_{i=1}^{|A|-1}\left(d_{i,i+1}/\overline{d}-1\right)^2}{|A|-1}} \tag{7}$$

where $|A|$ is the cardinality of A, $d_{i,i+1}$ symbolizes the Euclidean distance between two successive members in A and $\overline{d}$ is the average distance. Large deviations of the distances $d_{i,i+1}$ from the average distance result in high values of U, therefore a Pareto front with a lower value of this metric is preferable to another front with higher U value.
The performance measure called max spread is an indicator of the range in objective space covered by a Pareto optimal set A. Its formal definition is given in Equation (8):

$$MS = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\max_{i=1}^{|A|} f_j^i - \min_{i=1}^{|A|} f_j^i\right)^2} \tag{8}$$

where $|A|$ is the cardinality of A, N is the number of objective functions and f_j^i is the value of the j-th objective yielded by the i-th element in A. Higher values of MS signify better performance
Hyper-volume HV measures the size of the objective space that is dominated by the elements of a non-dominated set A. It is defined as $HV = volume\left(\bigcup_{i=1}^{|A|} v_i\right)$, where v_i is the hypercube with diagonal corners the objective vector $\mathbf{f}^i = [f_1^i, f_2^i, ..., f_N^i]$ of the i-th element in A and the anti-optimal objective vector $\mathbf{f}^{\max} = [f_1^{\max}, f_2^{\max}, ..., f_N^{\max}]$, where $f_j^{\max} = \max_{i=1}^{|A|} f_j^i$. Again, $|A|$ denotes the cardinality of A, N is the number of objective functions and f_j^i is the value of the j-th objective for the i-th element in A.

5. Results

Section 5 and its subsections are devoted to the presentation of the results that were obtained from the application of the SPEA2 algorithm to twelve instances of the single-machine scheduling problem described in Section 2. The randomly generated instances of the problem which drive the experimental investigation are given in the following subsection.

5.1 SMSP instances

All instances were selected at random from an extensive set of SMSP instances which was generated by (Valente & Goncalves, 2009) and can be found online at http://www.fep.up.pt/docentes/jvalente/benchmarks.html. An SMSP instance is simply a set of processing time-due date pairs and it is identified by using the naming convention adopted by (Valente & Goncalves, 2009) specifically *PV-N-T-R*, where *PV* symbolizes the processing time variability, *N* is the number of jobs, *T* is the tardiness factor and *R* is the due date range. All twelve instances considered in this chapter are of type *H-10-x-y*, meaning that they consist of ten jobs with high (*H*) processing time variability, i.e. the processing

times are uniformly distributed in the interval $[1-100]$. Each job has a due date which is drawn from the uniform distribution defined on the interval $[P(1-T-R/2), P(1-T+R/2)]$, where P is the sum of the processing times of all jobs for a particular instance and parameters P and R assume non-negative real values. In this study we consider three levels for the tardiness factor T, specifically 0.4, 0.6, and 0.8.

H-10-0.4-0.2		H-10-0.4-0.4		H-10-0.4-0.6		H-10-0.4-0.8	
p_i	d_i	p_i	d_i	p_i	d_i	p_i	d_i
73	439	9	236	37	144	64	473
51	411	35	290	65	254	82	290
69	411	59	296	34	206	88	159
82	356	47	180	9	239	9	447
51	353	46	294	18	118	23	378
59	363	38	295	39	226	47	144
41	475	44	291	7	206	30	525
85	390	83	225	47	184	97	388
86	377	61	218	21	162	30	461
89	461	12	222	44	124	67	287

Table 1. Instances *H-10-0.4-y*

Additionally, for each tardiness factor level four due date range levels (0.2, 0.4, 0.6, 0.8) are examined. The parameters of the twelve SMSP instances are shown in Tables 1 to 3.

H-10-0.6-0.2		H-10-0.6-0.4		H-10-0.6-0.6		H-10-0.6-0.8	
p_i	d_i	p_i	d_i	p_i	d_i	p_i	d_i
36	135	62	248	53	310	85	117
54	168	45	228	96	75	32	114
84	115	25	105	71	149	13	197
24	137	21	128	81	332	44	135
34	168	78	127	26	61	60	348
33	146	42	161	13	245	100	282
20	129	14	127	2	314	15	198
31	133	38	91	33	323	3	65
5	162	38	150	97	70	84	240
50	126	53	200	36	190	38	183

Table 2. Instances *H-10-0.6-y*

H-10-0.8-0.2		H-10-0.8-0.4		H-10-0.8-0.6		H-10-0.8-0.8	
p_i	d_i	p_i	d_i	p_i	d_i	p_i	d_i
53	108	4	107	85	212	50	-44
89	90	99	160	98	94	70	-52
59	147	41	115	13	7	45	68
31	70	89	116	46	8	79	168
56	171	45	112	99	9	73	196
91	163	41	216	28	73	68	60
73	191	99	51	36	220	28	44
81	132	49	157	6	211	39	192
54	193	73	94	100	250	12	321
57	182	39	83	19	84	94	118

Table 3. Instances *H-10-0.8-y*

5.2 Configuration of MOEAs

The parameters of the multi-objective genetic algorithm (SPEA2) which was used to approximate the true Pareto front for each instance are

- the population size
- the chromosome length
- the maximum number of iterations
- the size of the archive
- the tournament size (selection operator)
- the recombination fraction
- parameter δ (recombination operator)

Since each instance consists of ten jobs and individuals are encoded as sequences of random keys, the chromosome length is equal to 10 in all executions of the algorithm. The exhaustive search algorithm used to find the true Pareto fronts conducts $10! = 3,628,800$ feasible schedule evaluations for each problem instance. We selected the total number of evaluations performed by the MOEA in each instance to be fixed to the level of 35,000, which is approximately 1% of the number of the exhaustive search evaluations. As a consequence, the adjustment of the population size implicitly determines the maximum number of iterations too. The size of the archive was set to be equal to the size of the true Pareto front for each instance. Moreover, the recombination fraction determines the fraction of individuals created by recombination in a new generation, where the remaining individuals are generated by the migration operator.

As a consequence, the parameters which participate to the fractional factorial experiments conducted for each problem instance are

i. the population size (*PopSize*)
ii. the tournament size (*TourSize*)
iii. the recombination fraction (*RecFrc*)
iv. the parameter delta (δ)

The levels for factor i were set to be 50 and 100, whereas the levels for factor ii were 2 (binary tournament) and 10, or in the case were the arc consisted of less than ten elements, the high level of parameter ii was equal to the arc size. 0.6 was the low level for factor iii (recombination fraction) and 0.8 the high level. Finally, the two levels of parameter δ were 0 and 0.25. After the levels of the factors had been defined a 2^{4-1} fractional factorial design of resolution IV was generated by employing the Franklin-Bailey algorithm (Box et al., 2005). This type of design separates main effects and requires 8 treatments (parameter sets) to be evaluated, i.e. it is considerable more economical than the corresponding full 2^4 factorial experiment which requires 16 runs. The resulting experimental designs are presented in Table 4. For every design the evolutionary algorithm is executed 3 times and the non-dominated set which has the highest number of elements identical to that of the corresponding true Pareto front is selected.

	PopSize	*TourSize*	*RecFrc*	δ
design 1	50	2	0.6	0
design 2	50	2	0.8	0.25
design 3	50	10(archive size)	0.6	0.25
design 4	50	10(archive size)	0.8	0
design 5	100	2	0.6	0.25
design 6	100	2	0.8	0
design 7	100	10(archive size)	0.6	0
design 8	100	10(archive size)	0.8	0.25

Table 4. Experimental designs

5.3 Comparative evaluation – instances H-10-0.4-y

The cardinalities of the true Pareto fronts for instances H-10-0.4-0.2, H-10-0.4-0.4, H-10-0.4-0.6 and H-10-0.4-0.8 are 2, 10, 6 and 3, respectively. For all of these four instances the best

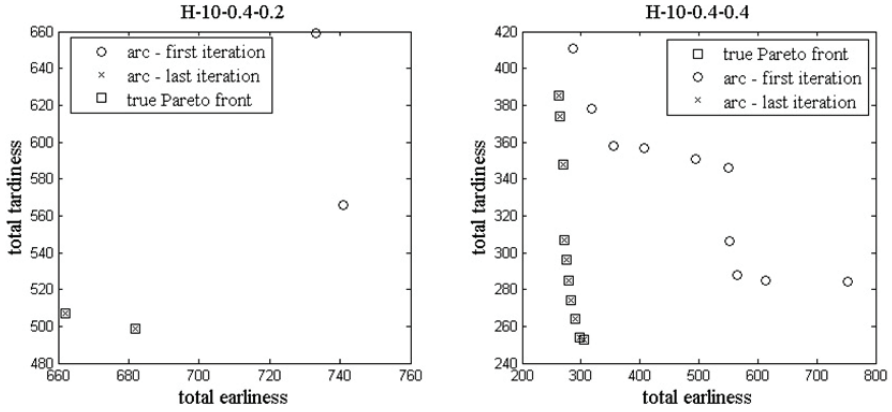


Fig. 1. True Pareto fronts, archives in first/last iteration – instances *H-10-0.4-0.2* and *H-10-0.4-0.4*

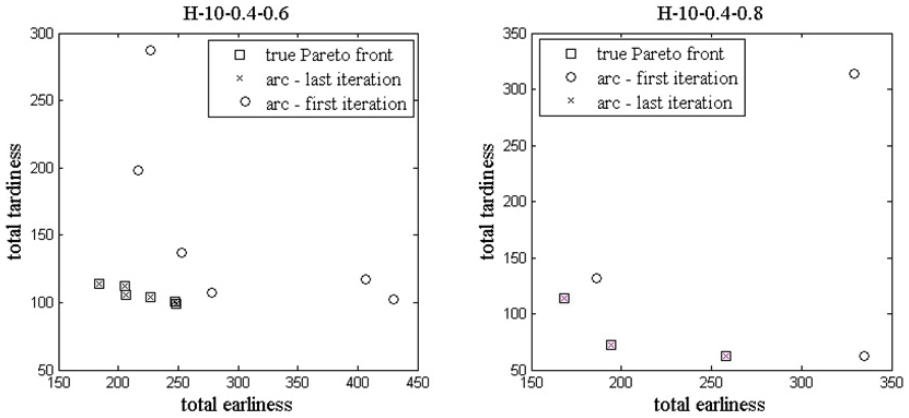


Fig. 2. True Pareto fronts, archives in first/last iteration – instances *H-10-0.4-0.6* and *H-10-0.4-0.8*

design for the MOEA was found to be the third which corresponds to $PopSize = 50$, $RecFrc = 0.6$, $\delta = 0.25$ and *TourSize* set to the high level. The non-dominated sets returned by the MOEAs initialized with that parameter set were identical to the true Pareto fronts for all problem instances examined in this subsection. The non-dominated sets related to executions of the MOEA with different parameter sets were not globally Pareto optimal and therefore further comparisons between the various sets using the metrics presented in Section 4 is redundant. Figures 1 and 2 illustrate the true Pareto fronts of the four instances of this subsection, and the individuals (in objective space) that populate the archive in the first and last iteration of the best evolutionary algorithm execution.

5.4 Comparative evaluation – instances H-10-0.8-y

The true Pareto fronts for instances H-10-0.8-0.2, H-10-0.8-0.4, H-10-0.8-0.6 and H-10-0.8-0.8 consist of 5, 8, 5 and 7 elements, respectively. Similarly to the previous four instances the

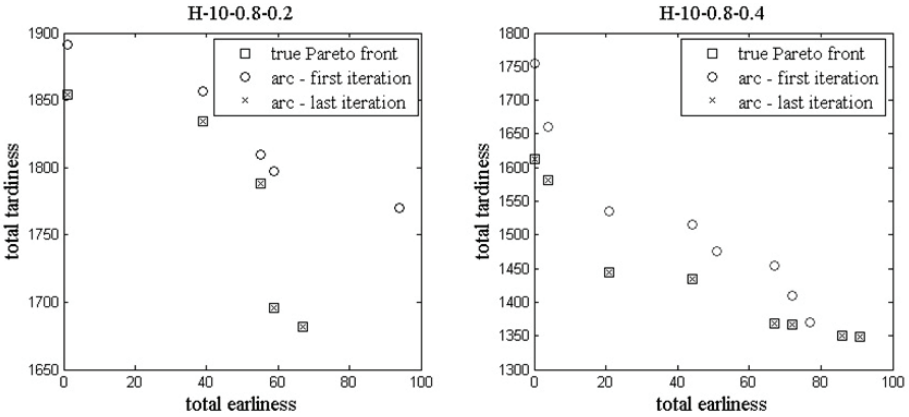


Fig. 3. True Pareto fronts, archives in first/last iteration – instances *H-10-0.8-0.2* and *H-10-0.8-0.4*

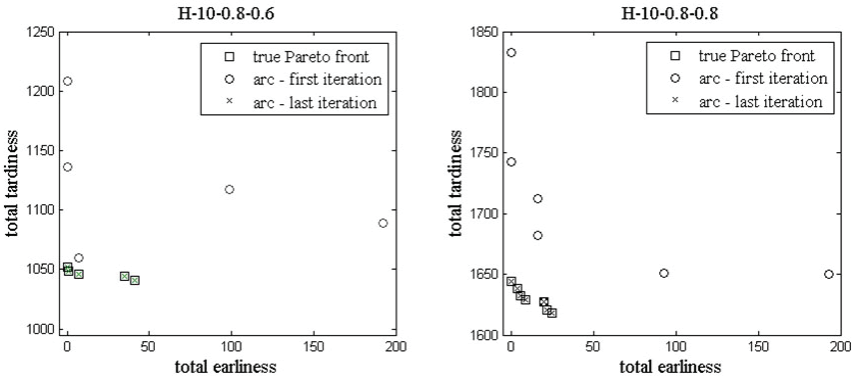


Fig. 4. True Pareto fronts, archives in first/last iteration – instances *H-10-0.8-0.6* and *H-10-0.8-0.8*

best parameter set for the MOEA was $PopSize = 50$, $RecFrc = 0.6$, $\delta = 0.25$ and *TourSize* set to the high level. Executions of the MOEA with those parameters managed to deliver the true Pareto fronts in all four cases whereas executions with alternative parameter sets failed to do so. Figures 3 and 4 present the actual Pareto fronts of the four instances of this subsection, and the individuals that constitute the archive in the first and last generation of the best evolutionary algorithm execution.

5.5 Comparative evaluation – instances H-10-0.6-y

The sizes of the non-dominated sets related to the last four instances of this experimental investigation are generally higher than those of the eight instances discussed in the previous two subsections. More specifically, the actual Pareto fronts for instances H-10-0.6-0.2, H-10-0.6-0.4, H-10-0.6-0.6 and H-10-0.6-0.8 have 13, 10, 14 and 9 elements, respectively. In this subset of problem instances, the true Pareto fronts corresponding to instances H-10-0.6-0.4 and H-10-0.6-0.8 were computed exactly by the MOEA. The parameter values related to the third design of the fractional factorial experiment were found to be the best for this series of problem instances too. In the case of instance H-10-0.6-0.2 the MOEA returned a set of

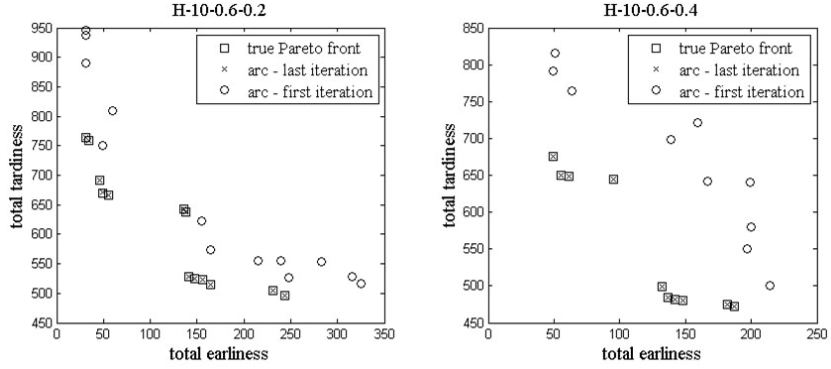


Fig. 5. True Pareto fronts, archives in first/last iteration – instances *H-10-0.6-0.2* and *H-10-0.6-0.4*

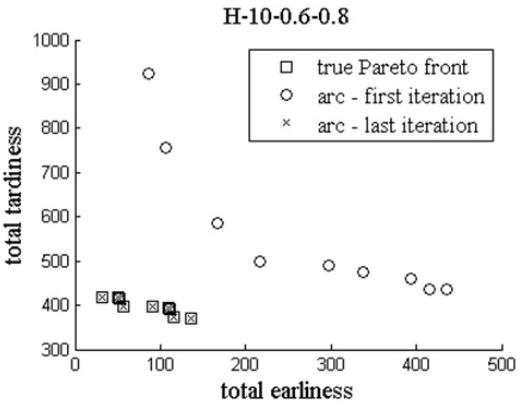


Fig. 6. True Pareto front, archive in first/last iteration – instance *H-10-0.6-0.8*

non-dominated solutions that is differentiated from the actual Pareto front only in one element, where the globally Pareto optimal element is $[49,670]$ and the corresponding element of the MOEA front is $[49,672]$. It is reasonable to argue that the approximation of the actual Pareto front by the MOEA is excellent in this problem instance too.

However, in instance H-10-0.6-0.6 the actual Pareto front was not obtained exactly from any execution. In addition to that, all eight non-dominated sets related to different combinations of parameter values of the MOEA were under-populated, i.e. none of them consisted of 14 elements as the true Pareto front because some elements were repetitions of others. This effect can be largely attributed to the random key encoding scheme which does not exclude the possibility that two or more seemingly different chromosomes decode to the same schedule. For example, both $[0.7,0.2,0.1]$ and $[0.6,0.35,0.05]$ decode to $[1,2,3]$. The true Pareto front and the fronts returned by the MOEA for each one of the eight parameter sets are displayed in Figures 7 and 8. Note that the actual Pareto front can be obtained if the non-dominated sets related to experimental designs 2, 4 and 6 are combined. Since no locally Pareto optimal set is clearly superior to the others, the metrics described in Section 4 can assist in establishing a ranking of the eight non-dominated sets of solutions. The performance of the eight fronts regarding the metrics of generational distance (*GD*), hypervolume (*HV*), non-uniformity of solutions (*U*) and maximum spread (*MS*) are presented in Table 5. We reiterate that for the metrics of hypervolume and maximum spread high values are preferable, whereas the opposite holds for the non-uniformity of solutions and the generational distance. The information contained in Table 5 is somewhat hard to interpret and in order to facilitate a clearer presentation of the results we transform the elements of each row to the interval $[0,1]$ and construct the *spider graphs* displayed in Figure 9. Note that design 3 corresponds to the minimum values of all four performance metrics and is therefore depicted as a single point in the left pane of Figure 9. By observing Figure 9 one can sees that the local Pareto front of the third design is the best regarding the measures of *U* and *GD* but in the same time exhibits the worst performance in terms of MS and *HV*. On the other hand, the non-dominated set linked to the seventh design achieves the highest values of *HV* and *MS* but is also the worst when it comes to minimizing *GD*. All other locally Pareto optimal sets correspond to different trade-offs between the four metrics under consideration.

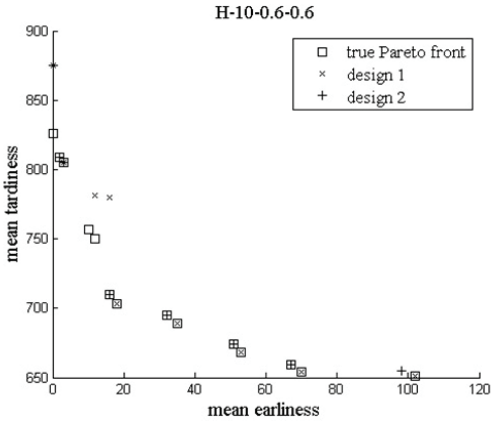


Fig. 7. True Pareto front, archives in last iteration of designs 1 and 2 – instance *H-10-0.6-0.6*

	design 1	design 2	design 3	design 4	design 5	design 6	design 7	design 8
GD	19.85	17.44	7.85	26.18	29.76	16.33	39.61	17.76
HV	18059	16948	12248	19288	20568	18323	24300	17690
U	0.67	0.75	0.45	0.49	0.59	0.54	0.65	0.56
MS	174.04	170.3	139.36	185.93	193.17	174.04	215.37	172.88

Table 5. Non-dominated sets performance metrics, instance *H-10-0.6-0.6*

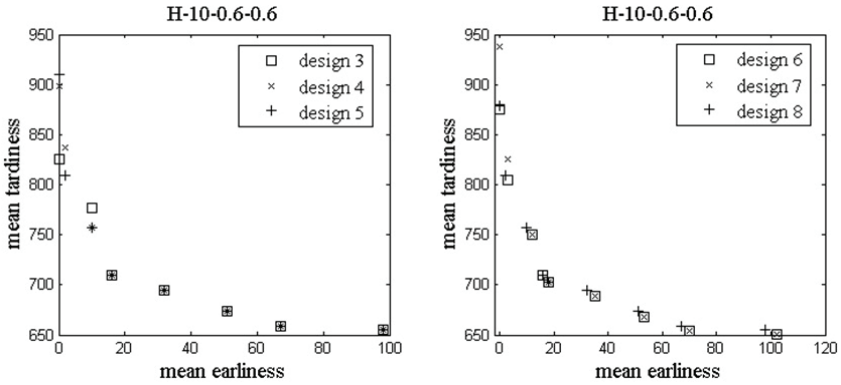


Fig. 8. Archives in last iteration of designs 3 to 8 – instance *H-10-0.6-0.6*

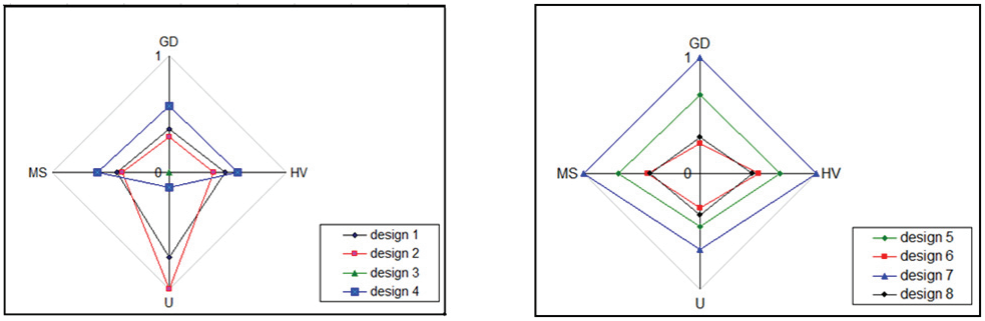


Fig. 9. Normalized performance metrics of non-dominated sets (designs 1-8) – instance *H-10-0.6-0.6*

6. Conclusion

The application of a MOEA to the multi-objective single-machine scheduling problem was investigated experimentally. The objective functions of interest were the total tardiness and the total earliness. For the purposes of this investigation the true Pareto fronts of 12 random problem instances were computed using exhaustive search. All instances consisted of ten jobs but the sizes of the Pareto fronts varied from one instance to another, indicating that the structure of the Pareto fronts does not depend only on the dimensionality of the problem. In order to tune the parameters of the MOEA to each problem instance we conducted 12 fractional factorial experiments. The use of methods from the field of Design of Experiments to select parameters for the evolutionary algorithm is appealing because they offer the ability to select an efficient parameter set with limited computational cost. The MOEA returned the actual Pareto front in ten out of twelve problem instances and computed a very good approximation of the true Pareto front in one instance. However, it exhibited a rather mediocre performance in a specific problem instance. An important advantage of the evolutionary approach to the underlying problem is that in all cases, the MOEA conducted only 1% of the number of evaluations performed by the exhaustive search algorithm. A possible direction of future research would be to apply evolution to problems with high dimensionality where the true Pareto front cannot be obtained exactly and compare the results with those from other meta-heuristic algorithms.

7. References

Bagchi, T. P. (1999). *Multiobjective scheduling by genetic algorithms*, Kluwer Academic Publishers, ISBN 0-7923-8561-6, Boston

Behnamian, J.; Fatemi Ghomi, S. M. T. & Zandieh, M. (2009). A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. *Expert Systems with Applications*, Vol. 36, No. 8, October 2009, pp 11057-11069, ISSN 0957-4174

Box, G. E. P.; Hunter, W. G. & Hunter, J. S. (2005). *Statistics for Experimenters: Design, Innovation and Discovery*. Wiley-Interscience, ISBN 0-471-71813-0, New Jersey

Corne, D. W.; Jerram, N. R.; Knowles, J. D. & Oates, M.J. (2001). PESA-II: region-based selection in evolutionary multiobjective optimization, *Proceedings of the Genetic and*

Evolutionary Computation Conference (GECCO-2001), pp 283-290, ISBN 1-59593-010-8, Seattle, Washington, USA, July 2006, ACM

Corne, D. W.; Knowles, J. D. & Oates, M. J. (2000). The Pareto envelope-based selection algorithm for multiobjective optimization. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp 839-848, ISBN 3-540-41056-2, Paris, France, September 200, Springer-Verlag, Berlin

Deb, K.; Pratap, A.; Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, April 2002, pp 182-197, ISSN 1089-778X

Everson, R. M.; Fieldsend, J. E. & Singh, S. (2002). Full elite sets for multi-objective optimization. *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002)*, pp 343-354, Exeter, Devon, UK, April 2002, Springer-Verlag, Berlin

Choobineh, F. F.; Mohebbi, E. & Khoo, H. (2006). A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times . *European Journal of Operational Research*, Vol. 175, No. 1, November 2006, pp 318-337, ISSN 0377-2217

Gupta, A. K. & Sivakumar, A. I. (2005). Multi-objective scheduling of two-job families on a single machine. *Omega*, Vol. 33, No. 5, October 2005, pp 399-405, ISSN 0305-0483

Loukil, T.; Teghem, J. & Tuyttens, D. (2005). Solving multi-objective production scheduling problems using metaheuristics. *European Journal of Operational Research*, Vol. 161, No. 1, February 2005, pp 42-61, ISSN 0377-2217

Lu, H. & Yen, G. G. (2002). Dynamic population size in multiobjective evolutionary algorithms. *Proceedings of the 2002 Congress on Evolutionary Computation*, pp 1648-1653, ISBN 0-7803-7282-4, Honolulu, Hawaii, May 2002, IEEE Computer Society, Washington DC

Moslehi, G. & Mahnam, M. (2010). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, In Press, Available online 10 August 2010, ISSN: 0925-5273

Pulido, G. T.; & Coello Coello, C. A. (2003). The micro genetic algorithm 2: towards online adaptation in evolutionary multiobjective optimization, *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO2003)*, pp 252-266, ISBN 3-540-01869-7, Faro, Portugal, April 2003, Springer-Verlag, Berlin

Tan, K. C.; Lee, T. H. & Khor, E. F. (1999). Evolutionary algorithms with goal and priority information for multi-objective optimization, *Proceedings of the 1999 IEEE International Congress on Evolutionary Computation*, Vol. 3, pp 106-113, ISBN 0-7803-5 536-9, Washington DC, USA, July 1999, IEEE Computer Society, Washington DC

Tan, K. C.; Lee, T. H. & Khor, E. F. (2005). *Multiobjective Evolutionary Algorithms and Applications*, Springer-Verlag, ISBN 1-85233-836-9, London

Valente, J. M. S. & Goncalves, J. F. (2009). A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic

tardiness penalties. *Computers and Operations Research,* Vol. 36, No. 10, October 2009, pp 2707-2715, ISSN 0305-0548

Yagmahan, B. & Mutlu Yenisey, M. (2010). A multi-objective ant colony system algorithm for flow shop scheduling problem. *Expert Systems with Applications*, Vol. 37, No. 2, March 2010, pp 1361-1368, ISSN 0957-4174

Zitzler, E.; Laumanns, M. & Thiele, L. (2001). SPEA2: improving the strength Pareto evolutionary algorithm, Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich

Zitzler, E.; Laumanns, M. & Thiele, L. (2002). SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization, In: *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN 2001),* K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou and T. Fogarty *(*Eds.*)*, pp 95-100

Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, pp 257-271, November 1999, ISSN 1089-778X

18

Evolutionary Algorithms in Decomposition-Based Logic Synthesis

Mariusz Rawski
Warsaw University of Technology
Poland

1. Introduction

Functional decomposition is a logic synthesis method that has recently gained much recognition. The main reason is the evolution of field programmable gate-arrays (FPGAs) as a new technology for digital system implementation. Architecture of FPGA is based on the lookup table (LUT) as basic building block. An *n*-input LUT is capable of implementing any Boolean function of up to *n* variables. Thus, logic synthesis for LUT-based FPGAs must transform a logic network into network that consists of nodes with up to *n* inputs only. Each node of such network can be then implemented by a single LUT. For this reason, for the case of implementation targeting FPGA structure, decomposition is a very efficient method. Modern FPGA devices have very complex structure. Today's FPGAs are entire programmable systems on a chip (SoC) which are able to cover an extremely wide range of applications. The Altera Stratix III and Xilinx Virtex-5 families of devices, both using a 65 nm manufacture process, can be used as examples of contemporary FPGAs. The basic architecture of FPGAs has not changed dramatically since their introduction in the 1980s. Early FPGAs used a logic cell consisting of a 4-input lookup table and register. Present devices employ larger numbers of inputs (6-input for Virtex-5 and 7-input for Stratix III) and have other associated circuitry. Another enhancement extensively used in modern FPGAs are specialized embedded blocks, serving to improve delay, power and area if utilized by the application, but waste area and power if unused. Early embedded blocks included fast carry chains, memories, phase locked loops, delay locked loops, boundary scan testing and multipliers. More recently, multipliers have been replaced by digital signal processing (DSP) blocks which add support for logical operations, shifting, addition, multiply-add, complex multiplication etc. Some architectures even contain hardware CPU cores. This greatly extends the space of possible solution during the process of mapping the design into FPGA structure with such embedded blocks. Unfortunately such heterogeneous structure of available logic resources greatly increases the complexity of mapping algorithms. The existing CAD tools are not well suited to utilize all possibilities that such modern programmable structures offer due to the lack of appropriate logic synthesis methods.
Functional decomposition is perceived as one of the best logic synthesis methods targeted FPGAs. It relies on breaking down a complex system into a network of smaller and relatively independent co-operating subsystems, in such a way that the original system's behavior is preserved. A system is decomposed into a set of smaller subsystems, such that each of them is easier to analyze, understand and synthesize. Decomposition allows

synthesizing the Boolean function into multilevel structure that is built of components, each of which is in the form of LUT logic block specified by truth tables.
Since the Ashenhurst-Curtis decomposition have been proposed, the research has been focused in forming new decomposition techniques (Łuba & Selvaraj, 1995; Sasao et al., 2001; Scholl, 2001; Brzozowski & Łuba, 2003; Rawski, 2007a). The researchers have developed many types of decompositions, but they are still based on Ashenhurst's ideas. Thanks to the fact that the functional decomposition gives very good results in the logic synthesis of combinational circuits, it is viewed for the most part, as a synthesis method for implementing combinational functions into FPGA-based architectures (Wurth et al, 1999; Scholl, 2001; Rawski et al., 2007). However, the decomposition-based method can be used beyond this field. Decomposition-like synthesis methods are not limited only to logic synthesis of digital circuits. The strong motivation for developing decomposition techniques comes recently from modern research areas such as pattern recognition, knowledge discovery and machine learning in artificial intelligence (Perkowski et al. 1997).
The practical usefulness of functional decomposition for very complex systems is limited by the lack of an efficient method for the construction of the high quality subsystems. In the subsystem construction process the following three factors play an extremely important role: an appropriate input support selection for subsystems, decision which (multi-valued) function will be computed by a certain subsystem and encoding of the subsystem's function with binary output variables. For large functions the solution space is so huge that heuristic method for solving this problem has to be used. This is an NP-hard problem and thus heuristic methods have to be used to efficiently and effectively search for optimal or near-optimal solutions.
There are two types of algorithms solving input variable partitioning problem. The algorithms finding decompositions without using any search heuristics. The basic idea of these algorithms is to limit the search to some input variable partitions. This is done by using different functional methods to choose which partitions will be evaluated. These methods select partitions through Reed-Muller expansions, Fourier transforms, binary difference equations, and technology-based mappings (Łuba et al., 1995; Perkowski, 1994; Steinbach & Stokert, 1994). The second type of algorithms utilize different heuristic methods. In (Rawski et al., 2001) input variable partitioning method based on information relationship measures was presented, which produced optimal or sub-optimal results for factions of considerable size.
In recent years the use of the genetic algorithms has received widespread attention. An evolutionary computing is inspired by Darwin's theory of evolution. In other words, problems are solved by an evolutionary process resulting in the best (fittest) solution (survivor) – the solution is evolved. 'Genetic algorithm' term was introduced by John Holland (Holland, 1975). The evolutionary algorithm is one of heuristics, which not necessarily provides the best possible solution. However, these sub-optimal solutions are considered as acceptable, because in many problems it is not possible to find the best solution in reasonable time. It means that evolutionary algorithms are especially useful for problems with a vast search space and non-polynomial time algorithms solving the given problem.
The evolutionary algorithms need individuals that represent a solution attempt to the problem they are trying to solve. The population needs to be tested to find how well individuals perform, and new individuals are created that are combinations of existing good solutions with some occasional variations. The cycle of testing and creation of new

individuals is repeated until a suitable solution is found, all the individuals represent the same solutions, or the search is abandoned.
This approach has been used to find approximate solutions to NP-complete optimization problems (Khuri, 1994). There have been attempts to apply genetic algorithms to functional decomposition (Noviskey et al., 1994). In (Rawski et al., 2004) the application of evolutionary algorithms was proposed to solve input support selection problem for functional decomposition based on blanket calculus. The solution has been extended to decomposition based on BDDs (Morawiecki & Rawski, 2008)
In this chapter an application of evolutionary algorithm for functional decomposition-based logic synthesis will be discussed. First an introduction to functional decomposition based on cubes and BDDs will be given. Next basics of evolutionary algorithms will be outlined. Subsequently the heuristic input partitioning method will be presented. Following that some experimental results will be discussed. The experimental results demonstrate that the proposed method is able to construct optimal or near optimal decompositions efficiently, even for large systems.

2. Basic information

In this section, only information that is necessary for an understanding of this chapter is reviewed. More detailed description of functional decomposition based on partition calculus can be found in (Brzozowski & Łuba, 2003), functional decomposition based on BDD in (Scholl, 2001).

2.1 Functional decomposition

The set X of input variables of Boolean function is partitioned into two subsets: *free variables* U and *bound variables* V, such that $U \cup V = X$. Assume that the input variables $x_1, ..., x_n$ have been relabeled in such a way, that:
$U = \{x_1, ..., x_r\}$ and
$V = \{x_{n-s+1}, ..., x_n\}$.
Consequently, for an n-tuple x, the first r components are denoted by x^U and the last s components are denoted by x^V.

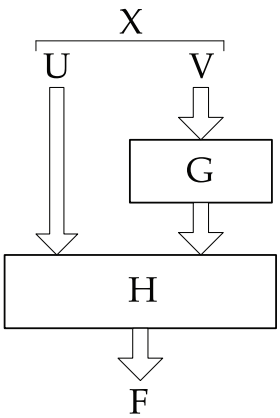


Fig. 1. Schematic representation of the functional decomposition.

Let F be a Boolean function with n inputs and m outputs and let (U, V) be the pair of sets defined above. Assume that F is specified by a set of the function's cubes. Let G be a function with s inputs and p outputs, and let H be a function with $r + p$ inputs and m outputs. The pair (G, H) represents a serial decomposition of F with respect to (U, V), if for every minterm b relevant to F, $G(b^V)$ is defined, $G(b^V) \in \{0, 1\}^p$, and $F(b) = H(b^U, G(b^V))$. G and H are called blocks of the decomposition (Fig. 1).

2.2 Functional decomposition based on blanket calculus

A Boolean function can be specified using the concept of cubes (input patterns) representing some specific sub-sets of minterms (Tab. 1.). In a minterm, each input variable position has a well-specified value. In a cube, positions of some input variables can remain unspecified and they represent "any value" or "don't care" (–). A **cube** may be interpreted as a p-dimensional subspace of the n-dimensional Boolean space or as a product of $n - p$ variables in Boolean algebra (p denotes the number of components that are '–'). For function from Table 1 truth table with $2^4 = 16$ rows would be required to describe the function using minterms. Since cube represents a set of minterms, application of cubes allows for much more compact description in comparison with minterm representation. For example cube 10–0 from row 2 of truth table from Table 1 represents set of two minterms {1000, 1010 }.

	x_1	x_2	x_3	x_4	y
1	0	0	–	0	1
2	1	0	–	0	1
3	–	0	0	–	1
4	–	–	1	1	0
5	–	1	1	0	0
6	1	1	–	1	0
7	0	–	0	1	1
8	–	1	0	0	0

Table 1. Example function.

For pairs of cubes and for a certain input subset B, we define the **compatibility relation** COM as follows: each two cubes S and T are compatible (i.e. $S, T \in \mathrm{COM}(B)$) if and only if $x(S) \sim x(T)$ for every $x \subseteq B$. The compatibility relation ~ on {0, –, 1) is defined as follows [1]: 0 ~ 0, – ~ –, 1 ~ 1, 0 ~ –, 1 ~ –, – ~ 0, – ~ 1, but the pairs (1, 0) and (0, 1) are not related by ~. The compatibility relation on cubes is reflexive and symmetric, but not necessarily transitive. In general, it generates a "partition" with non-disjoint blocks on the set of cubes representing a certain Boolean function F. The cubes contained in a block of the "partition" are all compatible with each other.

"Partitions" with non-disjoint blocks are referred to as blankets (Brzozowski & Łuba, 2003). The concept of blanket is a simple extension of ordinary partition and typical operations on blankets are strictly analogous to those used in the ordinary partition algebra.

Definition 1. Blanket

A **blanket** on a set S is such a collection of (not necessary disjoint) distinct subsets B_i of S, called blocks, that

$$\bigcup_i B_i = S \tag{1}$$

Each block B_i of blanket has its cube representative $r(B_i)$ that indicates the value of variables inducing blanket corresponding to this block.

Example 1 (Blanket-based representation of Boolean functions).

For function F from Table 1, the blankets induced by particular input and output variables and by the two-output function on the set of function F's input patterns (cubes) are as follows:

$$\beta_{x1} = \{\overline{1,\ 3,\ 4,\ 5,\ 7,\ 8};\ \overline{2,\ 3,\ 4,\ 5,\ 6,\ 8}\}, \tag{2}$$

$$\beta_{x2} = \{\overline{1, 2, 3, 4, 7};\ \overline{4, 5, 6, 7, 8}\}, \tag{3}$$

$$\beta_{x3} = \{\overline{1, 2, 3, 6, 7, 8};\ \overline{1, 2, 4, 5, 6}\}, \tag{4}$$

$$\beta_{x4} = \{\overline{1, 2, 3, 5, 8};\ \overline{3, 4, 6, 7}\}, \tag{5}$$

$$\beta_{y} = \{\overline{4, 5, 6, 8}\ ;\ \overline{1, 2, 3, 7}\}. \tag{6}$$

The product of two blankets β_1 and β_2 is defined as follows:

$$\beta_1 \bullet \beta_2 = \{\ B_i \cap B_j \mid B_i \in \beta_1 \text{ and } B_j \in \beta_2\ \}. \tag{7}$$

For two blankets we write $\beta_1 \le \beta_2$ if and only if for each B_i in β_1 there exists a B_j in β_2 such that $B_i \subseteq B_j$. The relation $\le$ is reflexive and transitive.

For example:

$$\beta_{x2x3} = \beta_{x2} \bullet \beta_{x3} = \{\overline{1, 2, 3, 7};\ \overline{1, 2, 4};\ \overline{6, 7, 8};\ \overline{4, 5, 6}\ \}, \tag{8}$$

$$\beta_{x2x3} \le \beta_{x2}. \tag{9}$$

Information on the input patterns of a certain function F is delivered by the function's inputs and used by its outputs with precision to the blocks of the input and output blankets. Knowing the block of a certain blanket, one is able to distinguish the elements of this block from all other elements, but is unable to distinguish between elements of the given block. In this way, information in various points and streams of discrete information systems can be modeled using blankets.

Theorem 1. Existence of the serial decomposition (Brzozowski & Łuba, 2003).

Let F be a Boolean function with n inputs and m outputs and let (U, V) be the pair of sets: *free variables* U and *bound variables* V, such that $U \cup V = X$. Let β_V, β_U, and β_F be blankets induced on the function F's input cubes by the input sub-sets V and U, and outputs of F, respectively.

If there exists a blanket β_G on the set of function F's input cubes such that $\beta_V \le \beta_G$, and $\beta_U \bullet \beta_G \le \beta_F$, then F has a serial decomposition with respect to (U, V).

Proof of Theorem 1 can be found in (Brzozowski & Łuba, 2003).

As follows from Theorem 1 the main task in constructing a serial decomposition of a function F with given sets U and V is to find a blanket β_G which satisfies the condition of the theorem. Since β_G must be $\ge \beta_V$, it is constructed by merging blocks of β_V as much as possible.

Two blocks B_i and B_j of blanket β_V are compatible (mergeable), if blanket γ_{ij} obtained from blanket β_V by merging B_i and B_j into a single block satisfies the second condition of Theorem 1, that is, if $\beta_U \bullet \gamma_{ij} \leq \beta_F$. Otherwise blocks B_i and B_j are incompatible (unmergeable). A subset δ of blocks of the blanket β_V is a compatible class of blocks if the blocks in δ are pair wise compatible. A compatible class is maximal if it is not contained in any other compatible class.

From the computational point of view, finding maximal compatible classes is equivalent to finding maximal cliques in a graph $\Gamma = (N, E)$, where the set N of vertices is the set of blocks of β_V and set E of edges is formed by set of compatible pairs.

The next step in the calculation of β_G is the selection of a set of maximal classes, with minimal cardinality, that covers all the blocks of β_V. The minimal cardinality ensures that the number of blocks of β_G, and hence the number of outputs of the function G, is as small as possible.

In certain heuristic strategies, both procedures (finding maximal compatible classes and then finding the minimal cover) can be reduced to the graph coloring problem.

Calculating β_G corresponds to finding the minimal number k of colors for graph $\Gamma = (N, E)$.

Example 2. For the function from Table 1 specified by a set F of cubes numbered 1 through 8, consider a serial decomposition with $U = \{x_1\}$ and $V = \{x_2, x_3, x_4\}$.

We find

$$\beta_U = \beta_{x1} = \{\overline{1,3,4,5,7,8}; \overline{2,3,4,5,6,8}\}, \tag{10}$$

$$\beta_V = \beta_{x2x3x4} = \{\overline{1,2,3}; \overline{3,7}; \overline{1,2}; \overline{4}; \overline{6,7}; \overline{5}; \overline{4,6}\}, \tag{11}$$

$$\beta_F = \beta_y = \{\overline{4, 5, 6, 8}; \overline{1, 2, 3, 7}\}. \tag{12}$$

Let β_G be as follows:

$$\beta_G = \{\overline{1,2,3,7}; \overline{4,5,6,8}; \overline{6,7}\}. \tag{13}$$

It is easily verified that β_G satisfies the condition of Theorem 1 (more detailed description of partition calculus can be found in (Brzozowski & Łuba, 2003)). Thus function F has a serial decomposition with respect to (U, V).

Number of blocks in blanket β_G determines the number of outputs of block G:

$$p = \lceil log_2(q) \rceil, \tag{14}$$

where q is the number of blocks in blanket β_G.

Since in example β_G has 3 blocks, to encode blocks of this blanket two encoding bits g_1 and g_2 have to be used. To define a function G by a set of cubes we calculate cube representatives, $r(B_i)$, assigned to each block B_i of β_V. The relationship between blocks of β_V and their cube representatives, $r(B_i)$, relies on containment of block B_i in blocks of β_{xj} from $x_j \in V$. Finally, the value of function G is obtained on the basis of containment of blocks B_i in blocks of β_G. To compute the cubes for function H we consider each block of the product $\beta_U \bullet \beta_G$. Their representatives are calculated in the same fashion. Finally, the outputs of H are calculated with respect to β_F

The process of functional decomposition based on blanket calculus consists of the following steps:

- the selection of an appropriate input support V for block G (input variable partitioning),
- the calculation of the blankets β_U, β_V and β_F,
- the construction of an appropriate multi-block blanket β_G (this corresponds to the construction of the multi-valued function of block G),
- the creation of the binary functions H and G by representing the multi-block blanket β_G as the product of a number of certain two-block blankets (this is equivalent to encoding the multi-valued function of block G defined by blanket β_G with a number of binary output variables).

2.3 Functional decomposition based on BDDs

A Boolean function can be represented using binary decision diagrams. BDDs as a method of representation of single-output Boolean functions were introduced by Lee (Lee, 1959) and later Ackers (Ackers, 1978).

Definition 2. Binary decision diagram (BDD)

Binary decision diagram is a rooted directed acyclic graph $\Gamma = (V, E)$ with node (vertex) set V and arc set E. The graph has terminal nodes called leaves. To each leaf node there is assigned a value 0 or 1. Each non terminal node $v \in V$ is labeled with a Boolean variable $var(v)$ and has arcs directed towards two children: $low(v) \in V$ corresponding to the case where the variable is assigned 0, and $high(v) \in V$ corresponding to the case where the variable is assigned 1.

When a Boolean function is represented by binary decision diagram with a given assignment to the variables, the value yielded by the function is determined by tracing a path from the root to a terminal vertex, following the branches indicated by the values assigned to the variables. The function value is then given by the terminal vertex label.

Definition 3. Ordered binary decision diagram (OBDD)

An ordered binary decision diagram is a BDD where an ordering $<$ over set of variables is defined, and for any node v and either nonterminal child u, their respective variables must be ordered $var(v) < var(u)$.

In (Bryant, 1986) Bryant presented algorithms that efficiently manipulated BDDs assuming ordering of the variables. He developed a method to reduce the size of BDDs by removing 'redundant' nodes and subgraphs which occur more than once. Bryant also proved that the reduced representation is canonical in respect to a given variable ordering.

Definition 4: **Reduced Ordered Binary Decision Diagram (ROBDD)** is an OBDD, that has no vertex v such that $low(v) = high(v)$ and for no pair $\{u, v\}$ sub-graphs rooted in v and u are isomorphic.

Binary decision diagrams made it possible to develop new algorithms for decomposition, feasible for much larger functions than previously possible. In a BDD, the decomposition can be easily computed by moving the bound variables V to the upper part of the graph and counting the number of children below the boundary line, usually called cut line.

Definition 5. Cut-set

Let Γ be the ROBDD representing a function F with variable ordering O, let $cut\_set(\Gamma, O, l)$ denote the set of nodes whose levels are greater than l and have edges from nodes of level lower or equal to l (top node has level 1).

Theorem 2. Existence of the serial decomposition.
Let F be Boolean function and (U, V) be the pair of sets: *free variables* and *bound variables*. Let Γ be ROBDD representing function F with variable ordering such that bound variables are in upper part of Γ. Let

$$p=\lceil log_2(|\, cut\_set(\Gamma,O,\, l)\,|)\rceil \tag{15}$$

If $p < l$, there exists decomposition in the form $F(X) = H(U, G(V))$, where function G has p outputs.
The size of *cut_set* from (15) plays the same role in BDD-based function decomposition as number of blocks in blanket β_G from (14).
Detailed description of functional decomposition based on BDD can be found in (Scholl, 2001).
Decomposition algorithms following a BDD-cut strategy proved to be orders of magnitude faster than those based on decomposition charts and cube representations. However, they require a reordering of the BDD to move the target set of variables to the top of the graph.

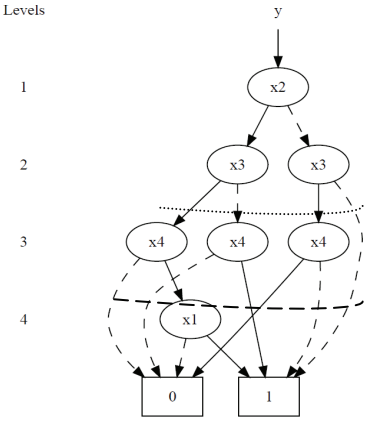


Fig. 2. ROBDD for function from Table 1.

Example 3. The ROBDD diagram Γ presented on Fig. 2 represents function F from Table 1 for ordering O={ x_2, x_3, x_4, x_1}. Let consider two cut-lines: at level 2 (dotted line) and at level 3 (dashed line). We have:

$$q_1=|\, cut\_set(\Gamma,O,\, 2)\,| = 4, \tag{16}$$

$$q_2=|\, cut\_set(\Gamma,O,\, 3)\,| = 3. \tag{17}$$

Following (15):

$$p_1=\lceil log_2(q_1)\rceil=2, \tag{18}$$

$$P_2=\lceil log_2(q_2)\rceil=2. \tag{19}$$

According to Theorem 2 decomposition with U = {x_4, x_1} and V = {x_2, x_3} does not exist since p_1 does not satisfy condition $p_1 < l$, where l = 2. Block G would require 2 outputs, while having 2 inputs.
However there exists decomposition with U = {x_1} and V = {x_2, x_3, x_4}, since p_1= 2 and l = 3. The size of block G will be 3 inputs and 2 outputs.

2.4 Evolutionary algorithms

An evolutionary computing is inspired by Darwin's theory of evolution. In other words, problems are solved by an evolutionary process resulting in the best (fittest) solution (survivor) – the solution is evolved. 'Genetic algorithm' term was introduced by John Holland (Holland, 1975). Here an evolutionary algorithm is used, which is more general term.

The evolutionary algorithm is the heuristics, which not necessarily provides the best possible solution. However, these sub-optimal solutions are considered as acceptable, because in many problems it is not possible to find the best solution in reasonable time. It means that evolutionary algorithms are especially useful for problems with a vast search space and non-polynomial time algorithms solving the given problem.

The evolutionary algorithms need individuals that represent a solution attempt to the problem they are trying to solve. The construction of an algorithm starts with mapping a problem into a set of chromosome representations. The population needs to be tested to find how well individuals perform, and new individuals are created that are combinations of existing good solutions with some occasional variations. The cycle of testing and creation of new individuals is repeated until a suitable solution is found, all the individuals represent the same solutions, or the search is abandoned. The basic steps of an evolutionary algorithm are presented on Fig. 3.

To construct the algorithm following qualities have to be defined:

- a population of individuals, where each individual represents an encoded form of a possible solution to the problem being solved,
- methods for testing individual solutions and assigning fitness (how good the solution is),
- methods for selecting suitable parents that will be used to produce new individuals (offspring),
- methods for manipulating the encoded forms of individuals, often called "genetic operators"; these operators are used to create new children from parents (for example, "crossover" techniques), and for introducing other variations (such as "mutation") into the population,
- parameters to manipulate the probability and effect of operators.

```
Evolutionary algorithm()
begin
    t := 0
    P^0 := create_initial_population()
    evaluate_fitnes(P^0)
    while (no_improvement_iterations > threshold) do
    begin
        T^t := selection_operator (P^t)
        O^t := crossover_operator (T^t)
        evaluate_fitnes(O^t)
        if (mutation_condition) then
            O^t := mutation_operator (O^t)
        P^(t+1) := O^t
        t := t +1
    end
end
```

Fig. 3. Outline of an evolutionary algorithm.

3. Evolutionary algorithm for input variable partitioning

The practical usefulness of functional decomposition for very complex systems is limited by lack of an efficient method for the construction of the high quality subsystems (*G* function from Fig. 1). In the subsystem construction process the following three factors play an extremely important role: an appropriate input support selection for subsystems, decision which (multi-valued) function will be computed by a certain subsystem and encoding of the subsystem's function with binary output variables. For function *F* of *n* input variables and the size *k* of input set of subsystem the number of possible solution is described by formula (20).

$$l = \binom{n}{k} = \frac{n!}{(n-k)!\,k!} \tag{20}$$

For large functions the solution space is so huge that heuristic method for solving this problem has to be used.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}
$\lvert U\rvert = 10,\ \lvert V\rvert = 3,\ \lvert\beta_G\rvert = 5$												
1	**1**	**1**	1	1	1	**0**	1	**1**	1	1	**0**	**0**
1	**1**	**1**	1	1	1	**0**	1	**1**	1	0	**1**	**0**
1	**1**	**1**	1	1	1	**0**	1	**1**	1	0	**0**	**1**
1	**1**	**1**	1	1	0	**0**	1	**1**	1	1	**0**	**1**
1	**1**	**1**	1	0	1	**0**	1	**1**	1	1	**0**	**1**
1	**1**	**1**	1	0	0	**1**	1	**1**	1	1	**0**	**1**
1	**1**	**1**	0	1	1	**0**	1	**1**	1	1	**1**	**0**
0	**1**	**1**	1	1	1	**1**	1	**1**	1	1	**0**	**0**
0	**1**	**1**	1	1	1	**1**	1	**1**	0	1	**1**	**0**
0	**1**	**1**	1	1	1	**1**	1	**1**	0	1	**0**	**1**
0	**1**	**1**	1	1	1	**1**	0	**1**	1	1	**1**	**0**
0	**1**	**1**	1	1	1	**1**	0	**1**	1	1	**0**	**1**
0	**1**	**1**	1	1	1	**1**	0	**1**	0	1	**1**	**1**
0	**1**	**1**	1	1	1	**0**	1	**1**	1	1	**1**	**0**
0	**1**	**1**	1	1	1	**0**	1	**1**	1	1	**0**	**1**
0	**1**	**1**	1	1	0	**1**	1	**1**	1	1	**0**	**1**
0	**1**	**1**	1	0	1	**1**	1	**1**	1	1	**0**	**1**
0	**1**	**1**	0	1	1	**1**	1	**1**	1	1	**1**	**0**
$\lvert U\rvert = 9,\ \lvert V\rvert = 4,\ \lvert\beta_G\rvert = 7$												
0	**1**	**1**	1	1	1	**0**	1	**1**	1	1	**0**	**0**
$\lvert U\rvert = 8,\ \lvert V\rvert = 5,\ \lvert\beta_G\rvert = 11$												
0	**1**	**1**	1	1	1	**0**	1	**1**	1	0	**0**	**0**
0	**1**	**1**	1	1	1	**0**	1	**1**	0	1	**0**	**0**
0	**1**	**1**	1	1	1	**0**	0	**1**	1	1	**0**	**0**
$\lvert U\rvert = 7,\ \lvert V\rvert = 6,\ \lvert\beta_G\rvert = 17$												
0	**1**	**1**	0	1	1	**0**	1	**1**	0	1	**0**	**0**
Frequency of appearance in *V* set.												
16	**0**	**0**	3	3	3	**13**	4	**0**	5	3	**16**	**13**

Table 2. Best input variable partitioning problem solutions of *plan* example.

The analysis of best possible solutions for given Boolean function results in interesting observations (Rawski, 2007b). Table 2 presents the best solutions of input variable partitioning for *plan* example Boolean function from standard Microelectronics Center of North Carolina benchmark set (Yang, 1991). This function has 13 inputs and 25 outputs.
Each row of Table 2 describes one partitioning of input variable set $X = \{x_1, \ldots, x_{13}\}$ into variables belonging to set U (marked by digit '1') and belonging to set V that leads to optimal decomposition (according to the number of blanket β_G's blocks). It presents the best solutions for different sizes of sets V and U, as well as the frequency of appearance of given input variable in V set. It can be easily noticed that certain variables appear in bound set often than others. For example variable x_1 appears in V set for 16 solutions listed in Table 2, while x_2 does not belong to V set for any of the best solutions. This suggests that some variables are more predestined to be included in V and other to be included in U set when constructing good input variable partitions.
There is another interesting observation that can be made analyzing this example. Let us assume that the size of V set is 4. Now, let us create an input variable partitioning in such way that V set consists of variables that according to Table 2 are least appropriate to be in bund set: $V = \{x_2, x_3, x_4, x_9\}$ and $U = \{x_1, x_5, x_6, x_7, x_8, x_{10}, x_{11}, x_{12}, x_{13}\}$. As we could expect, the quality of decomposition (according to the number of blanket β_G's blocks) is 16 – the worst possible for this size of V set. However let us move "good" variable x_1 from set U to set V and "bad" variable x_2 from set V to set U. The quality of decomposition is now 15, so it has improved. If we now swap variables x_3 and x_{12}, the decomposition will have quality 11, so further improvement has been obtained.
Let us assume that we have two variable partitioning solution (V_1, U_1) and (V_2, U_2). We can create another solution by taking part of variables from V_1 and part from V_2 and construct V_3 (similarly for U_3). Taking observation described above into account we can suspect that after such variable exchange it is probable that "good" variables from V_1 and V_2 will be included in V_3. This should improve the quality of new solution in comparison to solution used as "parents". If we preserve improved solutions and eliminate worsen solution we can apply this approach again. Such behavior is characteristic for evolutionary algorithms. This means that evolutionary algorithm may be an efficient way for solving input variable partitioning problem.
In (Rawski et al., 2004) the evolutionary algorithm has been proposed that solves input variable partitioning problem for functional decomposition. The evolutionary algorithm maintains a population of individuals (chromosomes), that represent potential solutions of a given optimization problem (Fig. 3). A survival of the fittest individuals is implemented by the selection mechanism. For the next population, as potential solutions, such single organisms are chosen, which adaptation to the environment is the best. The adaptation (quality) of a specific chromosome is evaluated by a fitness function. The chromosomes are evolving through the process of selection, recombination (crossover) and mutation. After a given number of algorithm loops (generations), it is expected that the algorithm has found a satisfactory solution. Details of the evolutionary algorithm solving input variable partitioning problem are discussed below.

3.1 Chromosome encoding

The single chromosome (organism) represents one, possible solution of the input variable partitioning problem. In the method presented in this paper chromosomes are encoded by the integer numbers, each of which represents the number of the input variable assigned to the set V (bound variables) of the decomposition.

Example 4.
For 4-input function F from Table 1 a possible solution of the variable partitioning problem can be represented by the set $U = \{x_1\}$ and set $V = \{x_2, x_3, x_3\}$.
The corresponding chromosome encoding is {2 3 4}.

3.2 Fitness function

In (Rawski et al., 1999) has been shown that there is a strong correlation of number of values in the sub-functions of the serial functional decomposition (represented by the number of blocks in β_G or size of *cut_set*) with the decomposition's quality. However this number strongly depends on the input variable partitioning chosen for the decomposition process. Therefore, the number of blocks in the β_G blanket or size of *cut_set* can be used as a good quality measure of the input variable partitioning. In the presented method the fitness function depends on this number – the less the number the better fitness of a given chromosome.
For the chromosome from Example 4 the number of blocks in a blanket β_G is $k = 3$ (Example 2).

3.3 Initial population selection

The initial population P^0 is created randomly. Once it is completed, the algorithm checks whether all the inputs (single genes) have been chosen at least once. If some are missing, the additional organism is created with genes which are not included in other organisms of the population.

3.4 Selection method

The selection method is combination of tournament selection and elitism. Tournament selection chooses randomly two organisms from the population P^t, compares them and takes the better one to the T^t population. The number of times such a tournament has to be done to complete whole T^t population depends on the population size. Elitism guarantees that the best organism from P^t is taken to T^t population regardless it was taking part at any tournaments or not.

3.5 Crossover (recombination)

Crossover operator chooses randomly two organism (called 'parents') and crosses their genetic material (Fig. 4). The crossover probability parameter specifies how often the crossover operator is performed. In proposed method this parameter is set to 0.9. The algorithm checks whether parents have the same genes or not. If so, the crossover operator is not launched and the other potential parents are chosen. If crossover is performed, two new organisms are created (and taken to O^t population). Otherwise parents are taken to O^t population.

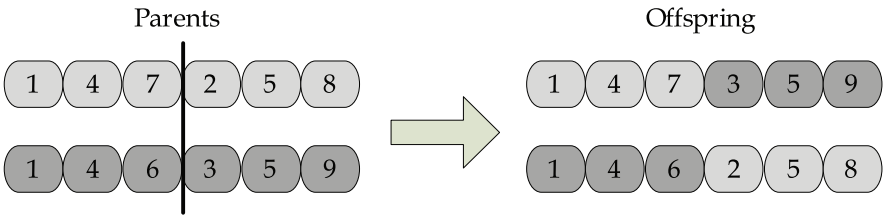


Fig. 4. Schematic representation of the crossover operator.

3.6 Mutation

Usually, mutation changes a single gene with very small probability (0.001). However, as experiments proved, in the case of the variable partitioning problem this kind of mutation does not bring any considerable profit for the algorithm performance.

The main problem with the presented algorithm is that it converges very fast to the local optimum. Once the algorithm gets to this area, it is very unlikely to find the better solution than this local optimum. To solve this problem, the special kind of mutation was implemented. If the average fitness among the population is very close to the best organism fitness, it is very likely the algorithm got stuck in the local optimum area. Then the special mutation is performed. One gene in each organism is mutated so the mutation probability is very high. As a result, the average fitness degenerates rapidly, but the algorithm gets out of the local optimum area and in many cases the better solution is found.

4. Input variable partitioning algorithm for heterogeneous LUTs

The methods presented in (Rawski et al., 2004), as well as in (Morawiecki & Rawski, 2008) were designed to solve the problem of input variable partitioning for given size of bound variable set V. In practice, during decomposition process there is a need to check existence of functional decomposition for several different sizes of V set before selecting the appropriate one. This is the case of applying functional decomposition in logic synthesis for FPGA architectures with heterogeneous logic resources. Such architectures are composed of adaptive logic elements that can be configured as LUTs of different sizes. In such situation application of concept presented in (Rawski et al., 2004) or (Morawiecki & Rawski, 2008) comes down to executing the algorithm for every possible LUT size.

However, the careful analysis of best possible solutions for *plan* example presented in Tab. 2 yields in interesting observation. Input variables that are present in bound set of best decompositions for set V of size k are often present in bound set of best decompositions for set V of size k – 1.

For example there is only one best solution for decomposition with set V of size 6 where $V = \{x_1, x_4, x_7, x_{10}, x_{12}, x_{13}\}$ and $U = \{x_2, x_3, x_5, x_6, x_8, x_9, x_{11}\}$. If we remove variable x_4 from set V and move it to set U we will obtain input variable partitioning that is one of 3 best solutions for decomposition with set V of size 5.

```
extended_evo_ivp( F , VSizeMin , VSizeMax )
begin
    V_VSizeMax := evo_ivp( F, VSizeMax )
    for k from VSizeMax– 1 downto VSizeMin do
    begin
        for i from 1 to k +1 do
        begin
            V_tmp := V_VSizeMax – { v_i }
            if (quality( V_tmp ) > best_quality) then
                V_best := V_tmp
                best_quality := quality( V_tmp )
        end
        V_k := V_best
    end
end
```

Fig. 5. Outline of algorithm that solves the problem of input variable partitioning for decomposition with V set of different sizes.

This can be used to construct the algorithm that applies evolutionary concept to find solution for largest size of set V only, and uses the found solution to construct solutions for decomposition with smaller set V. Let assume that we have evolutionary algorithm *evo_ivp*(F, *VSize*) that solves the problem of input variable partitioning for decomposition of function F with the bound set V of size *VSize*. To find good quality functional decompositions for V set of size from *VSizeMin* to *VSizeMax* the extended algorithm first will find solution (V set) for *VSizeMax* and then, by removing appropriate variables from found V set, it will construct solutions for decompositions with smaller sets V. The outline of the algorithm is presented on Fig. 5. The algorithm returns a list of V sets (V_k, where k = {*VSizeMin*, …, *VSizeMax*})

5. Results

The efficiency of evolutionary algorithm solving input variable partitioning problem for functional decomposition has been verified in (Rawski et al., 2004). This method was applied for number of combinational functions from MCNC logic synthesis benchmark set (Yang, 1991) and results ware compared with those obtained with the method based o information relationship measures presented in (Rawski et al., 2001) and with the systematic method. The systematic method is based on searching through the whole solution space and choosing an input support that produces blanket β_G with minimum possible number of blocks. All experiments were preformed on the computer with 512 Mbytes of RAM and AMD Athlon XP 3200.

Table 3 shows the comparison of the number of blanket β_G blocks for all methods for examples from MCNC logic synthesis benchmark set converted to truth table format (required by method from (Rawski et al., 2001). The results were obtained for decompositions with 3, 4, 5, and 6 input variables in set V. For these experiments the number of generations in the method based on the evolutionary algorithm was set to 30 and the size of a population was set to 40. Results obtained by the decomposition with the systematic search are optimal in the sense of the number of blocks of β_G. The method based on the evolutionary algorithm despite of its heuristic character produces results similar to the systematic method.

Table 4 present the minimum number of blocks of blanket β_G obtained by method based on the evolutionary algorithm for few large examples. The comparison with other two methods was impossible due to unacceptably long computation time for systematic method and due to the fact that method from (Rawski, 2001) accepts only truth table format. However examples from MCNC benchmark set are presented in espresso format, which in most cases is not truth table format and it is very difficult to convert such description for large multi-output systems into truth tables.

In Table 5 the comparison of execution time of the systematic method and the evolutionary-based algorithm is presented for different sizes of set V. As can be noticed, for large Boolean functions, method based on evolutionary algorithm is many times faster than the exact method. The difference in processing time between these two methods grows very fast with the function size. For the largest functions tried, the heuristic method was many thousands times faster.

	Size			Systematic method				Heuristic based on information relationship measures				Heuristic based on evolutionary algorithm			
	inputs	outputs	terms	3	4	5	6	3	4	5	6	3	4	5	6
Con1	7	2	20	5	6	6	5	5	7	7	5	5	6	6	5
Donfl	7	6	64	8	14	25	37	8	14	25	37	8	14	25	37
z4	7	4	128	4	6	8	12	4	6	8	12	4	6	8	12
Misex1	8	7	18	4	6	7	9	4	6	7	9	4	6	7	9
Root	8	5	71	5	9	15	17	5	9	15	17	5	9	15	17
Sqrt	8	4	53	3	4	7	12	3	4	7	12	3	4	7	12
Opus	9	10	23	4	6	8	10	4	6	8	10	4	6	8	10
9sym	9	1	191	4	5	6	7	4	5	6	7	4	5	6	7
Clip	9	5	430	6	10	14	18	6	10	14	21	6	10	14	18
Mark1	9	20	27	4	6	8	10	4	6	8	10	4	6	8	10
Alu2	10	3	391	6	12	24	43	6	12	24	43	6	12	24	43
Sao2	10	4	60	4	6	9	11	4	6	9	11	4	6	9	11
Cse	11	11	86	3	4	6	9	3	4	6	9	3	4	6	9
Sse	11	11	39	4	6	8	11	4	6	8	11	4	6	8	11
Keyb	12	7	147	6	9	13	19	6	9	13	19	6	9	13	19
S1	13	11	110	5	8	13	19	6	8	13	19	5	8	13	19
Plan	13	25	115	5	7	11	17	5	7	11	18	5	7	11	17
Styr	14	15	140	4	6	9	13	5	7	10	14	4	6	9	13
Ex1	14	24	127	4	6	8	11	4	6	8	11	4	6	8	11
Kirk	16	10	304	4	4	5	6	4	5	5	7	4	4	5	6
Duke2_7	18	1	64	3	4	4	4	4	5	5	5	3	4	4	4
Vg2_2	25	1	56	3	3	3	3	4	4	4	4	3	3	3	3
Apex3_3	34	1	208	2	3	4	5	4	4	4	6	2	3	4	5
Seq_2	36	1	211	2	2	3	–*)	3	4	4	6	2	2	3	5
Seq_1	37	1	286	2	3	3	3	3	3	3	4	2	3	3	3
Apex3_7	39	1	227	3	4	4	5	4	5	6	7	3	4	4	5

Table 3. Comparison of the number of blocks in blanket β_G obtained by the systematic method, heuristic method based on information relationship measures and heuristic method based on evolutionary algorithm for different size of set V. *) – too long computation time.

	Size			Heuristic based on evolutionary algorithm			
	inputs	outputs	terms	3	4	5	6
duke2	22	29	405	4	5	7	8
misex2	25	18	102	2	2	2	2
seq	41	35	3137	4	5	5	5
apex1	45	45	1440	4	5	6	7
apex3	54	50	1036	4	5	7	8
e64	65	65	327	4	5	5	7
apex5	117	88	2849	1	3	4	3

Table 4. The number of blocks in blanket β_G obtained by heuristic method based on evolutionary algorithm for different size of set V.

	Systematic method *				Heuristic based on evolutionary algorithm [s]			
	3	4	5	6	3	4	5	6
duke2	31s	2m 40s	10m 58s	34m 56s	37,7	38,9	40,4	58,5
misex2	10s	40s	4m 4s	13m 52s	10,2	12,4	14,9	24,9
seq	> 2 h	> 1 day	> 8 days	> 59 days	1656,8	1692,9	1704	1712,9
apex1	> 1 hour	> 12 hours	> 4 days	> 39 days	463,3	506,3	506	524,6
apex3	> 1 hour	> 16 hours	> 8 days	> 81 days	324	316,5	325	328,7
e64	31m 53s	> 8 hours	> 5 days	> 68 days	136,5	137,1	79	191,2
apex5	> 6 days	> 185 days	> 11 years	> 221 years	2849	3772,6	3799,5	3901,8

Table 5. Comparison of computation time of systematic method and heuristic method based on evolutionary algorithm for different sizes of set V.

In (Morawiecki & Rawski, 2008) the input selection method based on evolutionary algorithm was applied for decomposition based on BDDs. For manipulation on decision diagrams CUDD package was used. All the experiments were performed on the computer with 512 Mbytes of RAM and Pentium4 @ 2.8GHz.

	inputs	terms	Systematic method	Heuristic based on evolutionary algorithm
duke2_7	18	64	4	5
vg2_2	25	56	3	3
seq_2	36	211	2	4
apex1_16	38	179	2	3
apex1_23	39	2216	2	3

Table 6. The sizes of *cut_set* obtained by the systematic method and heuristic method based on evolutionary algorithm for size of set V equal to 4.

Table 6 presents the comparison of results obtained for single-output functions by applying the exhaustive search and evolutionary algorithm. The size of V set was 4 (typical value for FPGA-based synthesis). The size of *cut_set* is used as a quality measure. Results provided by the exhaustive search method can be considered as optimal. The results obtained by applying the evolutionary algorithm are very close to optimal or even optimal.

	Systematic method	Heuristic based on evolutionary algorithm [s]
duke2_7	80 sec	70
vg2_2	6,5 min	90
seq_2	31,5 min	75
apex1_16	46 min	120
apex1_23	49 min	130

Table 7. Comparison of computation time of systematic method and heuristic method based on evolutionary algorithm for sizes of set *V* equal to 4.

The comparison of execution time of the exhaustive search method and evolutionary algorithm has been presented in Table 7. It can be noticed that advantage of heuristic algorithm over exhaustive search grows fast with the size of decomposed Boolean function.

	Evolutionary algorithm				Extended evolutionary algorithm			
	3	4	5	6	3	4	5	6
duke2_7	4	4	5	5	4	4	4	5
vg2_2	3	3	3	3	3	3	3	4
seq_2	4	3	4	6	2	3	4	5
apex1_16	3	5	3	3	2	2	2	2
apex1_23	3	4	4	3	2	2	3	4

Table 8. Comparison of results of heuristic method based on evolutionary algorithm and its improved version.

Table 8 presents the comparison of results obtained with evolutionary algorithm and extended algorithm (Fig. 5) in case when existence of functional decomposition for several different sizes of *V* set has to be checked. It can be noticed that both methods provide results of comparable quality. However improved algorithm does it faster (Table 9). All the experiments were performed on the computer with 6 Gbytes of RAM and Intel Q9550 @ 2.83 GHz.

It has to be stressed that the multilevel decomposition consists of many single serial decomposition steps. Thus, application of the heuristic methods can speed up the multi-level decomposition process dramatically.

	Evolutionary algorithm [s]					Extended evolutionary algorithm [s]
	3	4	5	6	Total	Total
duke2_7	17.5	16.6	16.7	15.6	66.6	16.0
vg2_2	21.5	20.9	20.2	20.2	83.0	20.5
seq_2	15.9	16.0	15.1	16.1	63.3	16.6
apex1_16	17.1	16.6	15.7	14.6	64.2	16.0
apex1_23	17.1	15.2	15.1	15.4	63.0	15.5

Table 9. Comparison of computation time of heuristic method based on evolutionary algorithm and its improved version.

6. Conclusion

The heuristic method of variable partitioning based on the evolutionary algorithm turns out to be very efficient when applied for decomposition method based on cubes (Rawski, 2007a), as well on ROBDDs (Morawiecki & Rawski, 2008). The method delivers results of similar or comparable quality to results obtained from the exhaustive search, but does it many times faster. The algorithm parameters (the number of generations and the size of population) can be used to control the trade-off between the search time and quality of solutions. These features make the proposed heuristic method very useful for decomposition-based synthesis of large systems.

7. Acknowledgments

This work was partly supported by the Ministry of Science and Higher Education of Poland – research grant no. N N516 418538 for 2010-2012.

8. References

Akers, S. B. (1978). Binary decision diagrams. *IEEE Transactions on Computers*, Vol. 27, No. 6, pp. 509–516.

Bryant, R. E. (1986). Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, Vol. 35, No. 6, pp. 677–691.

Brzozowski, J. A. & Łuba, T. (2003). Decomposition of Boolean Functions Specified by Cubes, *Journal of Multiple-Valued Logic and Soft Computing*. Vol. 9, pp. 377–417.

Holland, J.H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press.

Khuri, S. (1994). An Evolutionary Approach to Combinatorial Optimization Problems. *Proceedings of CSC*, pp. 66 – 73.

Lee, C. Y. (1959) Representation of switching circuits by binary-decision diagrams. *The Bell System Technical Journal*, pp. 985–999.

Łuba, T.; Selvaraj, H.; Nowicka, M. & Kraśniewski, A. (1995). Balanced multilevel decomposition and its applications in FPGA-based synthesis. G.Saucier, A.Mignotte (ed.), *Logic and Architecture Synthesis*, Chapman&Hall.

Łuba, T. & Selvaraj, H. (1995). A general approach to Boolean function decomposition and its applications in FPGA-based synthesis, *VLSI Design*, Vol. 3, No. 3-4, pp. 289–300.

Morawiecki, P. & Rawski M. (2008) Method of Input Variable Partitioning in Functional Decomposition Based on Evolutionary Algorithm and Binary Decision Diagrams, *Proc. of IEEE 2008 Conference Human Systems Interaction*, publication on CD,.

Noviskey, M.J.; Ross, T.D.; Gadd, D.A. & Axtell M. (1994). Application of Genetic Algorithms to Functional Decomposition in Pattern Theory. Report WL-TR-94-1015, Wright Laboratory, WL/AART-2. WPAFB, OH 4533-5543.

Perkowski, M. (1994) A Survey of Literature on Function Decomposition. *Final Report for Summer Faculty Research Program*. Wright Laboratory. Sponsored by Air Force Office of Scientific Research. Bolling Air Force Base. DC and Wright Laboratory.

Perkowski, M; et al. (1997). Decomposition of Multiple-Valued Relations. *International Symposium on Multiple-Valued Logic*, pp. 13-18.

Rawski, M.; Selvaraj, H & Morawiecki, P. (2004) Efficient Method of Input Variable Partitioning in Functional Decomposition Based on Evolutionary Algorithms, *Proc. Euromicro Symposium on Digital System Design, Architectures, Methods and Tools*, IEEE Computer Society, Selvaraj H. (Editor), pp. 136-143.

Rawski, M. (2007a). Decomposition of Boolean function sets, *Electronics and Telecommunications Quarterly*, Vol. 53, No. 3, pp. 231–249.

Rawski, M. (2007b). Efficient Variable Partitioning Method for Functional Decomposition, *Electronics and Telecommunications Quarterly*, Vol. 53, No. 1, pp. 63-81.

Rawski, M.; Jóźwiak, L. & Łuba, T. (2001). Efficient Input Support Selection for Sub-functions in Functional Decomposition Based on Information Relationship Measures. *Journal of Systems Architecture*, Vol. 47 , No. 2, p. 137-155.

Rawski, M.; Jóźwiak, L. & Łuba, T. (1999). The Influence of the Number of Values in Sub-Functions on the Effectiveness and Efficiency of Functional Decomposition. *Proceedings of the 25th EUROMICRO Conference*, IEEE Computer Society, pp. 86-93.

Rawski, M.; Tomaszewicz, P.; Selvaraj, H. & Łuba T. (2005). Efficient implementation of digital filters with use of advanced synthesis methods targeted FPGA architectures, *Proceedings of Euromicro Symposium on Digital System Design, Architectures, Methods and Tools*, IEEE Computer Society, Wolinski C. (Editor), pp. 460–466.

Sasao, T.; Matsuura, M.; Iguchi, Y. & Nagayama, S. (2001). Compact BDD Representations for Multiple-Output Functions and Their Application, *Proceedings of ISMVL,* pp. 207-212.

Scholl, C. (2001). *Functional Decomposition with Application to FPGA Synthesis.* Kluwer Academic Publisher.

Steinbach, B. & Stokert, M. (1994). Design of Fully Testable Circuits by Functional Decomposition & Implicit Test Pattern Generation. *Proceedings of IEEE VLSI Test Symposium*, pp. 22–27

Wurth, B.; Schlichtmann, U.; Eckl, K. & Antreich, K. (1999). Functional multiple-output decomposition with application to technology mapping for lookup table-based FPGAs. *ACM Trans. Design Autom. Electr. Syst.* Vol. 4, No. 3, pp. 313-350.

Yang, S. (1991). Logic Synthesis and Optimization Benchmarks, Version 3.0. *Tech. Report*, Microelectronics Center of North Carolina.

19

A Memory-Storable Quantum-Inspired Evolutionary Algorithm for Network Coding Resource Minimization

Yuefeng Ji and Huanlai Xing
Beijing University of Posts & Telecommunications, China

1. Introduction

Network coding technology is a new communication paradigm that is superior to traditional routing in many aspects, especially in terms of the ability of increasing multicast throughput (Ahlswede et al., 2000; Li et al., 2003). Traditional routing adopts *store-and-forward* data forwarding scheme at every intermediate node that simply replicates and forwards the incoming data to downstream nodes. However, the maximum throughput of a multicast scenario could not be often achieved under such a scheme (Ahlswede et al., 2000; Li et al., 2003). With *code-and-forward* data forwarding scheme at network layer, network coding allows arbitrary intermediate node to combine (or code) the data received from different incoming links and output the coded information if necessary, being able to obtain a multicast throughput that is maximized according to the MAX-FLOW MIN-CUT theorem (Li et al., 2003).

Fig. 1 shows why network coding performs better than traditional routing in terms of the maximum multicast throughput they achieve. Fig.1(*a*) shows a network with source s and sinks y, z. Each direct link has a capacity of 1 bit per time unit. Source s expects to send two bits, a and b, to y and z. According to the MAX-FLOW MIN-CUT theorem, the min cut C_{min} between s and $\{y, z\}$ is 2 bits per time unit, which means the maximum multicast throughput from s to y and z should be 2 bits per time unit. However, if traditional routing is adopted, the multicast throughput is 1.5 bits information per time unit since link $w \rightarrow x$ could only forward 1 bit, a or b, to x, and thus y and z can not simultaneously receive two bits, a and b, as indicated in Fig.1(*b*). In Fig.1(*c*), if the intermediate node w is allowed to combine the 2 bits, a and b, it receives from t and u respectively to 1 bit $a \oplus b$ (here, symbol $\oplus$ is Exclusive-OR operation) and output $a \oplus b$ to x, y and z are both able to obtain $\{a, a \oplus b\}$ and $\{b, a \oplus b\}$, which means two bits information is available at both y and z. Meanwhile, y and z can use $\{a, a \oplus b\}$ and $\{b, a \oplus b\}$ to decode b and a by calculate $a \oplus (a \oplus b)$ and $b \oplus (a \oplus b)$ respectively.

To the best of our knowledge, most of the network-coding-related research works suppose that coding operation should be implemented at all coding-possible intermediate nodes. However, to achieve a desired throughput, coding operation may only be necessarily performed at a subset of those nodes (Kim et al., 2006; 2007a; 2007b). In Fig.2, there are two network coding schemes that could both achieve the maximum multicast throughput. Network coding scheme A adopts all coding-possible nodes, m and n, as shown in Fig.2(*a*).

Nevertheless, the same throughput is also obtained by network coding scheme B where one of the two coding-possible nodes, *m*, is required to perform coding operation (see Fig.2(*b*)). Since coding operation consumes computational time and increases date processing complexity, it is of vital importance to minimize the amount of coding operations required. Unfortunately, such problem is NP-Hard (Kim et al., 2006; 2007a).

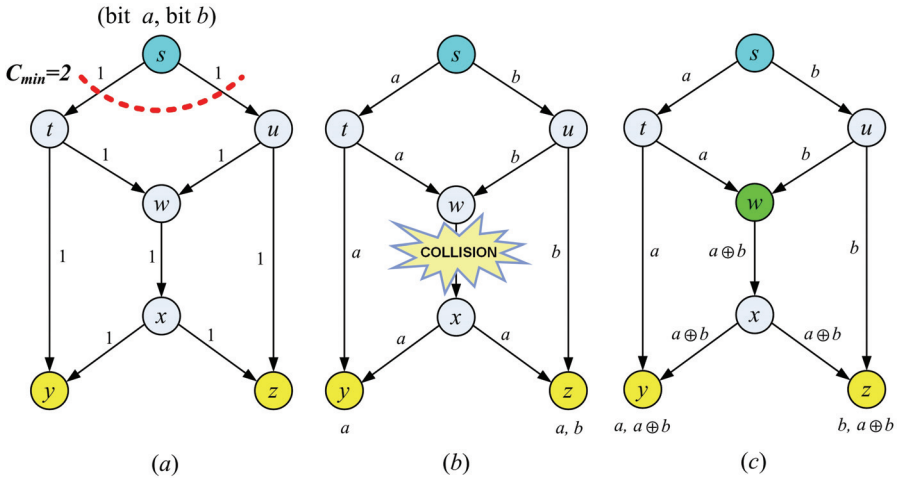


Fig. 1. Traditional routing vs. network coding. (*a*) A network topology with maximum multicast throughput of 2 bits per time unit. (*b*) Traditional routing scheme. (*c*) Network coding scheme.

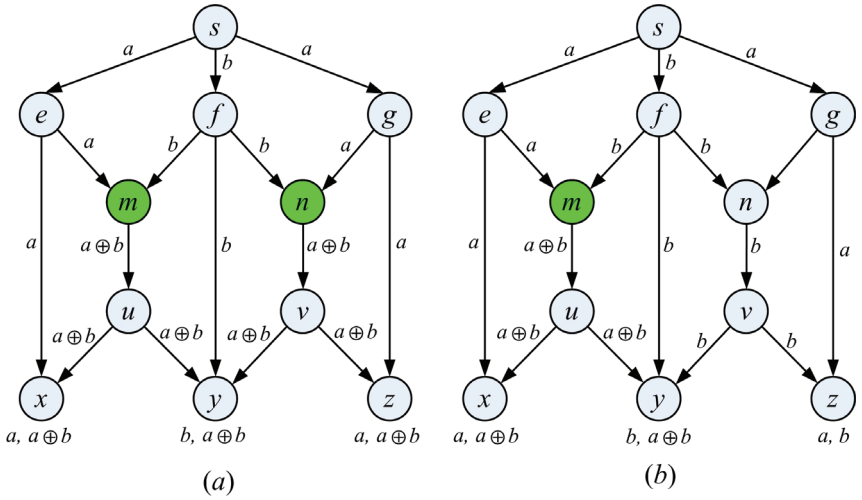


Fig. 2. Two different network coding schemes. (*a*) Network coding scheme A with two coding nodes. (*b*) Network coding scheme B with only one coding node.

In order to solve this problem, several algorithms have been proposed, which are mainly based on greedy algorithms or evolutionary approaches. In (Fragouli & Soljanin, 2006), a minimal subtree graph for a network coding multicast was created. The amount of coding operations totally depends on the link traversal order in a corresponding labeled line graph. Different link traversal order may result in different utilization of coding resource. Langberg et al. (2006) first transformed the given network to a network in which the degree of each node is up to 3. Then they checked on links one by one, and remove those which do not make contribution to the achievable rate. However, both of the above algorithms assume that the nodes with multiple incoming links must carry out network coding. Besides, their optimization performance depends on the link traversal order in the corresponding labeled line graph. In (Bhattad et al., 2005), linear programming formulations were proposed to optimize various network coding resources. Nevertheless, the number of variables and constraints grows with the number of sinks. Thus, this method limits itself to the case where the number of sinks is not large. Some genetic algorithms (GAs) with both centralized and distributed versions have been proposed to minimize the network coding operations required (Kim et al., 2006; 2007a; 2007b). The GA based algorithms seem to perform much better than the aforementioned minimal algorithms. However, due to the inherent shortcomings of GA such as pre-maturity, slow convergence speed, weak global searching capability, poor optimal performance is usually achieved.

Quantum-inspired evolutionary algorithm (QIEA), a combination of quantum computation and genetic algorithm, was formally introduced by Han and Kim (Han & Kim, 2002). Unlike other evolutionary algorithms, QIEA maintains a population of Q-bit representation based individuals, each of which represents a linear superposition of all states in search space probabilistically, and adopts various quantum gates, e.g. quantum rotation gate (Han & Kim, 2002) and quantum NOT gate (Xing et al., 2009a; 2009b), to drive the individuals toward the global optima. As one of estimation of distribution algorithms (EDAs), QIEA maintains and incrementally modifies multiple probabilistic models (Platel et al., 2009). QIEA is characterized by maintaining a diversified population due to the Q-bit representation, being able to explore the search space with a smaller number of individuals and exploiting the search space for a global solution within a short computational time (Han & Kim, 2004). However, in QIEA, global exploration and local exploitation can be provided simultaneously, only if proper evolution parameter values are set. Having a great effect on optimization performance of QIEA, how to set proper evolutionary parameters values for algorithms must be paid enough attention. However, in most of the existing QIEAs, the determination of evolutionary parameters does not take the differences among individuals into consideration. In (Han et al., 2001; Han & Kim, 2002; Li & Wang, 2007), fixed rotation angle step (FRAS) schemes were put forward. At arbitrary evolutionary generation, the algorithms use the same rotation angle step (RAS) strategy to evolve its population. If two individuals are under the same case according to the corresponding lookup table, they use the same RAS value to update. QIEA with FRAS scheme often results in slow convergence since the RAS values of lookup table never change. Later, the dynamic rotation angle step (called DRAS below) schemes were proposed (Zhang et al., 2003; Lv & Liu, 2007), where new RAS schemes were given at each generation. By using DRAS schemes, the searching grid of QIEA varies from large to small, and it is of some help to accelerate the convergence and achieve better solutions. However, at each generation, all the individuals under DRAS schemes only refer to one lookup table to update, which means DRAS schemes are also designed for a population but may be not suitable for every individual. Mutation is an

effective operation to prevent premature convergence and raise global search capability. Nevertheless, it was not adopted as a basic evolutionary operation in conventional QIEA (Han & Kim, 2002). Although, quantum mutation operations was introduced in (Yang et al., 2003), the differences among individuals were not considered so that the algorithm may sometimes be trapped in local search.

In order to provide an efficient network coding multicast scheme with less coding operations occupied and overcome the problems caused by the existing RAS schemes and conventional quantum mutation operation, this chapter presents a novel evolutionary algorithm called Memory-Storable Quantum-Inspired Evolutionary Algorithm (MS-QIEA). MS-QIEA is able to assign its individuals more suitable evolutionary parameter values according to their previous searching situations (such as, their fitness and evolutionary parameter values of the former generation). Each individual could easily evolve itself to a better searching position with respect to its former situation. Individuals, whose current searching situations are better than their previous searching situations, are allowed to have relatively large rotation angle step (RAS) values to accelerate the exploration speed and relatively small quantum mutation probability (QMP) values to survive, while those whose current searching situations are worse than their former searching situations are allocated with relatively small RAS values to avoid invalid evolution and relatively large QMP values to wait for a turn-to-excellent-individual opportunity. We evaluated the performance of MS-QIEA over a number of multicast scenarios. Simulation results showed that our algorithm performs better than some traditional evolutionary algorithms in terms of robustness, success ratio, convergence and global search capability.

2. Problem formulation

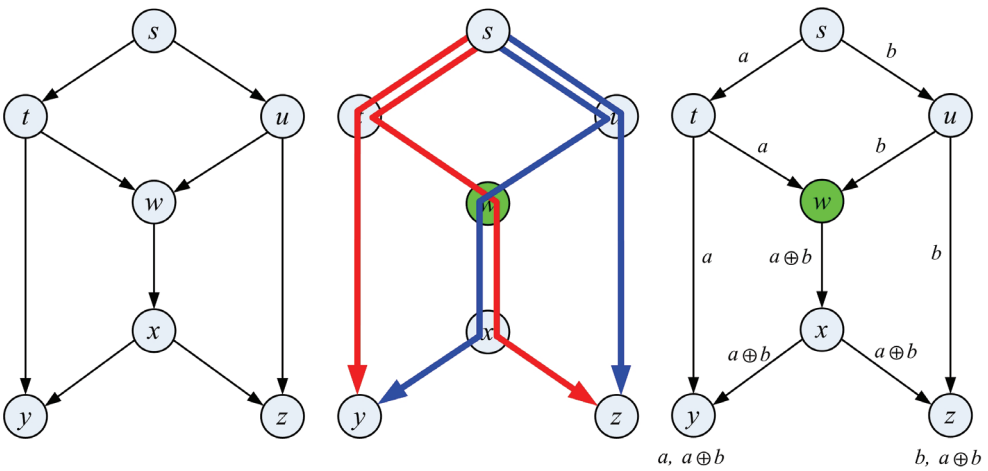
A communication network can be modeled as a directed graph $G = (V, E)$ where V is the set of nodes and E is the set of links (Li et al., 2003). Assume that each link $e \in E$ has a unit capacity. A single-source multicast scenario is considered as a 4-tuple (G, s, T, R) that includes a graph $G(V, E)$, a source node $s \in V$, a set of sinks $T = \{t_1, t_2, ..., t_d\} \subset V$, and a data rate R at which s wishes to transmit to all the sinks T. Rate R is said to be achievable only if there exists a transmission scheme that enables all $|T|$ sinks to receive all of the information sent at the same rate as R (Kim et al., 2007a). Since linear network coding is sufficient for multicast (Li et al., 2003), this chapter only considers linear coding, where the coded information is a linear combination of the information from its incoming links. Based on linear coding, a network coding based multicast subgraph (called NCM subgraph, denoted by $G_{NCM}(s,T)$) could be constructed if the subgraph guarantees to have R link-disjoint paths from s to each sink t_i, $i = 1, 2, \ldots, d$, respectively. A node is required to perform coding operation if there are at least two paths, $Path(s,t_m)$ and $Path(s,t_n)$, that are input to the node and have to flow out on the same outgoing link, where t_m and t_n are two sinks. Fig.3 shows how NCM subgraph is constructed and how it works. Fig.3(*a*) is a target network with source s and sinks y and z. All links has a capacity of one bit per time unit. Here, R is set to 2 bits per time unit. To generate a NCM subgraph, we need to find two link-disjoint paths between each source-sink pair, s-y and s-z. As indicated in Fig.3(*b*), there are two link-disjoint paths, a red path and a blue path, between s and y (or z). Meanwhile, node w needs to perform coding operation as there are two paths join together at w and both have link $w \rightarrow x$. Fig.3(*c*) shows the network coding scheme based on the constructed NCM subgraph. It is clear that w performs coding operation.

For a multicast scenario (G, s, T, R), it is expected to determine a minimal amount of coding operations to make the rate R achievable. The number of coding links is a precise estimator of the total amount of coding operations required (Langberg et al., 2006). Hence, our optimization objective hereafter is to minimize the number of coding links while achieving a desired throughput. It is easy to understand that different NCM subgraphs, with different amount of coding links but the same multicast throughput, may probably be constructed for a given multicast scenario (refer to Fig.2). Let $n_{cl}(G_{NCM}(s,T))$ be the number of coding links of the created NCM tree. Our goal is shown as follows:

$$\text{Min}\{n_{cl}(G_{NCM}(s,T))\} \tag{1}$$

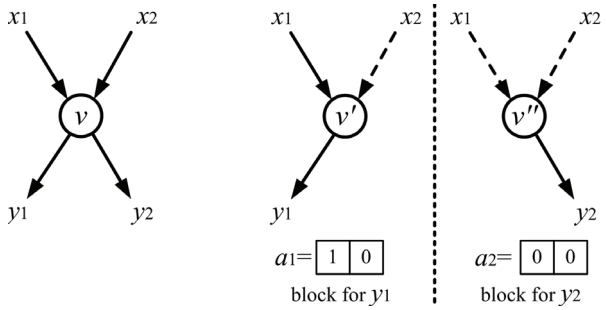
Note that no coding is necessary at a node with single incoming link, for such node has nothing to combine. According to (Kim et al., 2006; 2007a; 2007b), we refers to a node with multiple incoming links as a *merging node*. To determine whether coding is necessary on an outgoing link of a merging node, it is excessively imperative to verify whether the output depends on a single input without destroying the achievability of the given data rate. The necessity of coding at a link depends on which other links code and thus the problem of deciding where to perform network coding in general involves a selection out of exponentially many possible choices (Kim et al., 2006; 2007a).

Consider a merging node with $m \geq 2$ incoming links and $n \geq 1$ outgoing links. For each $i \in \{1, \ldots, m\}$ and each $j \in \{1, \ldots, n\}$, if the information from incoming link i contributes to the linearly coded output on outgoing link j, set a_{ij}=1, otherwise set a_{ij}=0. We refer to the '1' and '0' states as *active* and *inactive* states, respectively. Network coding is required over link j only if two or more link states are active at the same time. Thus, it is meaningful to consider $\boldsymbol{a}j = \{a_{ij} \mid i = 1, \ldots, m, j = 1, \ldots, n\}$ as a data block of length m (Kim et al., 2006; 2007a) (see Fig.4 for an example).



(*a*) Example network (*b*) Disjointed paths from s to y and z (*c*) Flows of data information

Fig. 3. An example of constructing a NCM subgraph and its data transmission scheme.



(a) Merging node v (b) Two blocks for outgoing links

Fig. 4. A possible input states of node v described by vectors $\boldsymbol{a}1$= (a_{11}, a_{21}) and $\boldsymbol{a}2$ = (a_{12}, a_{22}).

3. An overview of QIEA

QIEA is a probabilistic algorithm which exploits the power of quantum computation in order to accelerate genetic procedures (Han & Kim, 2002). The basic information unit is called quantum bit (Q-bit). A Q-bit is a two-level quantum system which may be in the |0> state, in the |1> state, or in any superposition of the two. The state of a Q-bit can be represented as

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2}$$

where $|\alpha|^2+|\beta|^2 = 1$, and α and β are complex numbers that specify the probability amplitudes of the corresponding states.
A Q-bit representation of m Q-bits individual is defined as:

$$\begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{pmatrix}, \tag{3}$$

where $|\alpha_i|^2+|\beta_i|^2 = 1$, $i = 1, 2,\ldots, m$. In this way, the m-Q-bit individual can simultaneously represent 2^m states. Thanks to the Q-bit representation, QIEA has a better characteristic of diversity than classical evolutionary approaches, since it can represent a linear superposition of many states. For example, whereas the 2-bit binary expression <0, 1> represents one state, a 2-bit Q-bit expression, e.g.

$$\begin{pmatrix} 1/\sqrt{2} & 1/2 \\ 1/\sqrt{2} & \sqrt{3}/2 \end{pmatrix}$$

represents four states: $\frac{1}{8}$<0,0> , $\frac{3}{8}$<0,1>, $\frac{1}{8}$<1,0>, and $\frac{3}{8}$<1,1>, where $\frac{1}{8}$, $\frac{3}{8}$, $\frac{1}{8}$, and $\frac{3}{8}$ are probabilities with which the corresponding states emerge.
For update, suitable quantum gate $U(\theta)$ is usually adopted in compliance with practical optimization problems. For the problem in this chapter, quantum rotation gate, such as

$$U(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, \tag{4}$$

where θ is rotation angle, is used as a basic gate of QIEA.
The procedure of QIEA is shown in Fig.5. More details are referred to (Han & Kim, 2002).

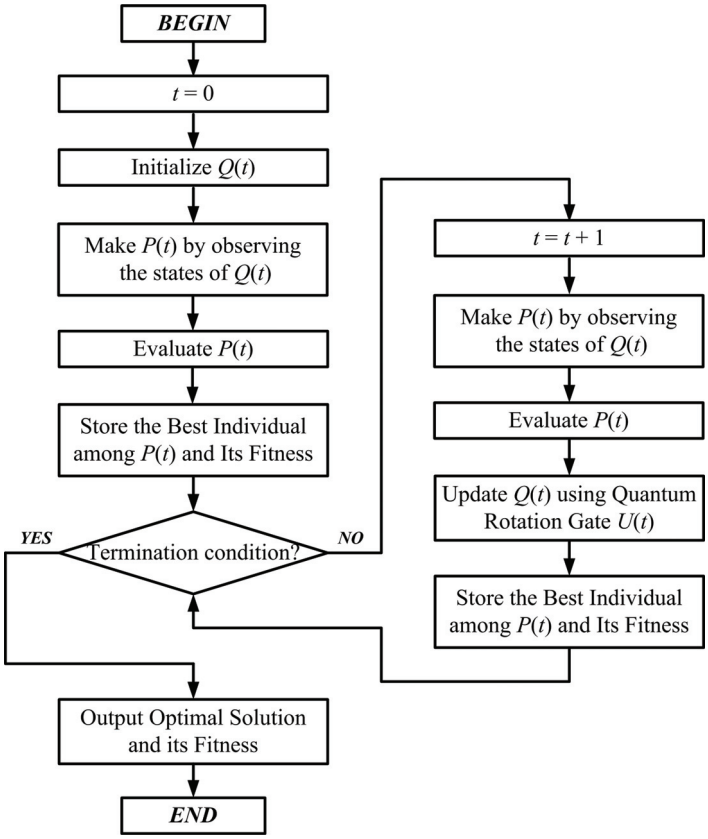


Fig. 5. Flow chart of QIEA

In Fig.5, $Q(t)=\{q_1^t, q_2^t, ..., q_N^t\}$ is a population of N Q-bit individuals at generation t, and q_j^t is the j-th (j = 1, 2, …, N) individual defined as

$$q_j^t = \begin{pmatrix} \alpha_{j1}^t & \alpha_{j2}^t & \cdots & \alpha_{jm}^t \\ \beta_{j1}^t & \beta_{j2}^t & \cdots & \beta_{jm}^t \end{pmatrix} \tag{5}$$

and $P(t) = \{X_1^t, X_2^t, ..., X_N^t\}$ is a set of binary solutions of observation states of $Q(t)$, where X_j^t is the binary solution obtained by observing q_j^t (j = 1, 2, …, N). In 'initialize $Q(t)$', each pair of Q-bit probability amplitudes, α_{ji}^t and β_{ji}^t, i = 1, 2, …, m, are initialized with $1/\sqrt{2}$, $\forall\, q_j^t \in Q(t)$. The step 'Make $P(t)$ by observing $Q(t)$' generates a set of binary solutions $P(t) = \{X_1^t, X_2^t, ..., X_N^t\}$, where each bit of X_j^t, j = 1, 2, …, N, is formed by determining each explicit state of the corresponding Q-bit of q_j^t, |0> state or |1> state, according to either $|\alpha_{ji}^t|^2$ or $|\beta_{ji}^t|^2$ of q_j^t, i = 1, 2, …, m. For example, to form a explicit state of the i-th bit of X_j^t (i = 1, 2, …, m), a

uniformly distributed variable *rand* between 0 and 1 is generated randomly. If *rand* < $|\beta_{ji}^t|^2$, the i-th bit of X_j^t is set to 1, otherwise, it is set to 0. Each solution $X_j^t \in P(t)$, $j = 1, 2, \ldots, N$, is a binary string of length m, and is evaluated according to the fitness function. The initial best solution X_{best}^t is then selected from the binary solutions $P(t)$ and stored. In the while loop, the quantum gate $U(\theta)$ is used to update $Q(t\text{-}1)$ so that fitter states of the Q-bit individuals are generated. The i-th Q-bit value $(\alpha_{ji}^t, \beta_{ji}^t)$ of q_j^t is updated as:

$$\begin{pmatrix} \alpha_{ji}^{t+1} \\ \beta_{ji}^{t+1} \end{pmatrix} = U(\theta_{ji}^t) \begin{pmatrix} \alpha_{ji}^t \\ \beta_{ji}^t \end{pmatrix} = \begin{pmatrix} \cos(\theta_{ji}^t) & -\sin(\theta_{ji}^t) \\ \sin(\theta_{ji}^t) & \cos(\theta_{ji}^t) \end{pmatrix} \begin{pmatrix} \alpha_{ji}^t \\ \beta_{ji}^t \end{pmatrix} \tag{6}$$

Here, θ_{ji}^t is the rotation angle to update the i-th Q-bit of q_j^t. The best solution among $P(t)$ is selected after 'Evaluate $P(t)$'. If the best solution of the current generation is fitter than the stored best solution X_{best}^t, the stored best solution X_{best}^t is replaced by this new solution.

4. The proposed MS-QIEA

This section provides an efficient coding resource optimization algorithm which is based on quantum-inspired evolutionary algorithm (QIEA) with Memory-Storable RAS (MS-RAS) scheme and Memory-Storable QMP (MS-QMP) scheme. At first, MS-RAS and MS-QMP schemes are introduced. Then, the individual representation and the fitness function are described. Eventually, we show the structure of the proposed algorithm MS-QIEA.

4.1 Memory-Storable RAS scheme

As RAS has a great influence on the convergence of QIEA, it is imperative to choose suitable RAS (Han & Kim, 2002). Individuals may fail to reach optimum solutions in few generations if RAS is excessively small, or they may miss optimum solutions if RAS is excessively large. However, most of the existing QIEA algorithms adopt either FRAS or DRAS as their RAS determination strategy, ignoring whether the RAS scheme is fit for all individuals. They, thus, may eventually result in bad optimization performance.

In order to improve the optimization efficiency, this chapter proposes an MS-RAS scheme, which adaptively allocates each individual a suitable RAS value at each generation. Under this scheme, every individual is treated separately according to its own situation. For arbitrary individual, if its fitness value at generation t is larger than that at generation t-1, the current RAS value is set to be smaller than its last RAS value; If its fitness value is smaller than that at last generation, the current RAS value is set to be larger than its last RAS value; Otherwise, the current RAS value is set to be equal to its last RAS value. Note that the problem concerned is a minimum problem where individuals with large fitness values are regarded to be inferior to those with small fitness values (refer to fitness definition in subsection 4.4).

A Q-bit individual q_j^t can be updated by rotation gate $U(\theta_{ji}^t)$, where θ_{ji}^t is the rotation angle for updating the i-th Q-bit of q_j^t, $i = 1, 2, \ldots, m$, (refer to section 3). Here, θ_{ji}^t is given as $S(\alpha_{ji}^t, \beta_{ji}^t) \cdot \Delta\theta_{ji}^t$, where $S(\alpha_{ji}^t, \beta_{ji}^t)$ and $\Delta\theta_{ji}^t$ are the sign and the RAS of θ_{ji}^t, respectively. The general updating scheme for the j-th individual at generation t is proposed, as shown in Table 1. The symbol $f(X)$ is the fitness value of observation state X, and b_i and x_i are the i-th bit of the stored best solution X_{best}^t and a current solution X, respectively.

x_i	b_i	$f(X)\leq f(X_{best}^t)$	$\Delta\theta_{ji}^t$	$S(\alpha_{ji}^t, \beta_{ji}^t)$			
				$\alpha_{ji}^t\beta_{ji}^t>0$	$\alpha_{ji}^t\beta_{ji}^t<0$	$\alpha_{ji}^t=0$	$\beta_{ji}^t=0$
0	0	*false*	0	0	0	0	0
0	0	*true*	0	0	0	0	0
0	1	*false*	0	0	0	0	0
0	1	*true*	δ_j^t	-1	1	±1	0
1	0	*false*	δ_j^t	-1	1	±1	0
1	0	*true*	δ_j^t	1	-1	0	±1
1	1	*false*	δ_j^t	1	-1	0	±1
1	1	*true*	δ_j^t	1	-1	0	±1

Table 1. MS-RAS scheme for the i-th individual

We denote by δ_j^t the RAS value of the j-th individual. The principle difference among FRAS, DRAS and MS-RAS schemes is shown in Fig.6. At each generation, all individuals under either FRAS scheme or DRAS scheme are assumed to use only one RAS value, while each individual under MS-RAS scheme is able to use a special RAS value to get updated. The expression of δ_j^t is defined as follows:

$$\delta_j^t = \begin{cases} \sigma_0 , & t = 0 \\ \delta_j^{t-1} + \Delta(f_j^{t-1}, f_j^t), & t > 0 \,\&\, \delta_j^{t-1} + \Delta(f_j^{t-1}, f_j^t) > 0 \\ \Delta\sigma , & t > 0 \,\&\, \delta_j^{t-1} + \Delta(f_j^{t-1}, f_j^t) \leq 0 \end{cases} \tag{7}$$

where, we have

$$\Delta(f_j^{t-1}, f_j^t) = \begin{cases} 0 , & f(X_j^t) = f(X_j^{t-1}) \\ -\Delta\sigma , & f(X_j^t) > f(X_j^{t-1}) \\ \Delta\sigma , & f(X_j^t) < f(X_j^{t-1}) \end{cases} \tag{8}$$

Here, σ_0 and $\Delta\sigma$ are two constants initialized at the beginning of the algorithm which affect the convergent speed. Besides, $\Delta\sigma$ is smaller than σ_0. We set $\Delta\sigma = 0.1\sigma_0$ in this chapter. X_j^t and X_j^{t-1} are the observation states of the j-th individual at generation t and t-1, respectively. Indicating whether an individual evolves to a better or worse searching situation, the value of $\Delta(f_j^{t-1}, f_j^t)$ is selected from $\{-\Delta\sigma, 0, \Delta\sigma\}$ according to whether $f(X_j^t)$ is larger than $f(X_j^{t-1})$. If $f(X_j^t) > f(X_j^{t-1})$, $\Delta(f_j^{t-1}, f_j^t)$ is set to $-\Delta\sigma$, which means searching ability of the j-th individual is becoming worse and δ_j^t is then set to be smaller than δ_j^{t-1} to slow down its turn-to-bad performance. If $f(X_j^t) < f(X_j^{t-1})$, $\Delta(f_j^{t-1}, f_j^t)$ is set to $\Delta\sigma$, which means the j-th individual has reached a better searching position than its previous situation and δ_j^t is then larger than δ_j^{t-1} to accelerate its searching speed. If $f(X_j^t) = f(X_j^{t-1})$, it is not clear that the j-th individual performs better or worse, so we just reserve its previous RAS value.

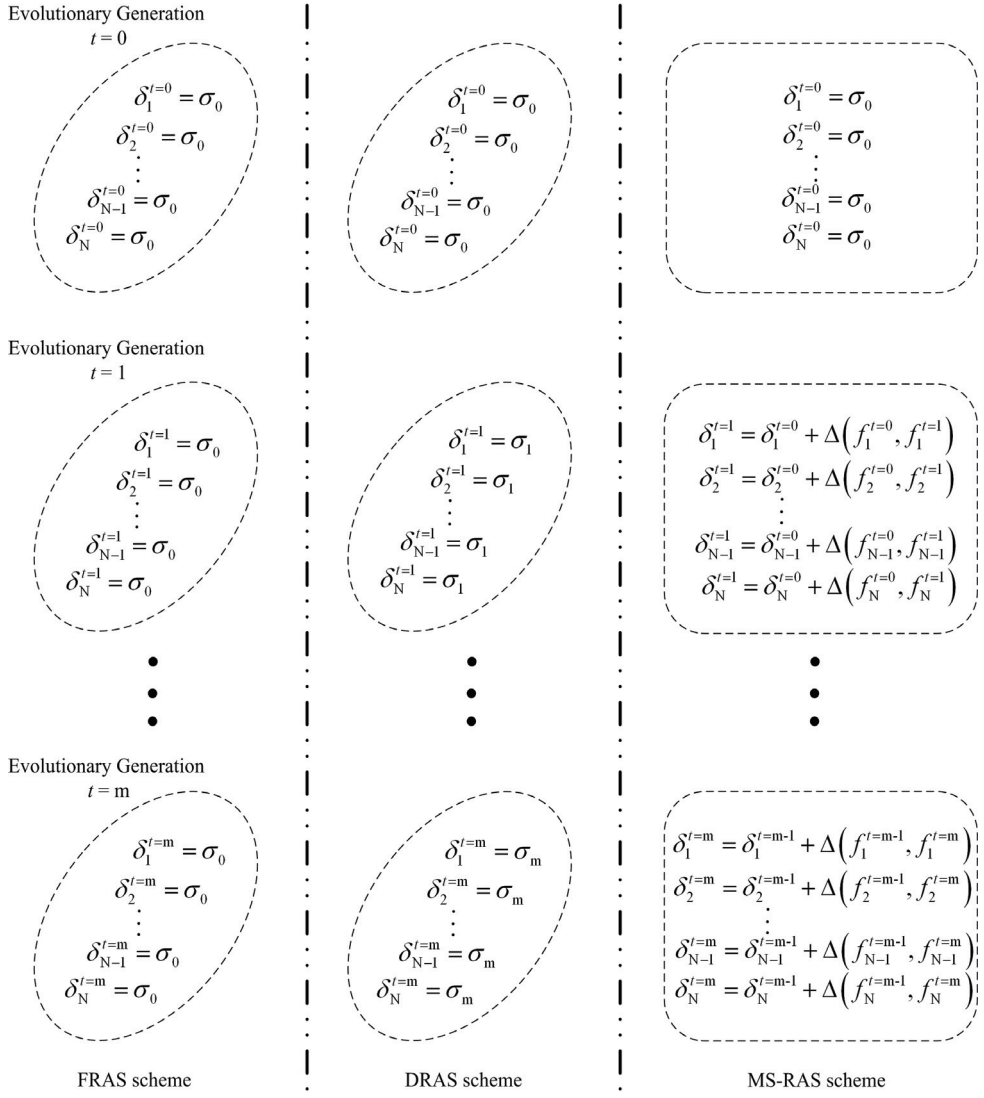


Fig. 6. Principle comparison of FRAS, DRAS and MS-RAS schemes

Based on the MS-RAS scheme, all individuals use the same RAS value σ_0 for update at the first generation. With the evolution continuing, the RAS value δ_j^t of the j-th individual at generation t is determined by its previous RAS value δ_j^{t-1} and its previous-and-current fitness values. When better than their previous searching performance, individuals are allowed to have relatively large rotation angle step (RAS) values to accelerate the exploration speed. When worse than their previous searching performance, individuals are allocated with relatively small RAS values to avoid invalid evolution. Thus, fast convergence and high optimization efficiency will be characterized.

4.2 Memory-Storable QMP scheme

Inspired by (Yang et al., 2003), a quantum mutation operation based on quantum NOT gate, called MS-QMP scheme, is proposed. Instead of employing only one mutation probability for the entire population, MS-QMP scheme assigns each individual a suitable QMP value according to the individual's own searching situation. For arbitrary individual at generation t, if its fitness value $f(X_j^t)$ is larger than $f(X_j^{t-1})$ at last generation, the current QMP value is set to be larger than its last RAS value; If its fitness value $f(X_j^t)$ is smaller than that of its last generation $f(X_j^{t-1})$, the current RAS value is set to be smaller than its last RAS value; Otherwise, the current RAS value is set to be equal to its last RAS value. The quantum mutation probability p_j^t of the j-th individual is defined as:

$$p_j^t = \begin{cases} p_0 & ,t = 0 \\ p_j^{t-1} + \varphi(f_j^{t-1}, f_j^t), & t > 0 \end{cases} \tag{9}$$

where we have,

$$\varphi(f_j^{t-1}, f_j^t) = \begin{cases} 0 & ,f(X_j^t) = f(X_j^{t-1}) \\ -\Delta p & ,f(X_j^t) < f(X_j^{t-1}) \\ \Delta p & ,f(X_j^t) > f(X_j^{t-1}) \end{cases} \tag{10}$$

where p_0 and Δp are two constants initialized at the beginning of the algorithm. We set $\Delta p = 0.1p_0$ in this chapter. Similar to $\Delta(f_j^{t-1}, f_j^t)$, $\varphi(f_j^{t-1}, f_j^t)$ is to indicate whether $f(X_j^t) > f(X_j^{t-1})$. If $f(X_j^t) > f(X_j^{t-1})$, $\varphi(f_j^{t-1}, f_j^t)$ is set to a positive constant Δp which makes p_j^t larger than p_j^{t-1} and the j-th individual is thus more likely to mutate. If $f(X_j^t) < f(X_j^{t-1})$, $\varphi(f_j^{t-1}, f_j^t)$ is set to a negative constant $-\Delta p$ which makes p_j^t smaller than p_j^{t-1} and the j-th individual thus has a higher probability to survive. If $f(X_j^t) = f(X_j^{t-1})$, we just set $\varphi(f_j^{t-1}, f_j^t) = 0$.

The better the individual, the smaller the assigned mutation probability. Based on MS-QMP scheme, excellent individuals have more chance to survive while those bad ones are more likely to mutate, hence pre-maturity is avoided and global searching capability is enhanced.

4.3 Individual representation

Number all merging nodes sequentially from 1 to L, where L is the number of merging nodes. Assume the i-th merging node has $m(i)$ incoming links and $n(i)$ outgoing links. There are precisely $2^{m(i)\cdot n(i)}$ different ways that an information flow passes by the node. Besides, there are $2^{m(i)}$ possible ways that the information flow from $m(i)$ incoming links may pass by the j-th outgoing link of the i-th merging node. Obviously, there are $2^{\sum_{i=1}^{L} m(i)\cdot n(i)}$ possible ways that a multicast passes through all merging nodes.

Since each Q-bit represents a superposition of states '1' and '0' and it collapses to an explicit state after each measurement, i.e. either '1' or '0', it is suitable to represent incoming link states *active* or *inactive*, of a merging node (refer to section 2). This chapter adopts a $m(i)\cdot n(i)$ Q-bit individual representation $G = \{g_1, g_2, ..., g_L\}$, where g_i, $i = 1, 2, \ldots, L$, is the block of incoming link states for the i-th merging node. The segment g_i includes $n(i)$ sub-segments, each of which represents the block information of the k-th outgoing link of the i-th merging node, where $k = 1, 2, ..., n(i)$. After observing g_i, a certain block of incoming link states for the i-th merging node is achieved (see Fig.7 as an example).

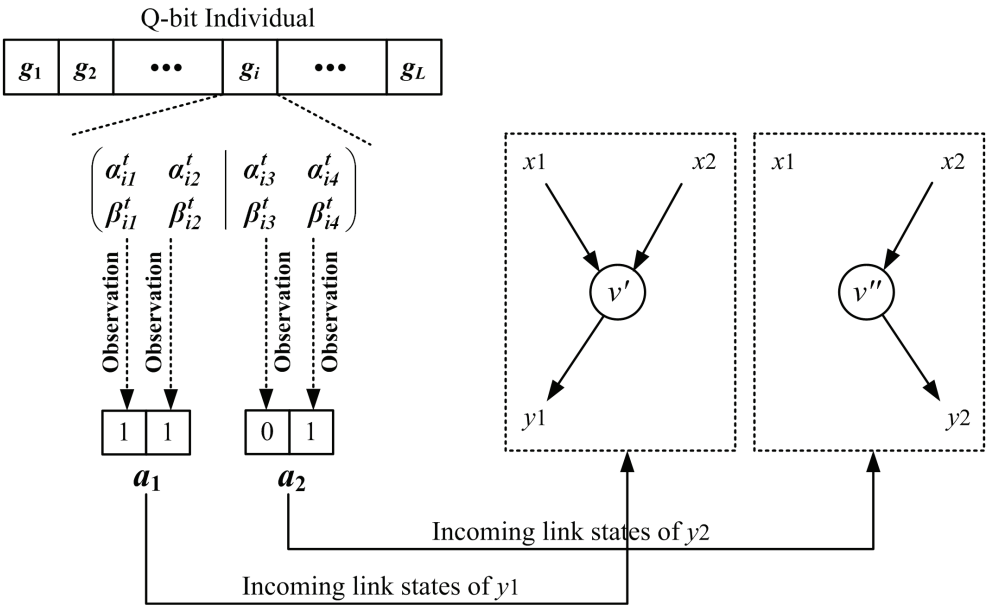


Fig. 7. Individual representation

4.4 Fitness evaluation

This section provides a penalty-function-based fitness function (PF-FF) definition. Based on PF-FF, the fitness value $f_{PF\text{-}FF}(X)$ of observation state X of individual q is defined as

$$f_{PF-FF}(X) = \begin{cases} n_{cl} & , \ Flag(X) = 0 \\ \sum_{i=1}^{d} \Theta(s,t_i), & Flag(X) = 1 \end{cases} \tag{11}$$

Where we have,

$$\Theta(s,t_i) = \begin{cases} 0, & Flag(s,t_i) = 0 \\ \Gamma, & Flag(s,t_i) = 1 \end{cases} \tag{12}$$

where, $Flag(X)$ is to show whether the observation state X of individual q is feasible (According to (Kim et al., 2007a; 2007b), if the achieved max-flow is no less than the desired max-flow, we regard X *feasible*, otherwise, we regard X *infeasible*). If $Flag(X) = 0$, it means X is feasible and $f_{PF\text{-}FF}(X)$ is set to n_{cl}, where n_{cl} is the number of actual coding links. If $Flag(X) = 1$, X is supposed *infeasible* and must be punished. $Flag(s,t_i)$ is to indicate whether a desired max-flow is achieved from s to t_i, where s is the source node and t_i, $i = 1, 2, \ldots, d$, is the i-th sink node. If $Flag(s,t_i) = 0$, the max-flow from s to t_i is achievable and the penalty factor $\Theta(s,t_i)$ is invalid. However, $Flag(s,t_i) = 1$ indicates that the max-flow from s to t_i can not be achieved and punishment should be added. Here, Γ is a constant that indicates the explicit amount of punishment.

To verify the feasibility of an observation state *X*, *Graph Decomposition Method*, by which every merging node in a given network is decomposed into a group of nodes, is adopted to calculate the max-flow from the source to the sinks (Kim et al., 2007a).

4.5 The structure of MS-QIEA

As each part of the algorithm being already introduced in details, the basic steps of the algorithm can be described in Fig.8.

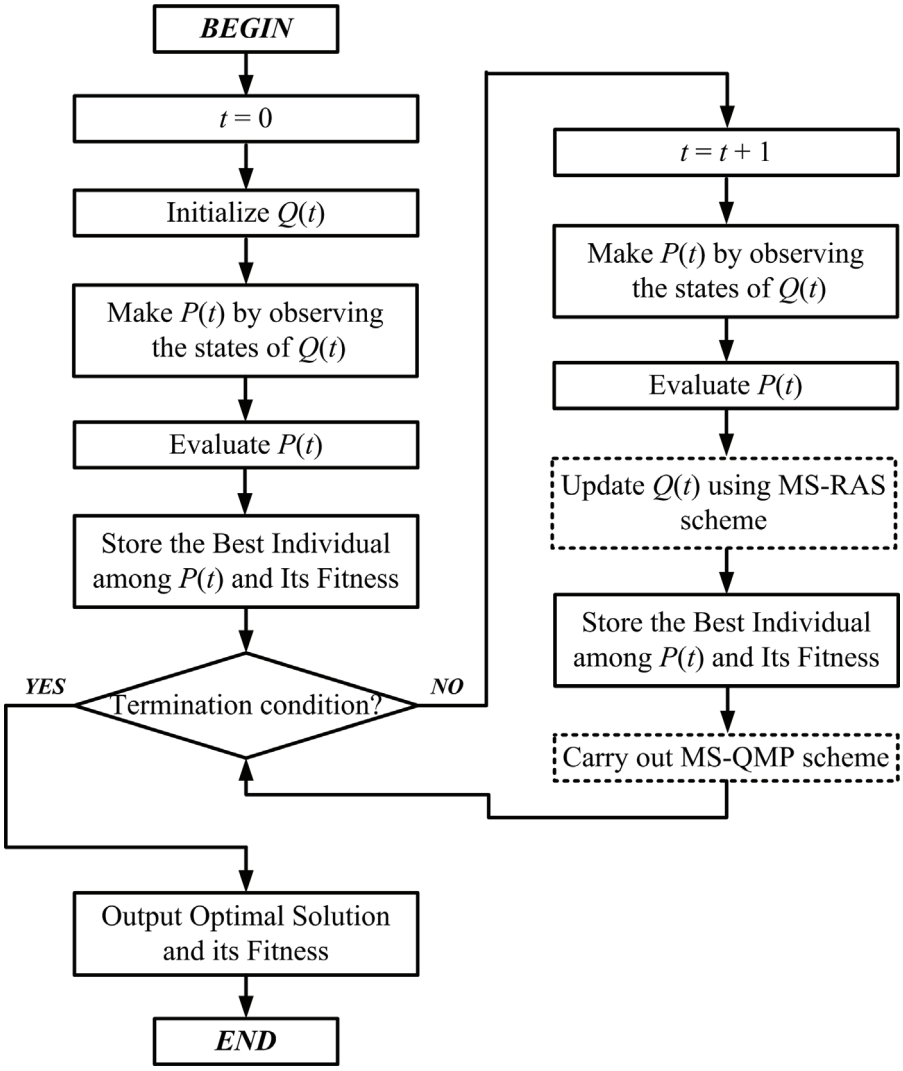


Fig. 8. Flow chart of MS-QIEA

The procedure of MS-QIEA is similar to that of QIEA except for two differences: (1) instead of FRAS and DRAS schemes, MS-RAS scheme is adopted to accelerate the convergence of the algorithm, and (2) MS-QMP scheme is introduced so as to enhance the global searching capability.

5. Experimental results and discussions

In order to evaluate the performance of the proposed algorithm, comparisons of GA (Kim et al., 2007b), QIEA with DRAS scheme (called D-QIEA below), and MS-QIEA have been carried out over three network topologies with the following parameters: Network-I (21nodes, 30links, 4sinks, rate=2), Network-II (20nodes, 37links, 5sinks, rate=3) and Network-III (40nodes, 85links, 10sinks, rate=4). Links in Network-I, Network-II and Network-III are supposed to have unit capacity. Note that all the above algorithms are based on binary link state (BLS) encoding approach (Kim et al., 2007b). However, different from (Kim et al., 2007b), no all one vector is inserted to the initial population so that the algorithms in this chapter may fail to find feasible NCM subgraphs. The population size n_{PS} and terminal iteration n_{TI} of the three algorithms under the cases of Network-I, Network-II, and Network-III are set to: n_{PS}(20, 20, 40) and n_{TI}(100, 300, 400) respectively. In GA, the crossover and the mutation rates are set to 0.8 and 0.1 respectively. In D-QIEA, the initialized RAS value is set to 0.05π. In MS-QIEA, set $\sigma_0 = 0.04\pi$, $\Delta\sigma = 0.004\pi$, $p_0 = 0.1$, $\Delta p = 0.01$ and $\Gamma = 20$. Performance comparisons and simulation results obtained in 500 random trials for each multicast scenario are shown in table 2 and Fig.9-Fig.11.

Multicast Scenarios		Algorithms		
		GA	D-QIEA	MS-QIEA
Network-I	MSR	100.0%	100.0%	100.0%
	BMNCL	2	2	2
	MMNCL	2.00	2.00	2.00
Network-II	MSR	98.7%	99.5%	100.0%
	BMNCL	0	0	0
	MMNCL	0.03	0.01	0.00
Network-III	MSR	99.1%	100.0%	100.0%
	BMNCL	0	0	0
	MMNCL	0.12	0.00	0.00

Table 2. Performance comparisons in three multicast scenarios

By running an algorithm (GA, QIEA, or MS-QIEA) once, one best NCM subgraph $G_{NCM}(s,T)$ can be obtained. So, by running an algorithm 500 times, 500 best NCM subgraphs were achieved. Among these subgraphs, some of them cannot achieve the desirable multicast rate while others (suppose there are W such subgraphs) are feasible ones. In table 2, MSR

represents Multicast Success Ratio, where MSR = W/500. BMNCL is the Best Minimum Number of Coded Links achieved among 500 trials, and MMNCL denotes the Mean Minimum Number of Coded Links over 500 trials. Table 2 shows that MS-QIEA outperforms D-QIEA and GA in each case since MS-QIEA always gets the highest MSR and the lowest BMNCL and MMNCL. The second best algorithm is D-QIEA, and the worst one is GA. High MSR indicates that MS-QIEA has higher probability to successfully construct a NCM subgraph with less coding links each time, which reflects the robustness of MS-QIEA. Furthermore, lower BMNCL and MMNCL demonstrate that MS-QIEA has better optimization performance on global searching than QIEA and GA.

Fig.9, Fig.10, and Fig.11 show the convergent speed of the three algorithms in three multicast scenarios. In each figure, note that MS-QIEA distinguishes itself by the fastest convergence. This is because MS-QIEA fully considers individual difference and is able to allocate suitable evolutionary parameter values to each individual according to the individual's previous searching situation, which makes it more suitable to speed up the search and to avoid local optima. As better characteristic of population diversity is achieved by the two QIEA algorithms, broader solution space is explored and exploited simultaneously so that MS-QIEA and D-QIEA perform better than GA.

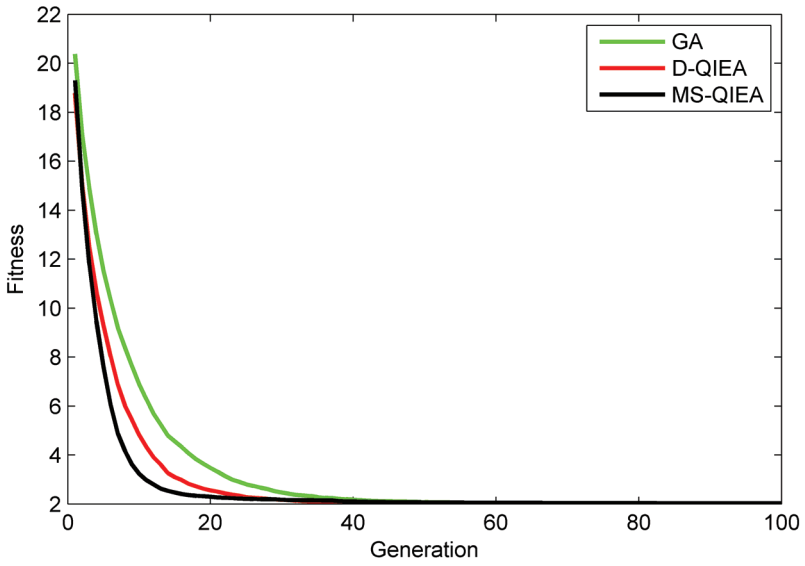
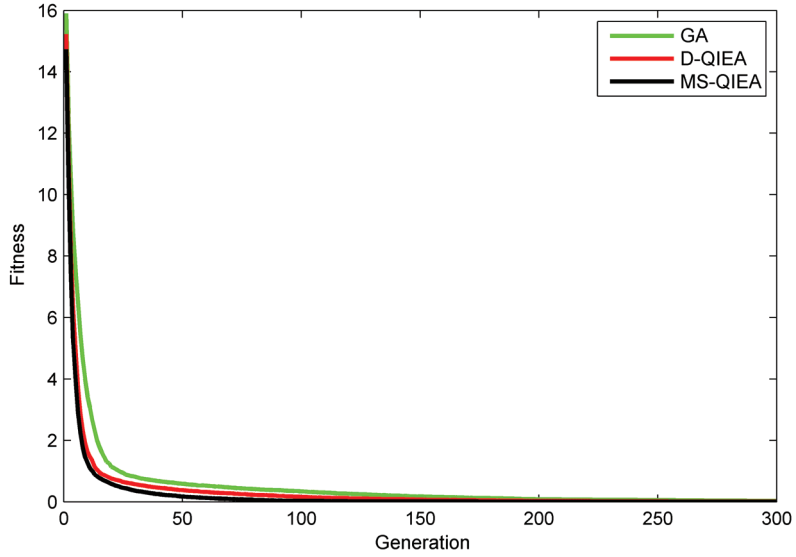
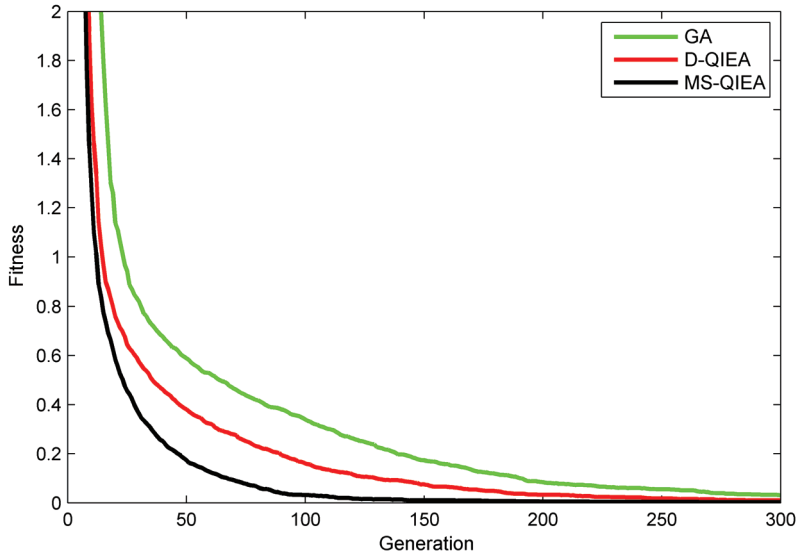


Fig. 9. Comparisons of convergence performance in Network-I



(*a*)



(*b*)

Fig. 10. Comparisons of convergence performance in Network-II. (a) Absolute curves without obviously identified trends. (*b*) Curves to indicate evidence in details that MS-QIEA is superior to D-QIEA and GA in terms of convergence and global searching capability.

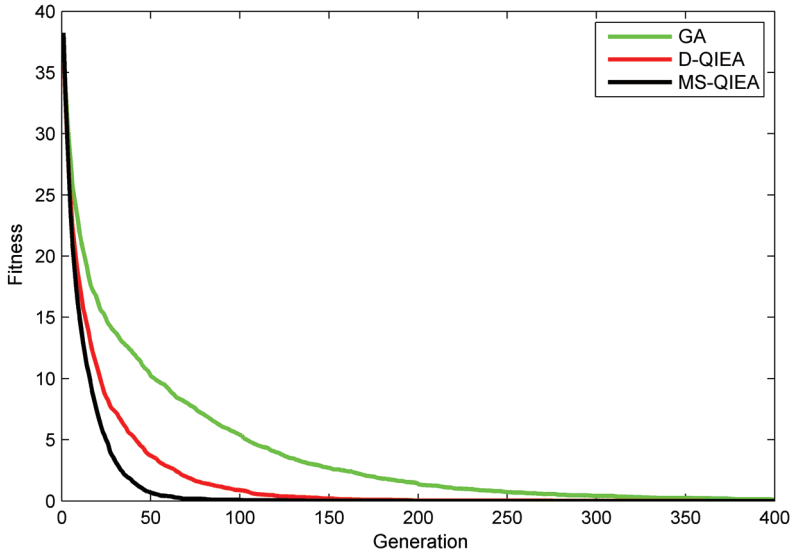


Fig. 11. Comparisons of convergence performance in Network-III

6. Conclusions

An improved QIEA (MS-QIEA) has been proposed for coding resource optimization problem. MS-QIEA updates its individuals with respect to their previous evolution situation. Excellent individuals (the ones achieve increasingly better solutions) are allowed to have relatively large RAS values to accelerate the exploration speed and relatively small QMP values to survive. Inferior individuals (the ones perform worse and worse) are allocated with relatively small RAS values to avoid invalid evolution and relatively large QMP values to expect a chance to become excellent individuals. The simulation results clearly demonstrate the superiority of this algorithm over GA and QIEA in terms of robustness, success ratio, convergence and global exploration.

7. Acknowledgement

This research was supported in part by NSFC (No. 60932004), National 973 Program (No. 2007CB310705), National 863 Program (No. 2009AA01A345), P. R. China.

8. References

Ahlswede, R.; Cai, N.; Li, S. R. & Yeung, R. W. (2000). Network information flow, *IEEE Transactions on Information Theory*, Vol.46, No.4, July 2000, pp. 1204-1216.

Bhattad, K.; Ratnakar, N.; Koetter, R. & Narayanan, R. (2005). Minimal network coding for multicast, *Proceedings on International Symposium on Information Theory (ISIT2005)*, pp. 1730-1734, Adelaide, Australia, September 2005.

Fragouli, C. & Soljanin, E. (2006). Information flow decomposition for network coding, *IEEE Transactions on Information Theory*, Vol.52, No.3, March 2006, pp. 829-848.

Han, K. H.; Park, K. H.; Lee, C. H. & Kim, J. H. (2001). Parallel quantum-inspired genetic algorithm for combinatorial optimization problem, *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, pp. 1422-1429, Seoul, Korea, May 2001.

Han, K. H. & Kim, J. H. (2002). Quantum inspired evolutionary algorithm for a class of combinatorial optimization, *IEEE Transactions on Evolutionary Computation*, Vol.6, No.6, December 2002, pp. 580-593.

Han, K. H. & Kim, J. H. (2004). Quantum-inspired evolutionary algorithms with a new termination criterion, H gate, and two-phase scheme, *IEEE Transactions on Evolutionary Computation*, Vol.8, No.2, April 2004, pp. 156-169.

Kim, M.; Ahn, C. W.; Medard, M. & Effros, M. (2006). On minimizing network coding resources: an evolutionary approach, *Proceedings of Second Workshop on Network Coding, Theory, and Applications (NetCod2006)*, April 2006.

Kim, M.; Medard, M.; Aggarwal, V.; O'Reilly, U.-M.; Kim, W.; Ahn, C. W. & Effros, M. (2007a). Evolutionary approaches to minimizing network coding resources, *Proceedings of 26th IEEE International Conference on Computer Communications (INFOCOM2007)*, pp.1991-1999, Anchorage, Alaska, May 2007.

Kim, M.; Aggarwal, V.; O'Reilly, U.-M.; Medard, M. & Kim, W. (2007b). Genetic representations for evolutionary optimization of network coding, *Proceedings of EvoComnet*, 2007.

Langberg, M.; Sprintson, A. & Bruck, J. (2006). The encoding complexity of network coding, *IEEE Transactions on Information Theory*, Vol.52, No.6, June 2006, pp. 2386-2397.

Li, B. & Wang, L. (2007). A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling, *IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics*, Vol.37, No.3, June 2007, pp. 576-591.

Li, S.R.; Yeung, R. W. & Cai, N. (2003). Linear network coding, *IEEE Transactions on Information Theory*, Vol.49, No.2, February 2003, pp.371-381.

Lv, Y. & Liu, N. (2007). Application of quantum genetic algorithm on finding minimal reduct, *Proceedings of 2007 IEEE International Conference on Granular Computing*, pp. 728-733, Silicon Valley, California, November 2007.

Platel, M. D.; Schliebs, S. & Kasabov, N. (2009). Quantum-inspired evolutionary algorithm: a multimodel EDA. *IEEE Transaction on Evolutionary Computation*, Vol.13, No.6, December 2009, pp. 1218-1232.

Xing, H.; Liu, X.; Jin, X.; Bai, L. & Ji, Y. (2009a). A multi-granularity evolution based quantum genetic algorithm for QoS multicast routing problem in WDM networks. *Computer Communications*, Vol.32, No.2, February 2009, pp. 386-393.

Xing, H.; Ji, Y.; Bai, L.; Liu, X.; Qu, Z. & Wang, X. (2009b). An adaptive-evolution-based quantum-inspired evolutionary algorithm for QoS multicasting in IP/DWDM networks. *Computer Communications*, Vol.32, No.6, April 2009, pp. 1086-1094.

Yang, J.; Li, B. & Zhuang, Z. (2003). Multi-universe parallel quantum genetic algorithm and its application to blind source separation, *Proceedings of 2003 IEEE International Conference on Neural Networks and Signal Processing*, pp. 393-398, Nanjing, China, December 2003.

Zhang, G.; Gu, Y.; Hu, L. & Jin, W. (2003). A novel genetic algorithm and its application to digital filter design, *Proceedings of 2003 IEEE Intelligent Transportation Systems*, Vol.2, pp.1600-1605, Shanghai, China, October 2003.

20

Using Evolutionary Algorithms for Optimization of Analogue Electronic Filters

Lukáš Dolívka and Jiří Hospodka
Czech Technical University in Prague
Czech Republic

1. Introduction

During several recent years, the performance of computer technology increased enough to enable the using of numerical methods in various branches more commonly then in the past when mainly analytical methods were used. Numerical methods are utilized both in a design and in optimization of various systems. One of the advantages of numerical methods is the possibility of meeting more requirements – they are able to solve multi-objective (multi-criteria) tasks.
Of course, numerical methods are applied also in electronics while designing electronic circuits. Thanks to these methods, circuits can be designed with more aspect taken into account. The aspects are, naturally, primarily main circuit requirements, e.g., a magnitude frequency response, then other various features, e.g., a group delay frequency response, a dynamic range, consumption etc. However, in addition, also the parameters of used components – the tolerance and spread of their values, their real features, circuit characteristic sensitivities to their values, and so on – can be taken into consideration.
If an analytical method shall be applied to satisfy so many requirements, the circuit design would become either unfeasible or so complicated that it would be unsuccessful or inaccurate. However, numerical methods are able to operate even with a low or at least acceptable complicacy rate. On the other hand, numerical methods have disadvantages as well. One of the most significant is a higher time consumption to obtain a result compared to analytical methods. Nevertheless, this is not such a problem mostly.

2. Brief description of Evolutionary Algorithms

Evolutionary algorithms (EAs) also belong in numerical methods (Corne et al., 1999). They represent robust and powerful optimization techniques. Their theme is explored in detail today and they have been applied many times in various branches.
Other methods can be also utilized to find the extreme of a function. For instance: searching the extreme by means of the differentiations of the function, various gradient methods, simple numerical methods, or other optimization methods (Pintér, 1996), but these methods often provide not the global but a local extreme. However, the result of an optimization task should be always the most advantageous – optimal – state, which corresponds to the global extreme.
EAs are applied to finding the solution of optimization tasks. This solution has to satisfy some determined conditions. The merit of the found solution is evaluated by the value of an

objective function (OF), which is necessary for EA operation. EAs aim to optimize the OF value, i.e., aim to find its maximal or minimal value – it depends on a particular optimization task. Thus, they try to find the global extreme of the OF. The OF has a certain number of variables, denoted *n* here, (which is given by the task) and EAs search for its optimal value by finding suitable values of the variables. EAs proceed progressively in generations (cycles), in which a better and better OF value is achieved. Every generation is composed of a certain number of vectors, denoted *N* here, whose entries are the OF variables. A set in which the values of the variables of the OF, i.e., also the solution of the task, can occur (a search space) has to be defined in advance. The set can be identical to the definition scope of the OF or it can be its subset.

EAs feature several advantages compared to other methods for finding a global extreme, e.g.:

- The only property required from the OF is its value for given variable values from its definition range (which is obviously fulfilled for every function). Neither the continuity nor differentiability of the OF is required.
- If the OF has more than one global extreme, EAs are able to find them, i.e., they can provide more than one solution.
- They focus on searching for global extreme(s), not for local one(s).

However, EAs have also disadvantages:

- They need a longer time for finding the optimization task solution. This is due to their robustness.
- They utilize randomness, thus the optimization time cannot be predicted. Therefore, computing times may be different when the same optimization task is solved several times.

EAs are appropriate for solving complicated tasks. Such tasks cannot be usually solved by analytical methods. Alternatively, it would be possible but it would not be lucid, consequently, errors could arise.

Several techniques are ranked among EAs:

- genetic algorithms,
- evolutionary strategy,
- genetic programming,
- evolutionary programming,
- differential evolution,
- other algorithms.

The utilization of EAs is widespread today, both in electronics, e.g., (Dolívka & Hospodka, 2007 c; Storn, 1996 b; Vondraš & Martinek, 2002; Žiška & Vrbata, 2006) – used the differential evolution, (Haseyama et al., 1996) – used genetic algorithms, (Gielen et al., 1990) – used simulated annealing, and in other branches, e.g., (Brutovský et al., 1995; Chambers, 2000; Dasgupta & Michalewicz, 1997).

3. Optimization of analogue electronic circuits

EAs are very suitable for an optimization of analogue electronic circuits as well. This is thanks to their advantageous features mentioned above. Several aspects of optimizing analogue electronic circuits are discussed in this section.

3.1 Objective function

The general form of the OF denoted U is

$$U(x_1, x_2, \ldots, x_n): X \rightarrow \mathbf{R}, \tag{1}$$

where $\mathbf{R}$ is the set of real numbers. The set X (a search space) and all the variables x_1 to x_n of the OF are as follows

$$X = X_1 \times X_2 \times \cdots \times X_n \subseteq S_1 \times S_2 \times \cdots \times S_n, \tag{2}$$

$$x_i \in X_i, \;\; X_i \subseteq S_i, \;\; i = 1, 2, \ldots, n, \tag{3}$$

where S_i is the set of real, rational, integer, natural, complex, discrete or other numbers. Thus, the OF converts an n-dimensional space into a one-dimensional set of real numbers.
If all the sets S_i are the sets of real numbers (i.e., if $S_i = \mathbf{R}$ for $i = 1$ to n) – this occurs often, (2) is changed into a simpler form

$$X \subseteq \underbrace{\mathbf{R} \times \mathbf{R} \times \cdots \times \mathbf{R}}_{n} = \mathbf{R}^n \tag{4}$$

and all the variables x_1 to x_n are real numbers.
In case of electronic circuit optimization, the OF has a special form. The OF includes a function $O(x_0, x_1, x_2, \ldots, x_n)$, whose shape shall be optimized, i.e., changed so that it satisfies requirements. The function O has the same variables x_1 to x_n as the OF. In addition to them, it has a variable x_0, which is usually a real number. After the optimization, the values of the function O at defined values of the variable x_0 should be in defined intervals. The other variables x_1 to x_n of the optimized function O can be regarded as its parameters. The required shape of the function O is obtained by finding suitable values for them. Hence, the function O after the optimization should meet this condition

$$O(w_i, x_1, x_2, \ldots, x_n) \in \langle O_{\mathrm{D}}(w_i), O_{\mathrm{H}}(w_i) \rangle \;\; \forall w_i, \tag{5}$$

where $O(w_i, x_1, x_2, \ldots, x_n)$ is the value of the function O if the variable x_0 is substituted by a value w_i. $O_{\mathrm{D}}(w_i)$ and $O_{\mathrm{H}}(w_i)$ are a lower and upper bound of the values of the function O if the variable x_0 is equal to w_i.
Note that, generally, both of the bounds need not be determined. For some values w_i, the condition (5) can be changed to one of these forms

$$O(w_i, x_1, x_2, \ldots, x_n) \leq O_{\mathrm{H}}(w_i) \;\; \text{if } O_{\mathrm{D}}(w_i) = -\infty, \tag{6}$$

$$O(w_i, x_1, x_2, \ldots, x_n) \geq O_{\mathrm{D}}(w_i) \;\; \text{if } O_{\mathrm{H}}(w_i) = \infty, \tag{7}$$

$$O(w_i, x_1, x_2, \ldots, x_n) = O_{\mathrm{K}}(w_i) \;\; \text{if } O_{\mathrm{D}}(w_i) = O_{\mathrm{H}}(w_i) = O_{\mathrm{K}}(w_i). \tag{8}$$

While using the function O, the OF is expressed by (9), d and h are the number of $O_{\mathrm{D}}(w_i)$ and $O_{\mathrm{H}}(w_i)$, respectively

$$U(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{d} U_{\mathrm{D}i}(x_1, x_2, \ldots, x_n) + \sum_{i=1}^{h} U_{\mathrm{H}i}(x_1, x_2, \ldots, x_n), \tag{9}$$

where the terms U_{Di} and U_{Hi} are

$$U_{Di}(x_1, x_2, \ldots, x_n) = \begin{cases} \dfrac{O_D(w_i) - O(w_i, x_1, \ldots, x_n)}{O_D(w_i)} & \text{if } O_D(w_i) > O(w_i, x_1, \ldots, x_n), \\ 0 & \text{else,} \end{cases} \tag{10}$$

$$U_{Hi}(x_1, x_2, \ldots, x_n) = \begin{cases} \dfrac{O(w_i, x_1, \ldots, x_n) - O_H(w_i)}{O_H(w_i)} & \text{if } O_H(w_i) < O(w_i, x_1, \ldots, x_n), \\ 0 & \text{else.} \end{cases} \tag{11}$$

It is obvious that if condition (5) is fulfilled, the OF has the zero value, which is its global minimum.

If the optimization shall meet more requirements, the OF can be created by the sum of more terms from (9) or terms with another form. Therefore, the OF can contain more than one function O. A penalization function can be included into the OF too. The penalization can express, e.g., a requirement for the stability of the optimized circuit.

When the OF is created by more terms, finding the global extreme of the whole OF means satisfying all the particular requirements.

In practise, the variables x_1 to x_n represent parameters the optimized circuit. The parameters can be, e.g., the values of its components or the values of zeros and poles of its transfer function. The variable x_0 is mostly frequency. The function O means an optimized circuit characteristic, e.g., a magnitude frequency response, group delay frequency response. The bounds $O_D(w_i)$ and $O_H(w_i)$ specify the intervals of the values of this characteristic. For instance, if the variable x_0 is frequency f and the function O is a magnitude frequency response $M(f)$, the bounds $O_D(w_i) = M_D(f_i)$ and $O_H(w_i) = M_H(f_i)$ determine the range in which the value of the magnitude can be at a certain frequency f_i.

3.2 Programs for optimization and analysis

A program for implementing calculations of an applied optimization algorithm is necessary for performing the optimization of a circuit.

Because mathematical operations have to be made during performing the optimization algorithm, a program capable of doing it is necessary for this purpose. Another feature that the program should have is the possibility of symbolical calculation. Suitable programs are Maple™ (Waterloo Maple) and MATLAB® (MathWorks) – one of the most widespread mathematical programs.

Moreover, it is necessary to carry out the analysis of an optimized circuit during its optimization. As a result, a program for the analysis of the optimized circuit is necessary besides the program for implementing calculations of the used optimization algorithm. This program can be, e.g.:

- PraCAn (Bičák & Hospodka, 2008): a library of functions for the Maple™ program, which facilitates a symbolic and semisymbolic analysis of continuous-working and discrete-working real linearized circuits. (PraCAn is an acronym for Prague Circuits Analyzer.)
- WinSpice (Smith): a general-purpose well-known Spice-compatible program for circuit simulation.
- Cadence (Cadence Design Systems) and Mentor Graphics (Mentor Graphics Corp.): professional programs (not only) for numerical analyzing circuits.

Of course, the most suitable program for analyzing circuits is the one that can be utilized directly in a program for implementing the optimization algorithm, otherwise, the analyzing program works as an external one, which have to be called from the mathematical program. This can lead to lower efficiency of the optimization. Thus, for the Maple™ program, the PraCAn library can be recommended. This combination of programs is used also by the authors.

3.3 Optimization methods

A powerful enough optimization method should be applied for the optimization of analogue electronic circuits. According to authors' experience, the most suitable one is the differential evolution (DE). Therefore, the authors applied this method for their optimization tasks described below.

The DE (Corne et al., 1999; Storn & Price, 1997) comes from the first half of the 90s. It is a general-purpose powerful algorithm, which has been already used in many applications. It was chosen by the authors owing to its several advantageous features (e.g., good convergence properties, ease of use, conceptual simplicity, and only a few control variables) and because it is able to achieve better results than other EAs (Storn & Price, 1996). The DE is controlled by two parameters:

- CR – a crossover constant, $CR = 0$ to 1,
- F – a mutation constant, $F = 0$ to 2.

There are a few versions of the DE (Storn & Price, 1997).

A detailed description of the DE cannot be presented in this chapter because of its limited extent. For more information about the DE, refer to the mentioned references.

From the other EAs, genetic algorithms (GA) (Goldberg, 1989) are not so powerful.

When the DE is combined with another method, its efficiency is better. The most suited one is the simplex method (Nelder & Mead, 1965).

Five examples of an analogue electronic circuit optimization are shown in the next sections to better explain and document the theory of this optimization described above. In the last example, several optimization methods are compared to each other.

4. Examples of optimization of analogue continuous-working circuits

Analogue continuous-working circuits represent a big group of electronic circuits, e.g., filters, power supplies, amplifiers, oscillators, etc. Many publications about optimization of analogue continuous-working circuits have been written, e.g., (Gielen et al., 1990; Tichá & Martinek, 2005; Vondraš & Martinek, 2002; Žiška & Vrbata, 2006).

From all the mentioned kinds of analogue continuous-working circuits, filters (i.e., selective circuits) were chosen for this section. Filters can be implemented by several techniques. In this section, attention is paid to two of them, whose nonideal features are optimized by means of an EA. This section describes two optimizations:

- optimization of an LC filter – a passive filter,
- optimization of an ARC filter – an active filter.

4.1 Optimization of LC filter

4.1.1 Introduction

The utilization of LC filters in electronics started in the beginning of the previous century. However, they are utilized still owing to their several advantages, e.g.:

- low sensitivity of their transfer function to component values,
- the methods of their design are examined thoroughly,
- they can be applied in a wide frequency range – from tens Hz up to hundreds MHz.

The design of an LC filter should satisfy a determined magnitude filter specification. In some cases, also the ripple of a group delay frequency response should be considered besides the magnitude filter specification for a better circuit design. This ripple should be as low as possible. This can be done analytically but it can lead to a high filter order – a high number of circuit components. If a numerical method (e.g., an EA) is applied, the order of filter can be lower.

The components creating LC filters are inductors L and capacitors C. Hence, the nonidealities in these filters are related to the nonidealities of these components. The nonidealities of real capacitors can be neglected whereas the nonidealities of real inductors are mostly so significant that they should be taken into account. The most important nonideality of real inductors is their finite quality factor. In most cases, the effect of this nonideality on the filter transfer function has to be eliminated during the filter design. This can be accomplished either by means of prewarping the magnitude filter specification (usually causing a higher filter order) or by means of optimization.

4.1.2 Optimized circuit and its required parameters

The optimized LC filter, depicted in Fig. 1, realizes a band-pass transfer function.

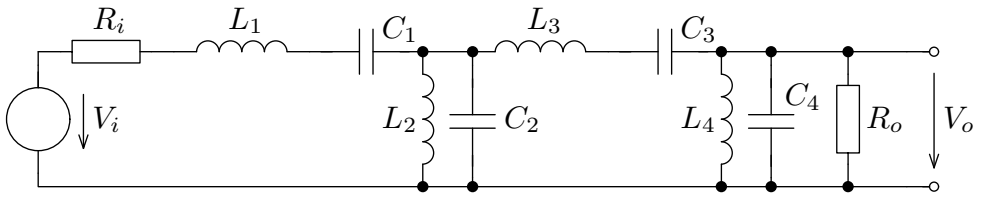


Fig. 1. Eighth-order LC band-pass filter.

The input and output resistance R_i and R_o are 1 kΩ.

The transfer function $P(f)$ of the filter is defined as the ratio of the output and input voltage

$$P(f) = \frac{V_o}{V_i}. \tag{12}$$

The magnitude frequency response $|P(f)|$ is denoted $M(f)$.

The shape of the magnitude frequency response of this filter should be according to a required magnitude filter specification in Fig. 2.

4.1.3 Description of optimization

The result of optimization should be:

- the meeting of a required magnitude filter specification – see Fig. 2,
- achieving the ripple of the group delay frequency response in the pass band not higher than 400 ns.

These requirements should be fulfilled by means of finding suitable values of the inductors and capacitors. The nonideality considered in the filter was a finite quality factor of the inductors. Its used value was 50.

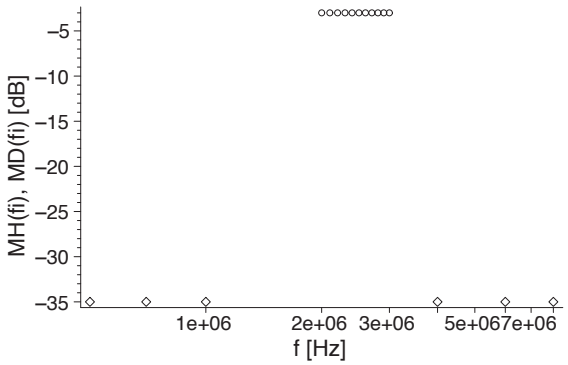


Fig. 2. Magnitude filter specification for optimization, diamonds: upper bounds $M_H(f_i)$ of magnitude ranges, circles: lower bounds $M_D(f_i)$ of magnitude ranges.

The DE was applied as the optimization algorithm in this example. The analytical solving of this example is feasible but complicated. The parameters of the optimization process were as follows:

- $CR = 0.9$, $F = 0.5$,
- $d = 11$, $h = 6$,
- $N = 80$, $n = 8$ (= the number of all the inductors and capacitors),
- S_1 to S_8 = **R**,
- X_1 to $X_4 = \langle 10^{-6}, 10^{-3} \rangle$ – the values of all the inductances can be from 1 μH to 1 mH,
- X_5 to $X_8 = \langle 10^{-12}, 10^{-9} \rangle$ – the values of all the capacitances can be from 1 pF to 1 nF.

The optimized function O is the magnitude M. It has a very long form. Therefore, it is not presented here. The meaning of the variables of the functions O is as follows:

- x_0 represents frequency f, w_i is substituted by f_i,
- x_1 to x_4 represent L_1 to L_4,
- x_5 to x_8 represent C_1 to C_4.

The OF was created by adding a term Z_{GD} to (9) in order to express the group delay optimization, τ_R means the value of the group delay ripple, τ_{Rmax} is the required maximal value of the group delay ripple (400 ns)

$$Z_{GD}(L_1,\ldots,L_4,C_1,\ldots,C_4) = \begin{cases} \dfrac{\tau_R(L_1,\ldots,L_4,C_1,\ldots,C_4) - \tau_{Rmax}}{\tau_{Rmax}} \\ \quad \text{if } \tau_R(L_1,\ldots,L_4,C_1,\ldots,C_4) > \tau_{Rmax}, \\ 0 \quad \text{else.} \end{cases} \tag{13}$$

4.1.4 Result from optimization

The component values arisen from the optimization are listed in Table 1. The needed number of generations was 515. The magnitude frequency response of the circuit using these values is displayed in Fig. 3.

i	1	2	3	4
L_i [μH]	282.94	18.823	191.71	27.524
C_i [pF]	15.177	241.51	24.542	164.87

Table 1. Component values from optimization.

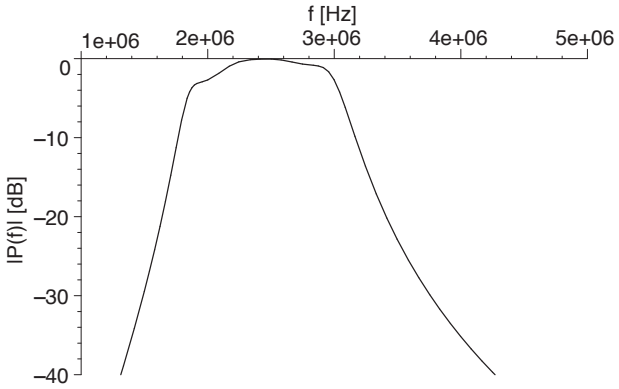


Fig. 3. Magnitude frequency response $|P(f)|$ obtained from optimization.

4.2 Optimization of ARC Filter

4.2.1 Introduction

Combining an RC network with a gain element can lead to transfer function with hogh Q factor complex poles (Schaumann et al., 1990). These filters are called ARC (active RC). They are used instead of LC filter to miniaturize the realization, for example, because inductors tend to be large and bulk, especially at low frequencies. ARC filters are often applied in both discrete and integrated form – both hybrid and monolithic. The active component can be realized by different elements – operational amplifiers, transconductance or transimpedance amplifiers, current conveyors, etc.

4.2.2 Optimized circuit and Its required parameters

The circuit in Fig. 4 was chosen to demonstrate possibilities of optimization of ARC filters. It represents a basic band-pass filter with cascade realization by two 2nd order blocks – biquads. They use operational amplifiers as a gain element.

The nonidealities of the resistors and capacitors are not necessary to be considered whereas in case of the amplifiers, their nonidealities have to be respected.

If the amplifiers are common operational amplifiers, they have these main nonidealities:

- finite input resistance,
- nonzero output resistance,
- finite slew rate,
- finite unity-gain bandwidth,
- finite voltage gain.

The effect of input resistance is usually insignificant, especially when field-effect transistors on the operational amplifier inputs are used. The output resistance effect is usually also less important. The slew rate can be neglected when signals have low amplitude. However, the

remaining two features – unity-gain bandwidth and voltage gain – affect the transfer function of ARC filters substantially.

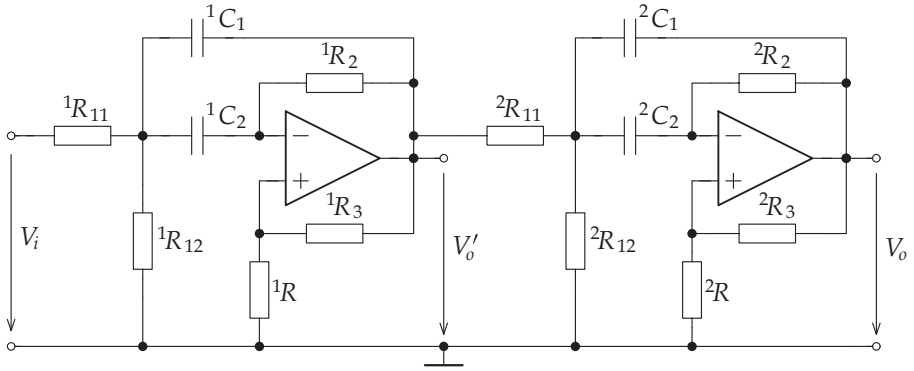


Fig. 4. Two-stage ARC band-pass filter.

Filter transfer functions are defined as follows

$$P_1 = \frac{V_o'}{V_i}, P_2 = \frac{V_o}{V_i}. \quad (14), (15)$$

The magnitude frequency response $|P_1(f)|$ is denoted $M_1(f)$, in the same way $M_2(f)$ corresponds to $|P_2(f)|$. The symbols $P_1(f)$, $P_2(f)$, $M_1(f)$, and $M_2(f)$ are for the filter with ideal components. When nonideal components are used, these symbols are changed to $P_{1N}(f)$, $P_{2N}(f)$, $M_{1N}(f)$, and $M_{2N}(f)$.

The circuit (transfer function $P_2(f)$) should implement a band-pass filter with a magnitude filter specification presented in Fig. 7.

4.2.3 Description of optimization

During the optimization, these nonidealities were respected:

- finite unity-gain bandwidth of operational amplifiers,
- finite voltage gain of operational amplifiers.

The model in Fig. 5 was used for the operational amplifiers in the filter (Sedra & Smith, 2004). The input resistance R_{IN} and output resistance R_{OUT} were not considered (see section 4.2.1). The value of transadmittance g is 1 S and the value of voltage gain a is 1.

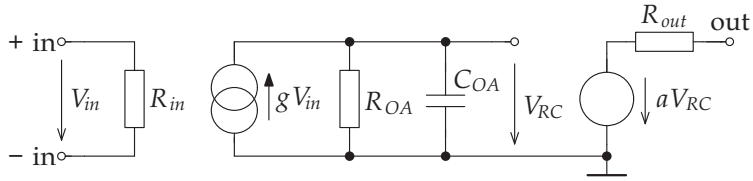


Fig. 5. Applied model of operational amplifier.

The value of the resistor R_{OA} and the capacitor C_{OA} depends on the unity-gain bandwidth B_1 and the voltage gain A_0 of the applied operational amplifier and they can be calculated according to the following formulae

$$R_{OA} = A_0,\ C_{OA} = \frac{\sqrt{A_0^2 - 1}}{2\pi B_1 A_0}. \qquad (16), (17)$$

In Fig. 6 there is the magnitude frequency response of the filter with component values designed for ideal filter. The parameter values of the operational amplifiers were the following: B_1 = 0.5 MHz, A_0 = 2·10^5. This figure shows the difference between magnitude frequency responses with using ideal and real components.

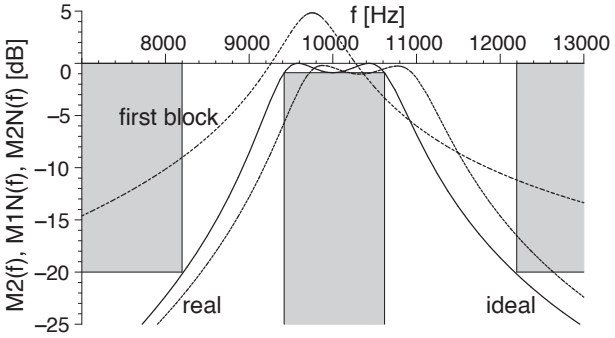


Fig. 6. Magnitude frequency responses of the ARC filter with designed component values for ideal components, solid line: magnitude M_2, for real components, dotted line: magnitude M_{1N} and dashed line: magnitude M_{2N}.

An optimization was applied to remove the difference between the magnitude frequency responses M_2 and M_{2N} and correct dynamic conditions so that max|M_{1N}(f)| = max|M_{2N}(f)| The result of optimization should be:

- The magnitude frequency response M_{2N} should satisfy a determined magnitude filter specification in Fig. 7.
- The magnitude frequency responses M_{1N} and M_{2N} should have their maximum values as similar as possible (because of obtaining an optimal dynamic range).

Suitable values for the resistors and capacitors in the circuit have to be found to meet the requirements.

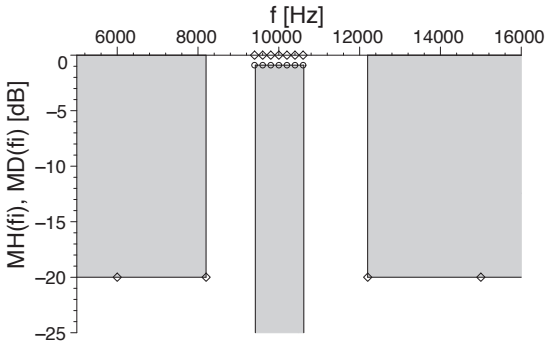


Fig. 7. Magnitude filter specification for optimization, diamonds: upper bounds $M_H(f_i)$ of magnitude ranges, circles: lower bounds $M_D(f_i)$ of magnitude ranges.

The DE was applied with the similar parameters, as in the previous example. Number of optimized components values was n = 12, (N = 120). Values of both resistors R (for both filter section) were chosen to ${}^{1}R = {}^{2}R$ = 10 kΩ.
The OF was created by adding a term U_M to (9) in order to express the dynamic conditions optimization, where ΔM_{max} is the permitted difference of the magnitudes maxima (10^{-6}).

$$U_M({}^{i}R_x, {}^{i}C_y) = \begin{cases} |\max(M_{1N}(f)) - 1| + |\max(M_{2N}(f)) - 1| \\ \quad \text{if } |\max(M_{1N}(f)) - 1| + |\max(M_{2N}(f)) - 1| > \Delta M_{max}, \\ 0 \quad \text{else,} \end{cases} \tag{18}$$

where ${}^{i}R_x$ and ${}^{i}C_y$ represent all resistor and capacitor values of the filter form Fig. 4.

4.2.4 Result from optimization

The optimization found the component values shown in Table 2. To find them, the optimization required 4831 generations. The magnitude frequency responses corresponding to these values (and chosen values of ${}^{1}R = {}^{2}R$ = 10 kΩ) are plotted in Fig. 8.

Stage	R_2 [kΩ]	R_3 [kΩ]	R_{11} [kΩ]	R_{12} [Ω]	C_1 [nF]	C_2 [nF]
First	11.18	78.47	38.38	558.5	7.963	6.370
Second	8.620	91.52	13.96	630.9	6.125	7.994

Table 2. Component values from optimization.

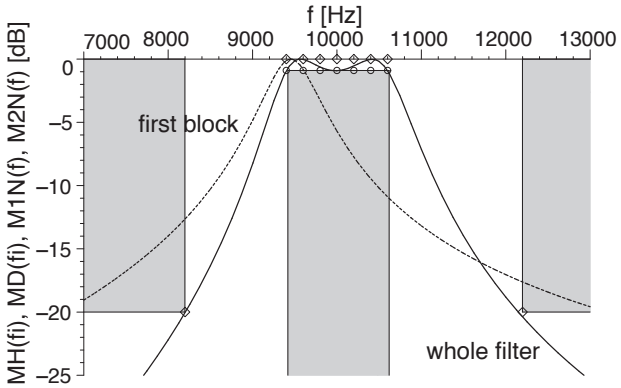


Fig. 8. Magnitude frequency responses after optimization, dotted line: magnitude M_{1N}, solid line: magnitude M_{2N}.

5. Examples of optimization of analogue discrete-working circuits

Analogue discrete-working circuits represent a group of electronic circuits utilized for circuit implementation nowadays. Two kinds of techniques belong in this group:

- switched-capacitor technique,
- switched-current technique.

However, the analysis of analogue discrete-working circuits is more complicated than in case of classical (continuous-working) circuits due to the discrete character of their operation (Bičák & Hospodka, 2003; Bičák & Hospodka, 2005). Consequently, the optimization of this kind of circuits is more difficult compared to continuously working ones (since their analysis is a necessary part of optimization).
Two optimizations are included in this section:

- optimization of a switched-capacitor filter,
- optimization of a switched-current filter.

5.1 Optimization of switched-capacitor filter

5.1.1 Introduction

One of common methods for circuit realization, integrated circuits in particular, is the switched-capacitor (SC) technique. This technique is widespread because it has a few advantages in comparison with other techniques (Ananda et al., 1995), for instance:

- The transfer of SC circuits depends not on capacitor values, but on the ratios of them. These ratios can be substantially more accurate than the capacitor values.
- A clock frequency signal f_C, which is needed for SC circuit operation, can be used for their tuning.
- SC circuits do not require resistors, whose implementation is difficult in integrated form.

As the switches in SC circuits, field effect transistors are commonly used (Ananda et al., 1995). However, this switch implementation has several nonidealities:

- nonzero off-state conductance,
- nonzero on-state resistance,
- parasitic capacitances.

From the mentioned switch nonidealities, one can say that nonzero on-state resistance R_{ON} shows itself mostly. Its effect on the transfer function of a SC circuit consists in charging a capacitor C in the circuit via this resistance. Therefore, the time constant of the charging $\tau = R_{ON}C$ is not zero as in case of an ideal switch with zero on-state resistance. The higher on-state resistance is, the higher ratio τ/T_C is and the more expressively the nonzero on-state resistance shows itself – the stronger effect of on-state resistance on the SC circuit behaviour is; T_C is the period of a clock frequency f_C, $T_C = 1/f_C$.
Another nonideality that can occur in SC circuits is the effect of the features of real operational amplifiers. These nonidealities have been discussed in section 4.2.1.
Both the nonidealities of switches and operational amplifiers affect the transfer function of SC circuits negatively.
The number of publications dealing with the optimization of SC circuits is not large, e.g., (Dolívka & Hospodka, 2006; Dolívka & Hospodka, 2007 b; Dolívka & Hospodka, 2008; Storn, 1996 a).

5.1.2 Optimized circuit and its required parameters

The SC circuit chosen for the optimization was an SC biquad (biquadratic section) with schematic diagram in Fig. 9 (Bičák & Hospodka, 2005). The symbols ϕ_1 and ϕ_2 stand for phase 1 and phase 2, respectively.

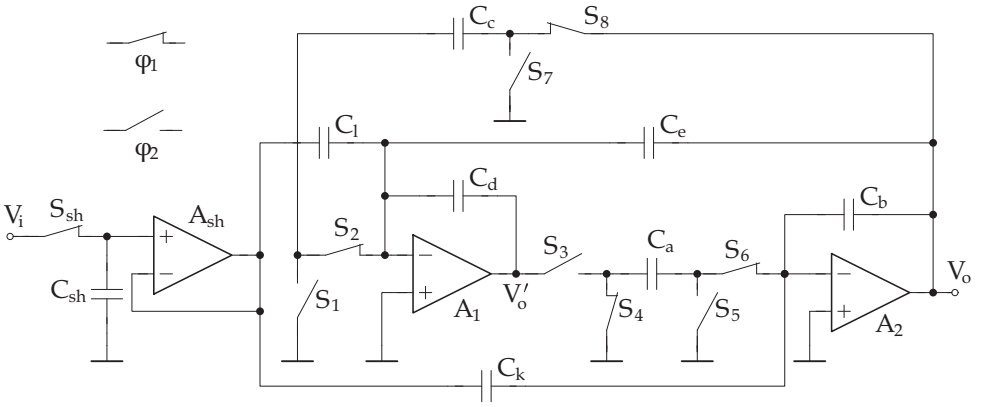


Fig. 9. Filter implemented in the switched-capacitor technique.

The transfer function from the input V_i to the output V_o' is denoted P_1 and the transfer function from the input V_i to the output V_o is denoted P_2. This transfer function labelling is for the filter with ideal components. Hence, the following equations are valid for P_1 and P_2

$$P_1 = \frac{V_o'}{V_i}, \quad P_2 = \frac{V_o}{V_i}. \tag{19), (20}$$

The transfer functions of the filter with both ideal and nonideal components are considered from phase 1 on the input to phase 1 on the output.

The magnitude of a transfer $P(z)$ is symbolized by $M(f)$. In case of the transfers $P_1(z)$ and $P_2(z)$, the magnitudes are calculated as follows

$$M_1(f) = \left| P_1\left(e^{j\frac{2\pi f}{f_C}} \right) \right|, \quad M_2(f) = \left| P_2\left(e^{j\frac{2\pi f}{f_C}} \right) \right|. \tag{21), (22}$$

For the transfer functions of the filter with nonideal components, labelling P_{1N} and P_{2N} is used instead of P_1 and P_2, respectively. The magnitudes of the transfer functions $P_{1N}(z)$ and $P_{2N}(z)$ are denoted $M_{1N}(f)$ and $M_{2N}(f)$, respectively. They are calculated from P_{1N} and P_{2N} in the same way as in case of M_1 and M_2 – according to (21) and (22).

Four kinds of filters can be implemented by this filter: low-pass, high-pass, band-pass, and notch filter. From these types, the band-pass filter was chosen. The filter was required to have these parameters:

- centre frequency: f_O = 400 kHz,
- clock frequency: f_C = 6 MHz,
- gain at f_O: G_O= 20 dB,
- quality factor: Q = 10,
- transfer function implemented from the input V_i to the output V_o.

If the filter with ideal components is designed according to a common method (Ananda et al., 1995), it has magnitude frequency responses depicted in Fig. 10.

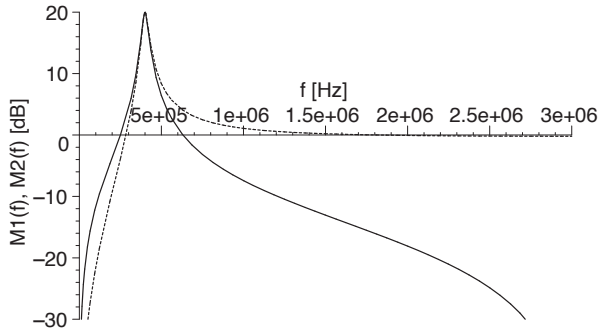


Fig. 10. Magnitude frequency responses of the SC filter with ideal components, dotted line: magnitude M_1, solid line: magnitude M_2.

5.1.3 Description of optimization

These nonidealities were taken into account during the optimization because their effect on SC filter characteristics is the most relevant:

- nonzero on-state resistance of the switches,
- finite unity-gain bandwidth of operational amplifiers,
- finite voltage gain of operational amplifiers.

The model in Fig. 5 was used for the operational amplifiers in the filter. Fig. 11 shows the magnitude frequency response of the filter with capacitor values designed for ideal components. The filter was analyzed with ideal and real components. The used value of switch on-state resistance was 1 kΩ. The parameter values of the operational amplifiers were the following: R_{IN} = 1 TΩ, R_{OUT} = 50 Ω, B_1 = 20 MHz, $A_0 = 2 \cdot 10^5$. From this figure, one can see that the magnitude frequency responses are different.

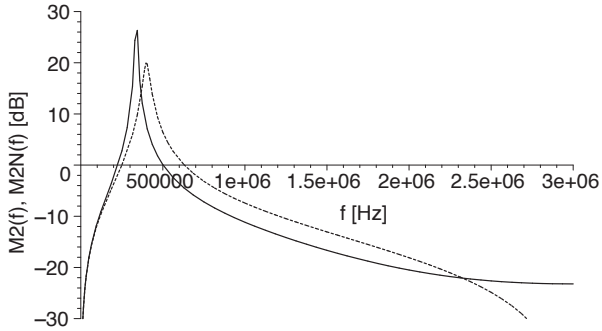


Fig. 11. Magnitude frequency responses of the SC filter with capacitor values designed for ideal components, dotted line: magnitude M_2, solid line: magnitude M_{2N}.

The effect of the three chosen nonidealities on the magnitude frequency response of the SC circuit was eliminated using optimization. The optimization had the following aims, which shall have been satisfied by finding suitable capacitor values:

- The magnitude frequency response M_{2N} should fulfil a defined magnitude filter specification (see Fig. 12), which was derived from the magnitude frequency response M_2.

- The magnitude frequency responses M_{1N} and M_{2N} should have their maximum values as similar as possible (because of obtaining an optimal dynamic range).
- The optimized filter should be stable in order to be applicable. (However, this aim is evident.) The condition of the stability is well known – all the poles of the transfer function in the z plane have to have the absolute value lower than 1.

The spread of capacitor values was not considered in the optimization. Nevertheless, the obtained capacitor values (see Table 3) have a spread, which is acceptable.

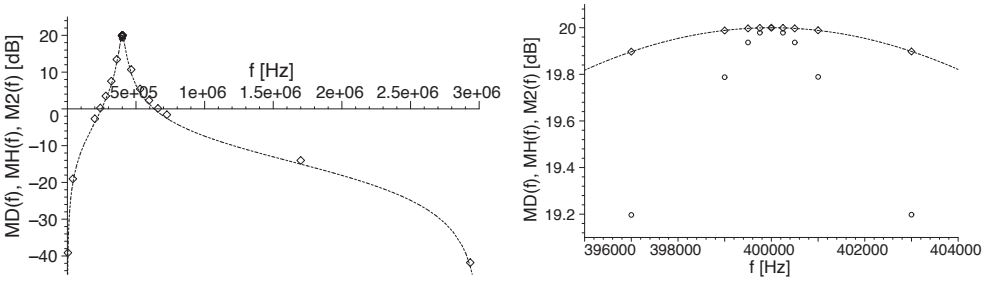


Fig. 12. Magnitude filter specification for optimization, circles: lower bounds $M_D(f_i)$ of magnitude ranges, diamonds: upper bounds $M_H(f_i)$ of magnitude ranges, dotted line: magnitude frequency response M_I. On the right: Detail for a vicinity of the frequency f_O.

The DE was applied as the optimization algorithm in this example. Most probably, this example could not be solved analytically. The parameters of the optimization process were as follows:

- $CR = 0.9$, $F = 0.5$,
- $d = 9$, $h = 23$,
- $N = 70$, $n = 7$ (= the number of all the capacitors),
- S_1 to S_7 = **R**,
- X_1 to $X_7 = \langle 10^{-12}, 10^{-10} \rangle$ – the values of all the capacitances can be from 1 pF to 100 pF.

The optimized function O is the magnitude M_{2N}. It has a very long form. Therefore, it is not presented here. The meaning of the variables of the functions O is as follows:

- x_0 represents frequency f, w_i is substituted by f_i,
- x_1 to x_7 represent C_1 to C_7.

The form of the OF was a modification of (9) – a term for an optimization of a dynamic range was added to (9) and a penalization function expressing the requirement of the circuit stability was included into (9). The resulting OF was

$$U(C_1,\ldots,C_7) = \begin{cases} \sum_{i=1}^{d} U_{Di}(C_1,\ldots,C_7) + \sum_{i=1}^{h} U_{Hi}(C_1,\ldots,C_7) + \left|\max M_{1N}(C_1,\ldots,C_7) - M_{2Nmax}\right| \\ \qquad \text{if the biquad is stable,} \\ 1000 \quad \text{if the biquad is unstable,} \end{cases} \tag{23}$$

where max M_{1N} means the maximal value of the magnitude M_{1N} and M_{2Nmax} means the required maximal value of the magnitude M_{2N}, $M_{2Nmax} = 10$ (= 20 dB = G_O).

The optimization was performed while using the values of the real components listed above. However, the parameters R_{IN} and R_{OUT} in the model of the operational amplifiers were not

used (so $R_{IN} = \infty\ \Omega$ and $R_{OUT} = 0\ \Omega$). The input and output resistances of the operational amplifiers were neglected because of simplifying the analysis of the filter during the optimization and thereby speeding it up. This simplification was done since their effect was supposed to be not significant. After the optimization, the analyses of the filter with the input and output resistances and without them were carried out and these analyses confirmed this assumption. The difference between these analyses can be seen in Fig. 14.

5.1.4 Result from optimization

The optimization reached the value of the OF of 0.000034 during 1523 generations. Using more generations did not improve the OF value (the total number of generations was 4000). Table 3 shows the capacitor values arisen from the optimization.

i	a	b	c	d	e	k	l
Ci [pF]	65.319	43.877	69.006	58.628	13.450	4.1664	52.682

Table 3. Capacitor values from optimization.

The magnitude frequency responses M_{1N} and M_{2N} with using the resulting capacitor values are shown in Fig. 13. The magnitude frequency response M_2 is also shown in this figure for comparing. The magnitudes M_{1N} and M_{2N} have their maxima on almost the same level; the difference between them is only about 0.00025 dB. The difference between the frequencies of the maxima occurs even in case of the magnitudes M_1 and M_2.

The magnitudes M_{1N} and M_{2N} plotted in Fig. 13 are with $R_{IN} = \infty\ \Omega$ and $R_{OUT} = 0\ \Omega$ – these parameters were not considered since the optimization was carried out without them (see section 5.1.3). However, their effect on the filter magnitude frequency responses is not significant. The magnitude frequency responses M_{1N} and M_{2N} with using the R_{IN} and R_{OUT} in the model of the operational amplifiers are denoted M_{1NR} and M_{2NR}, respectively.

In Fig. 14 and 15, there are the difference between the magnitude frequency responses M_{1NR} and M_{1N} and the difference between M_{2NR} and M_{2N}. It is apparent from the figure that the differences are not high. Higher values (but not too high) are only in the stop-band of the difference between M_{2NR} and M_{2N}.

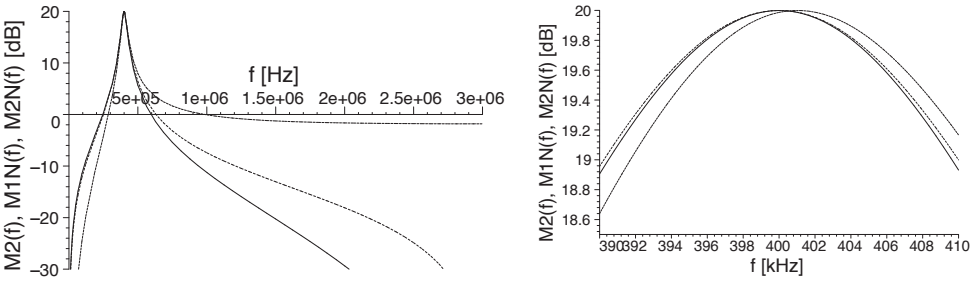


Fig. 13. Magnitude frequency responses of the SC filter, dotted line: magnitude M_2, dashed line: magnitude M_{1N} after optimization, solid line: magnitude M_{2N} after optimization. On the right: Detail for a vicinity of the frequency f_O.

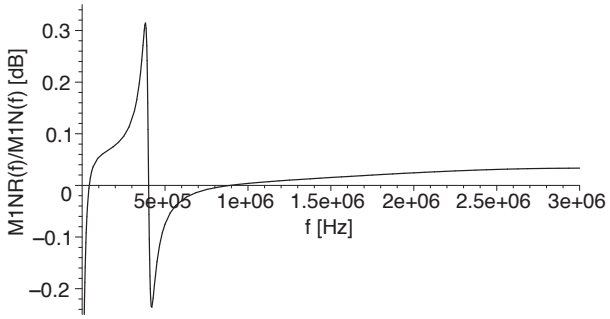


Fig. 14. Difference between magnitude frequency responses M_{1NR} and M_{1N}.

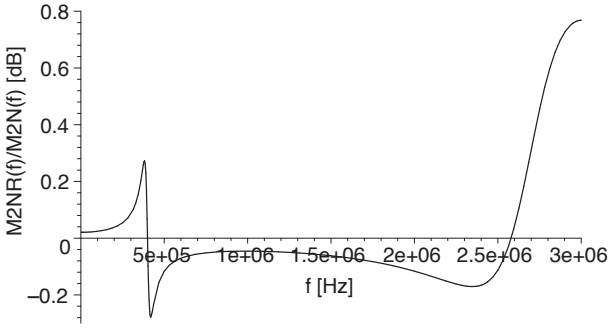


Fig. 15. Difference between magnitude frequency responses M_{2NR} and M_{2N}.

Of course, the optimization could be accomplished including the parameters R_{IN} and R_{OUT} but it would take a longer time than without them (about three times).

5.2 Optimization of switched-current filter

5.2.1 Introduction

The switched-current (SI) technique is applied commonly for the implementation of functional blocks because it has a few advantages (Toumazou et al., 1993):

- Suitable for the integrated form of circuits with utilizing VLSI-CMOS technology, i.e., possible integration of SI circuits with digital ones.
- Operation in the current mode – a high dynamic and frequency range.
- The transfer function of SI circuits depends on the ratios of the transconductances g_m of the transistors in individual circuit stages, not on the transconductance itself. The ratios can be substantially more accurate than the transconductances.
- No need of floating capacitors, required grounded ones only.
- Capacitor values do not affect the transfer function.

The nonidealities that can occur in SI circuits are especially related to the used switches and transistors. The transistors in SI circuits operate as controlled current sources. Their main nonideal features are finite output resistance and parasitic capacitances. In case of the switches, the nonideal features are these: nonzero on-state resistance, finite off-state resistance, and parasitic capacitances. These nonidealities are caused by the implementation of the switches by field-effect transistors.

Other properties of circuits realized by the SI technique that can be optimized to achieve a better circuit design are the sum of all transconductance values and the ratio between the highest and lowest transconductance value. These parameters should be as small as possible. Minimizing the sum of all transconductance values is owing to a small area of the chip with the SI circuit. Transconductance g_m of a transistor is linearly dependent on the ratio W/L, where W is the width of the transistor channel on the chip and L is its length. The minimal value of the length is limited by the used technology for circuit implementing. The transistor area on the chip is dependent on $W \cdot L$. Thus, for a given length, the smaller transconductance (smaller width) is, the smaller area is occupied by the transistor. Moreover, a smaller width causes smaller parasitic capacitances. The ratio between the highest and lowest transconductance value – the spread of transconductance values – should be minimized because its lower value is more suitable for circuit design.
The optimization of SI circuits is described in few publications. Authors know only about (Erten et al., 1999), which describes the optimization of SI circuits by means of simulated annealing. Authors' publications about the optimization of SI circuits are (Dolívka & Hospodka, 2007 a; Dolívka & Hospodka, 2007 c; Dolívka & Hospodka, 2008).

5.2.2 Optimized circuit and its required parameters

The SI circuit used in this section was a filter working as a biquad (biquadratic section), whose schematic diagram is in Fig. 16 (Toumazou et al., 1993). The symbols ϕ_1 and ϕ_2 stand for phase 1 and phase 2, respectively. Every transistor T_i has transconductance g_{mi} and the ratio of the currents of any two current sources in the upper part of Fig. 16 is the same as the ratio of the transconductances of the transistors connected to these current sources. Only one of the current values a_iI has to be chosen so that the transistors are in the linear part of their output characteristic.

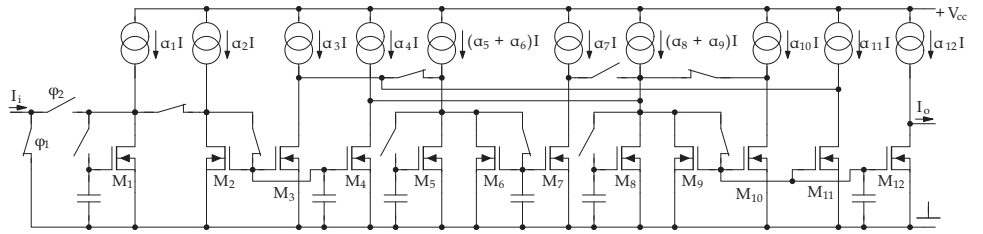


Fig. 16. Filter implemented in the switched-current technique.

The transfer function of the filter from the input I_i to the output I_o with using ideal components is denoted as P_I. All the filter transfer functions (with both ideal and nonideal components) in this section are considered from phase 2 on the input to phase 2 on the output. The transfer function P_I can be express according to (24) and the magnitude $M_I(f)$ of the transfer function $P_I(z)$ according to (25).

$$P_I = \frac{I_o}{I_i}, \quad M_I(f) = \left| P_I\left(e^{j\frac{2\pi f}{f_C}} \right) \right|. \tag{24), (25}$$

The transfer function of the filter using the nonideal components is symbolized P_N. The magnitude of this transfer is symbolized $M_N(f)$ and is related to the transfer $P_N(z)$ in the same manner to M_I – in accordance with (25).

This filter can realize a few filter types. A low-pass filter was chosen here. Its transfer function should have these parameters:

- pass-band cut-off frequency: f_P = 1 MHz,
- clock frequency: f_C = 10 MHz,
- quality factor: Q = 0.707,
- gain at 0 Hz: G_0 = 20 dB.

5.2.3 Description of optimization

Two nonidealities from those listed above were chosen for this optimization:

- finite output resistance of the transistors working as current sources,
- nonzero on-state resistance of the switches.

These nonideal features can be considered as the most important for these components.

To express the output resistance R_{OUT} of the transistors in the SI filter, the equivalent circuit in Fig. 17 (Sedra & Smith, 2004) was used for them. The output resistance R_{OUT} of the transistors is connected in parallel to the output resistance of the current sources in the upper part of Fig. 16 for alternating input current I_i. The value that was considered for this resulting resistance was 20 kΩ.

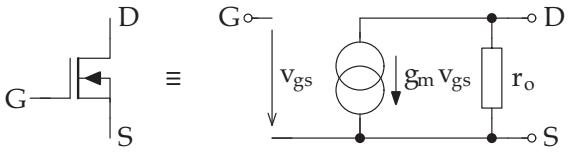


Fig. 17. Used equivalent circuit for transistors.

The nonzero on-state resistance of the switches was represented by a resistor connected in series to the ideal switch. The chosen value for the resistance was 1 kΩ.

Fig. 18 shows the magnitude frequency responses for the SI filter with transconductance values designed for ideal components. It is obvious from this figure that the difference between the magnitudes M_I and M_N is not small. Thus, the nonidealities affect the transfer function of the filter markedly.

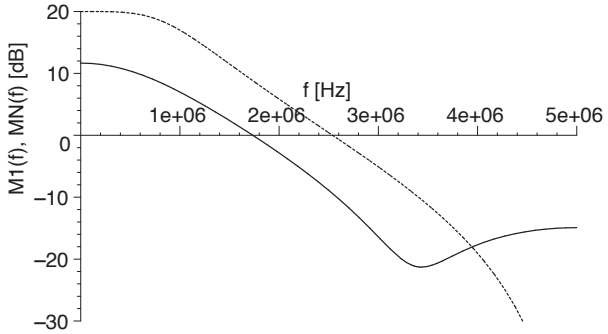


Fig. 18. Magnitude frequency responses of the SI filter with transconductance values designed for ideal components, dotted line: magnitude M_I, solid line: magnitude M_N.

To remove the undesirable effect of the nonidealities on the filter transfer function, new values of transconductances had to be found. This was made by optimization. In addition to this requirement, the optimization result had to meet two more aims. Hence, all the three requirements for this multi-objective (multi-criteria) optimization result were these:

- removing the undesirable effect of the nonidealities on the filter transfer function, i.e., achieving the shape of the magnitude frequency response M_N of the SI filter with the nonideal components as similar as possible to the magnitude frequency response M_I of the ideal filter,
- achieving the sum of all transconductance values as small as possible,
- achieving the ratio between the highest and lowest transconductance value as small as possible.

These requirements shall have been met by finding suitable transconductance values.
Because of the first aim of the optimization, a magnitude filter specification was defined – see Fig. 19.

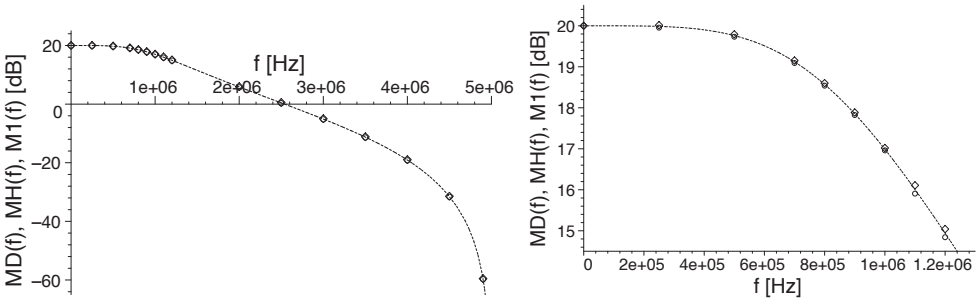


Fig. 19. Magnitude filter specification for optimization, circles: lower bounds $M_D(f_i)$ of magnitude ranges, diamonds: upper bounds $M_H(f_i)$ of magnitude ranges, dotted line: magnitude frequency response M_I. On the right: Detail for a frequency range of 0 to 1.3 MHz.

The DE was applied as the optimization algorithm in this example. Most probably, there is not any analytical method capable of accomplishing this optimization task. The parameters of the optimization process were as follows:

- $CR = 0.9$, $F = 0.5$,
- $d = 16$, $h = 16$,
- $N = 120$, $n = 12$ (= the number of all the transistor transconductances),
- S_1 to S_{12} = **R**,
- X_1 to $X_{12} = \langle 10^{-5}, 10^{-2} \rangle$ – the values of all the transconductances can be from 10 μS to 10 mS.

The optimized function O is the magnitude M_N. It has a very long form. Therefore, it is not presented here. The meaning of the variables of the functions O is as follows:

- x_0 represents frequency f, w_i is substituted by f_i,
- x_1 to x_{12} represent g_{m1} to g_{m12}.

To meet the second and third aim of the optimization, two terms were added to (9). This form of the OF was applied

$$U(g_{m1}, g_{m2}, \ldots, g_{m12}) = b_M \underbrace{\left(\sum_{i=1}^{d} U_{Di}(g_{m1}, g_{m2}, \ldots, g_{m12}) + \sum_{i=1}^{h} U_{Hi}(g_{m1}, g_{m2}, \ldots, g_{m12}) \right)}_{U_M} + \\ + b_S \underbrace{\sum_{i=1}^{n} g_{mi}}_{U_S} + b_R \underbrace{\frac{\max\limits_{i \in \{1,2,\ldots,n\}} (g_{mi})}{\min\limits_{i \in \{1,2,\ldots,n\}} (g_{mi})}}_{U_R}, \tag{26}$$

with this symbol meaning:

$\max\limits_{i \in \{1,2,\ldots,n\}} (g_{mi})$ the maximal value from all transconductance values,

$\min\limits_{i \in \{1,2,\ldots,n\}} (g_{mi})$ the minimal value from all transconductance values,

b_M the weight of the optimization of the magnitude frequency response,

U_M the rate of satisfying the magnitude filter specification by the optimized magnitude frequency response,

b_S the weight of the optimization of the sum of all transconductance values,

U_S the sum of all transconductance values,

b_R the weight of the optimization of the ratio between the highest and lowest transconductance value,

U_R the ratio between the highest and lowest transconductance value.

It is obvious from (26) that the value of the OF U is always higher than 0. Whereas the OF term U_M can be zero (which expresses that the optimized magnitude frequency response meets the magnitude filter specification), the term U_S is always higher than 0 and the term U_R is always higher than 1.

The values of the weights b_M, b_S, and b_R are presented in the following section.

5.2.4 Result from optimization

For a satisfactory optimization result, proper setting of the weights in the OF (26) is necessary. Table 4 presents various values of the weights and corresponding results of the optimization. One of the weights can be always equal to 1 because only the ratios between the weights are important for the optimization.

b_M	b_S	b_R	U_{1G}	U_M	U_S	U_R	U	ΔU_{3000G}
1	1	1	24.9	0.00668	0.0857	3.42	3.51	$4.28 \cdot 10^{-5}$
1	10	1	25.4	0.00376	0.0715	3.48	4.19	$6.62 \cdot 10^{-5}$
1	10	3	46.0	1.64	0.0795	2.76	10.7	$1.60 \cdot 10^{-3}$
1	30	1	26.7	0.349	0.0510	3.57	5.45	$2.11 \cdot 10^{-4}$
4	100	1	74.6	0.0960	0.0477	4.17	9.32	$6.04 \cdot 10^{-1}$
2	30	1	41.2	0.0194	0.0697	3.49	5.62	$1.02 \cdot 10^{-3}$

Table 4. The results of the optimization with various values of the weights b_M, b_S, and b_R in the OF, U_{1G}: the value of the OF after the 1st generation, UM, U_S, U_R, U: values after 4000 generations, ΔU_{3000G}: the improvement (i.e., lowering) of the OF value in the last 3000 generations, i.e., since the 1000th generation.

The last column in Table 4 shows that using more than 1000 generations in this optimization yields only a low improvement of the OF value.
From the values of the weights in Table 4, the most suitable ones are those on the last line because then the values of U_M, U_S, and U_R are optimized equally, so they are regarded as the optimization result here. However, of course it is possible to use another combination of the weight values from Table 4 to get a utilizable optimization result.
Table 5 shows the transconductance values arisen from the optimization with the weight values on the last line of Table 4, i.e., $b_M = 2$, $b_S = 30$, and $b_R = 1$.

i	1	2	3	4	5	6
g_{mi} [mS]	2.4417	2.4417	8.5222	2.4417	8.5222	6.3116

i	7	8	9	10	11	12
g_{mi} [mS]	8.5222	8.5222	4.8032	6.2199	2.4417	8.5222

Table 5. Transconductance values from optimization.

Because the value of U_M is not zero, the optimized magnitude frequency response does not quite fulfil the magnitude filter specification. The violations of the magnitude filter specification occur only at the following three frequencies:

- 0 Hz: required value: 20 dB, obtained value: 19.99999999995 dB,
- 4.5 MHz: required values: −31.640 to −31.440 dB, obtained value: −31.6400000007 dB,
- 4.9 MHz: required values: −59.737 to −59.537 dB, obtained value: −59.3704 dB.

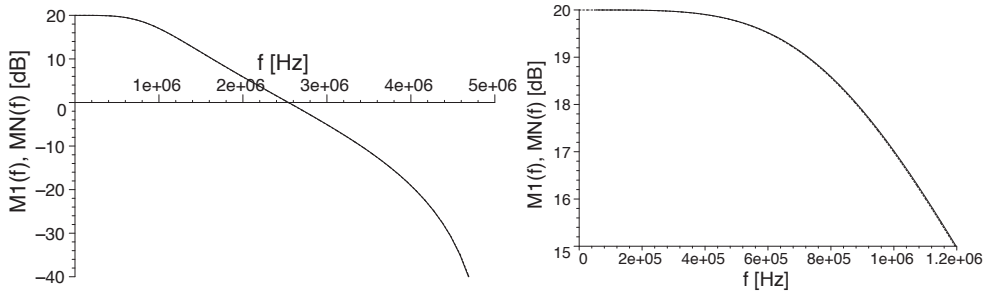


Fig. 20. Magnitude frequency responses of the SI filter, dotted line: magnitude M_I, solid line: magnitude M_N after optimization. On the right: Detail for a frequency range of 0 to 1.2 MHz.

It is apparent that these violations are slight.
In Fig. 20, there is the magnitude frequency response M_N with using the resulting transconductance values. The magnitude frequency response M_I is also shown in this figure for comparing. It is evident from the figure that the difference between the magnitudes M_N and M_I is very little.
If the terms U_S and U_R were not considered in the optimization (i.e., if the value of the weights b_S and b_R were 0), the obtained sum of all transconductance values would be 0.0737 and the ratio between the highest and lowest transconductance value would be 4.11. Hence,

both of them would be worse than in case of this optimization. Therefore, it is obvious that considering these two requirements in optimization leads to a better circuit design.

6. Comparing several optimization methods

This section shows the suitability of ten optimization methods for optimization of analogue electronic filters. The methods are compared as for their speed of optimization and ability to search for the global extreme, not a local one. The chosen methods are:

a. DE, version best/1/bin
b. DE, version best/1/bin combined with the simplex method
c. DE, version EDE
d. DE, version EDE combined with the simplex method
e. DE, version rand/1/bin
f. DE, version rand/1/bin combined with the simplex method
g. DE, version rand-to-best/1/bin
h. DE, version rand-to-best/1/bin combined with the simplex method
i. GA
j. GA combined with the simplex method

The version EDE of the DE differs from other versions of the DE in the way of generating a trial vector (Vondraš & Martinek, 2002).

The optimization task utilized for this comparing was similar to the one presented in section 5.1, so it is a common optimization of an analogue electronic filter.

The value of the applied OF is 0 if the optimization is successful. The optimization methods were required to achieve the OF value lower or equal to 10^{-10}. To get this result, the methods needed numbers of generations G_N, which are listed in Table 6, where the OF value U_{OBT} obtained by them is presented too.

Method	Without the simplex method		With the simplex method	
	G_N	U_{OBT}	G_N	U_{OBT}
DE, best/1/bin	94	$8.6 \cdot 10^{-11}$	60	$2.3 \cdot 10^{-11}$
DE, EDE	173	$4.4 \cdot 10^{-11}$	61	$6.1 \cdot 10^{-11}$
DE, rand/1/bin	>2000	$1.4 \cdot 10^{-9}$	114	$4.0 \cdot 10^{-11}$
DE, rand-to-best/1/bin	179	$9.1 \cdot 10^{-11}$	72	$4.1 \cdot 10^{-11}$
GA	>2000	$3.9 \cdot 10^{-2}$	>2000	$7.3 \cdot 10^{-10}$

Table 6. Results obtained by means of the chosen methods.

Fig. 21 illustrates the achieved value of the OF versus the number of generations. The speed of optimization of all the methods can be seen easily from it. Fig. 21 together with Table 6 shows the ability to search for the global extreme too. The method i could not find the global extreme but a local one only whereas the other methods except for e converged to the global extreme quickly. The method b turned out to be the best one.

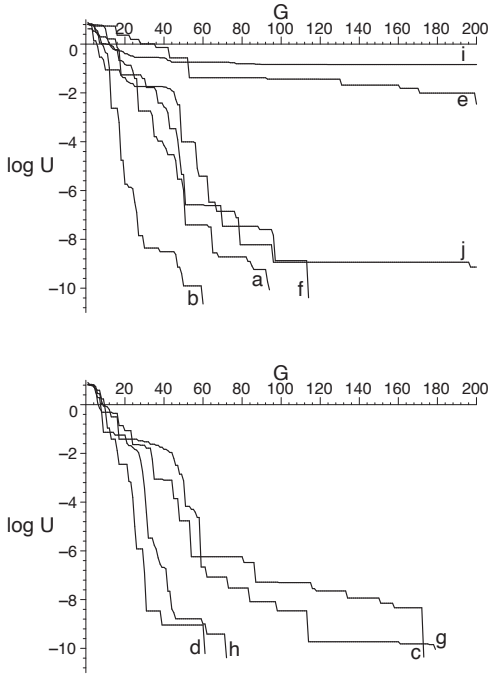


Fig. 21. Dependence of the value of the OF on the number of generations during optimization process performed by means of the chosen methods.

7. Conclusion

The aim of this chapter was to show possibilities of an optimization of analogue electronic filters by means of evolutionary algorithms. In practise, there are many cases when common analytical methods cannot be used for obtaining required circuit characteristics and/or eliminating nonideal circuit features. Evolutionary algorithms are very suitable for this purpose.

The ways to carry out the optimization of analogue electronic filters were described in the beginning of this chapter. Then a few examples of optimizations were presented to explain the description better. The optimized circuits were chosen from both analogue continuous-working ones and analogue discrete-working ones. Several evolutionary algorithms were compared regarding their efficiency while optimizing analogue electronic filters at the end of the chapter.

8. References

Ananda P. V. M.; Ramachandran V. & Swamy M. N. S. (1995). *Switched Capacitor Filters – Theory, Analysis and Design*, Prentice Hall International, ISBN 0-13-879818-4

Bičák J. & Hospodka J. (2003). Frequency response of switched circuits in SPICE, *Proceedings of conference ECCTD 2003*, pp. I/333–I/336, IEEE, ISBN 83-88309-95-1

Bičák J. & Hospodka J. (2005). PraSCAn – Maple package for analysis of real periodically switched circuits, *Proceedings of Maple Conference 2005*, pp. 8–18, Maplesoft, a division of Waterloo Maple Inc., Waterloo Ontario, ISBN 1-894511-85-9

Bičák J. & Hospodka J. (2008). PraCAn – Maple package for symbolic circuit analysis, *Proceedings of conference Digital Technologies*, EDIS Žilina University Publisher, Žilina, ISBN 978-80-8070-953-2

Brutovský B.; Ulicný J. & Miškovský P. (1995). Application of genetic algorithms based techniques in the theoretical analysis of molecular vibrations, *Proceedings of conference Mendel'95*, pp. 29–33, Brno, ISBN 80-214-0672-0

Cadence Design Systems, Inc.: http://www.cadence.com

Chambers L. D. (2000). *The Practical Handbook of Genetic Algorithms, Applications*, Chapman & Hall/CRC, ISBN 1-58488-240-9

Corne D.; Dorigo M. & Glover F. (editors). (1999). *New Ideas in Optimization*, McGraw-Hill, London, ISBN 0-07-709506-5

Dasgupta D. & Michalewicz Z. (1997). *Evolutionary Algorithms in Engineering Applications*, Springer-Verlag, Berlin, ISBN 3-540-62021-4

Dolívka L. & Hospodka J. (2006). Elimination of switch on-state resistance effect on a switched-capacitor filter characteristic, *Proceedings of conference PRIME 2006*, pp. 177–180, IEEE, ISBN 1-4244-0156-9

Dolívka L. & Hospodka J. (2007 a). Switched-current filter optimization based on evolutionary algorithms, *Proceedings of conference ECCTD 2007*, pp. 551–554, IEEE, ISBN 1-4244-1342-7

Dolívka L. & Hospodka J. (2007 b). Switched-capacitor filter optimization with respect to switch on-state resistance and features of real operational amplifiers, *Radioengineering*, vol. 16, no. 2, pp. 34 – 39, Brno, ISSN 1210-2512

Dolívka L. & Hospodka J. (2007 c). Using the differential evolution algorithm for the multi-objective optimization of a switched-current circuit, *Proceedings of conference CEC 2007*, pp. 1351–1358, IEEE, ISBN 1-4244-1340-0

Dolívka L. & Hospodka J. (2008). Ways to optimize analogue switched circuits, *Radioengineering*, vol. 17, no. 4, pp. 48–54, Brno, ISSN 1210-2512

Erten I. G.; Dundar G. & Balkir S. (1999). Optimization and synthesis of switched current filters with nonideal MOS transistors, *Proceedings of Third International Workshop on Design of Mixed-Mode Integrated Circuits and Applications 1999*, pp. 13–17, ISBN 0-7803-5588-1

Gielen G. G. E.; Walscharts H. C. C. & Sansen W. M. C. (1990). Analog circuit design optimization based on symbolic simulation and simulated annealing, *IEEE Journal of Solid-State Circuits*, vol. 25, no. 3, pp. 707–713, IEEE, ISSN 0018-9200

Goldberg D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co., Reading, Massachusetts, ISBN 0-201-15767-5

Haseyama M.; Aketa Y. & Kitajima H. (1996). A method quantizing filter coefficients with genetic algorithm and simulated annealing, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E79-A, no. 8, pp. 1130–1134, ISSN 0916-8508

MathWorks, Inc.: http://www.mathworks.com

Mentor Graphics Corp.: http://www.mentor.com

Nelder J. A. & Mead R. (1965). A simplex method for function minimization, *Computer Journal*, vol. 7, no. 4, pp. 308–313

Pintér J. D. (1996). *Global Optimization in Action*, Kluwer Academic Publishers, ISBN 0-7923-3757-3

Sedra A. S. & Smith K. C. (2004). *Microelectronic Circuits*, 5th edition, Oxford University Press, ISBN 0-19-514251-9

Schaumann R.; Ghausi M. S. & Laker, K. R. (1990). *Design of Analog filters,* Prentice-Hall, New Jersey 1990, ISBN 0-13-200288-4.

Smith M. *WinSpice User's Manual*. http://www.winspice.com

Storn R. (1996 a). System design by constraint adaptation and differential evolution, *Technical report TR-96-039*, ICSI

Storn R. (1996 b). Differential evolution design of an IIR-filter, *Proceedings of conference ICEC'96*, pp. 268–273, IEEE, ISBN 0-7803-2902-3

Storn R. & Price K. (1996). Minimizing the real functions of the ICEC'96 contest by differential evolution, *Proceedings of conference ICEC'96*, pp. 842–844, IEEE, ISBN 0-7803-2902-3

Storn R. & Price K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Kluwer Academic Publishers, ISSN 0925-5001

Tichá D. & Martinek P. (2005). OTA-C lowpass design using evolutionary algorithms, *Proceedings of conference ECCTD 2005*, vol. 2, pp. 197–200, University College Cork, ISBN 0-7803-9066-0

Toumazou C.; Hughes J. B. & Battersby N. C. (1993). *Switched-Currents: An Analogue Technique for Digital Technology*, Peter Peregrinus Ltd., London, ISBN 0-86341-294-7

Vondraš J. & Martinek P. (2002). A Design of Active Filters with Corrected Group Delay by Means of the EDE Method (in Czech), *Proceedings of NSSS VI.*, vol. 1, pp. 36–39, Military Academy, Liptovský Mikuláš, ISBN 80-8040-180-2

Waterloo Maple, Inc.: http://www.maplesoft.com

Žiška P. & Vrbata J. (2006). Method for design of analog group delay equalizers, *Proceedings of conference ISCAS 2006*, pp. 445–448, IEEE, ISBN 0-7803-9389-9

21

Evolutionary Optimization of Microwave Filters

Maria J. P. Dantas[1], Adson S. Rocha[3], Ciro Macedo[2], Leonardo da C. Brito[2], Paulo C. M. Machado[2] and Paulo H. P. de Carvalho[3]
[1]*Catholic University of Goiás, Goiânia, GO,*
[2]*Federal University of Goiás, Goiânia, GO,*
[3]*University of Brasília, Brasília, DF*
Brazil

1. Introduction

The optimization of circuits with the aim of improving performance, lowering costs, and more recently, to reduce the size and weight of electronic devices, among other objectives, has become a research field in the areas of mathematics and engineering around the world (Antoniou & Lu, 2007). The problem can be interpreted as: among all possible models for the circuit, considering topology and values of components, find a model with minimum size and appropriate parameters, able to meet a set of typically conflicting hard specifications.
The deterministic methods are not effective in the design of new circuits, since it does not always have available accurate mathematical models of the processes to be optimized (Levy et al., 2001). However, a number of new stochastic algorithms have allowed the development of nearly optimal circuits, which are quite acceptable from the standpoint of their implementations. The techniques currently used require much prior knowledge about the circuit to be optimized and this knowledge is not always available. The challenge, then, is the development of robust techniques that rely less in specialist knowledge or even incorporating this knowledge, and show efficiency equal or superior to the traditional methods.
In general, due to the complexity of the problems associated with different types of circuit designs, sub-optimal solutions are looked for in order to mitigate the mathematical complexity of the analytical models, generating approximate solutions, but with acceptable results (Levy et al., 2001). Evolutionary methods, so called because they use principles of evolution found in nature, are natural candidates to this task, since they are able to find optimal solutions or solutions near the optimum, using mechanisms of selection, crossover and mutation (Holland, 1970). Most methods in the literature consider an initial topology for the circuit and only optimize its parameters, such as, for example, in (Hsu & Huang, 2005). However, it is desirable that the topological structure also goes under optimization, allowing, among other things, the search for new topologies, which is a requirement of current applications. Other methods, using the traditional Genetic Programming as in the work of Koza (Koza et al., 1996), optimize the topology, without considering any prior knowledge, but demand a high computational cost and also have some methodological drawbacks such as premature convergence and stagnation, among others (Hu & Rosemberg, 2004).

This chapter discusses the application of evolutionary methods to design RF/microwave filters, making comparisons with traditional methods and other evolutionary methodologies proposed in the literature for the same class of problems. The design of analog circuits will be addressed as a problem of simultaneous optimization of topology and parameters, and we will present a general evolutionary method for synthesis of analog circuits, whose application produces competitive solutions when compared with other methods already proposed in the literature.

2. Description of the method

Fig. 1 shows the flowchart of the Memetic Evolutionary Algorithm. This hybrid (memetic) method associates a local search technique widely explored in the literature, the Simulated Annealing algorithm (Michalevicz & Fogel, 2004), to the global search Evolutionary Algorithm. In the following sections, the features of the method will be described.

2.1 Two-port circuit representation

Two-port circuit representation is widely used to describe microwave circuits, from passive to active elements, such as transistors. So, it becomes necessary to represent the evolvable circuit shown in Fig. 2 by two-port sub-circuits. Thus, the circuit is encoded into the hereinafter called Positional Matrix.

The initial size of this evolvable matrix is given by the number of elements of the initial circuit, which is composed by a set of elements cascaded from the source towards the load. The build-up process of the initial Positional Matrix is as follows:

1. Randomly define the size n of the Positional Matrix;
2. Randomly select n circuit elements from a database and assemble the initial Positional Matrix by cascading them into its main diagonal, from the source to the load;
3. Repeat m times

Randomly select a row i and a column j of the Positional Matrix, subject to $i \leq j$. Randomly select a circuit building-block and its respective connection type based on constraining rules and encode it in the position (i, j) of the Positional Matrix (e.g., as depicted in Fig. 4).

Along the evolution process, other building blocks are placed into or removed from the variable-size Positional Matrix, as described in the following sections. This specific feature of this approach gives degrees of freedom to the process of suitable topology discovery. The possible types of connection are: cascade, serial, parallel, or hybrid. Besides the building-blocks, topology constraining rules also feeds the algorithm, as depicted by the database in Fig. 3.

Figs. 4 and 5 illustrate the proposed representation. In the Positional Matrix of Fig. 4, the circuit elements (building-blocks) $b_{11}^{(1)}$, $b_{22}^{(1)}$, and $b_{33}^{(1)}$ are connected in cascade; $b_{11}^{(2)}$ and $b_{11}^{(1)}$, are connected in parallel (p); $b_{13}^{(1)}$ is in series (s) with the input port of $b_{11}^{(1)}$ and in parallel (p) with the output port of $b_{33}^{(1)}$ (hybrid connection); $b_{23}^{(1)}$ is in parallel (p) with the input port of $b_{22}^{(1)}$ e and in series with the output port of $b_{33}^{(1)}$; (hybrid connection). Fig. 5 shows its respective topological representation.

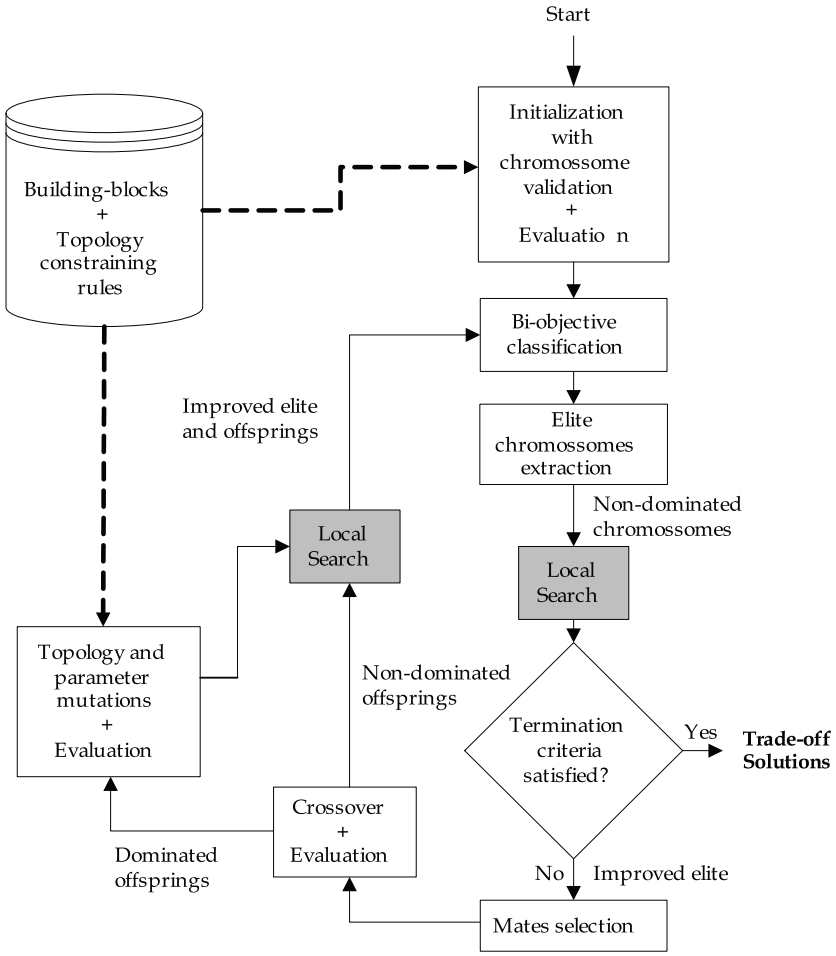


Fig. 1. The Bi-objective Memetic Evolutionary Algorithm.

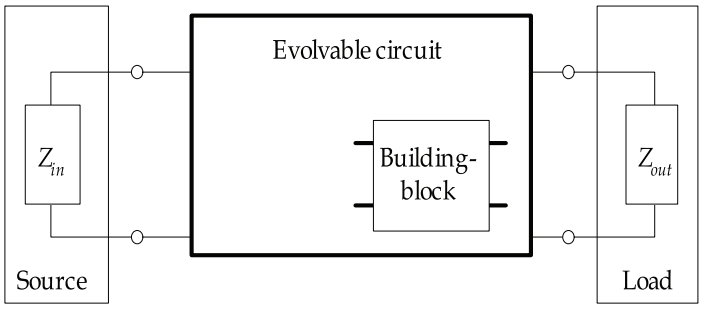


Fig. 2. Template of the evolvable circuit.

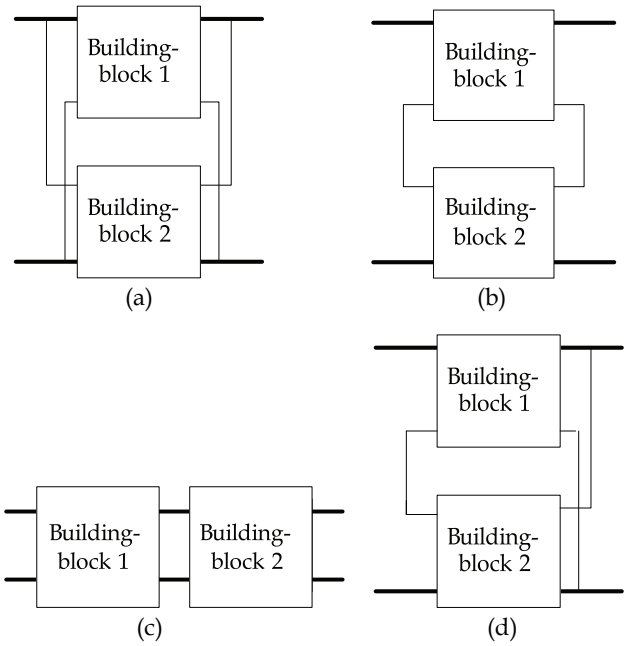


Fig. 3. The possible connections types between two-port building blocks: (a) parallel, (b) serial, (c) cascade, and (d) hybrid (serial connection in the input port and parallel in the output port).

Source	(1,1) $b_{11}^{(2)} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ (p) $b_{11}^{(1)} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ (c)	(1,2)	(1,3) $b_{13}^{(1)} = \begin{bmatrix} 2 & 7 \\ 10 & 8 \end{bmatrix}$ (h) - s/p	Load
	(2,1)	(2,2) $b_{22}^{(1)} = \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix}$ (c)	(2,3) $b_{23}^{(1)} = \begin{bmatrix} 3 & 8 \\ 4 & 9 \end{bmatrix}$ (h) - p/s	
$\begin{bmatrix} 1 \\ 10 \end{bmatrix}$	(3,1)	(3,2)	(3,3) $b_{33}^{(1)} = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$ (c)	$\begin{bmatrix} 7 \\ 9 \end{bmatrix}$

Fig. 4. Example of Positional Matrix (size 3×3).

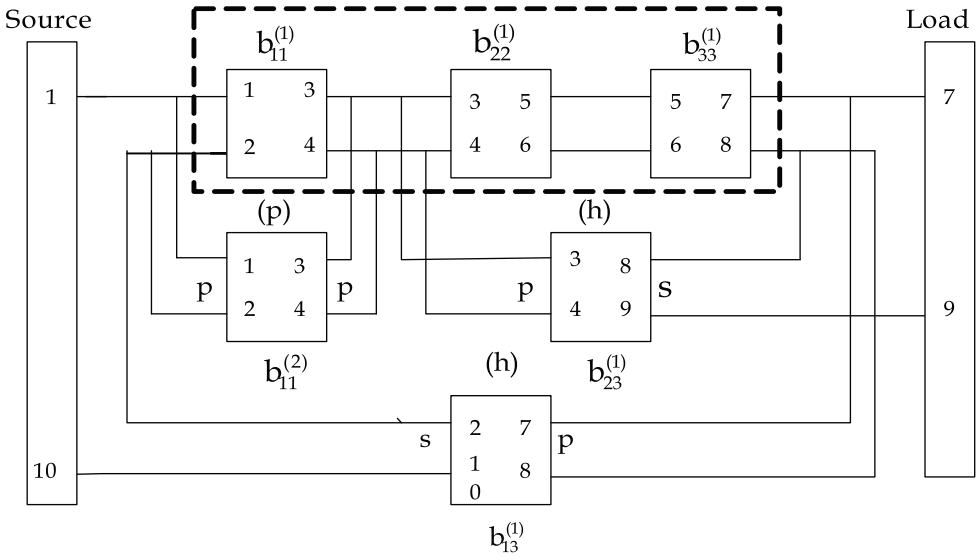


Fig. 5. Topological representation.

Note that the matrix is handled directly by the algorithm without the need of a one-dimensional equivalent representation. In (Mesquita et al., 2002) a matrix representation for circuit synthesis is proposed but the mapping bi to one-dimensional is performed before the crossover operator is applied. However, in (Im et al., 2003), it is shown that this procedure losses neighbor structures, which is fundamental for the good behavior of the evolution process. In the representation presented in this chapter, the Positional Matrix is always associated with valid circuits, what does not occur in other methods, with always present the generation of anomalous circuits in each generation. In the representation proposed by (Mesquita et al., 2002) the percentual of anomalous circuits is about 5% , but the author says that this percentage can reach 80%, which is undesirable.

2.2 Database entries

The presented algorithm uses an approach with combinations of building-block and topology constraining rules, as used in (Dastidar et. al, 2005, Lai & Jeng, 2006). This allows the use of expert knowledge to reduce the search space, avoiding anomalous circuits and producing structured circuits. The building-blocks are structures known in the literature or can be defined by the user. The rules allow topological constraints. For example, the user may allow only the occurrence of inline topologies. He may set the maximum number of connections between the circuit blocks (between source and load or between other circuit elements), the types of integration (serial, parallel, cascade, mixed), and the types of connections between blocks of circuits (direct, cross), among others. These constraints are used to compose the initial population and to accept or reject a new circuit composed during the evolution process. This feature significantly improves the quality of initial population, which represents an early memory stage. This adds expert knowledge, which somehow also contributes for reducing the search space, and still avoiding anomalous circuits at this stage. Fig. 6 illustrates the process.

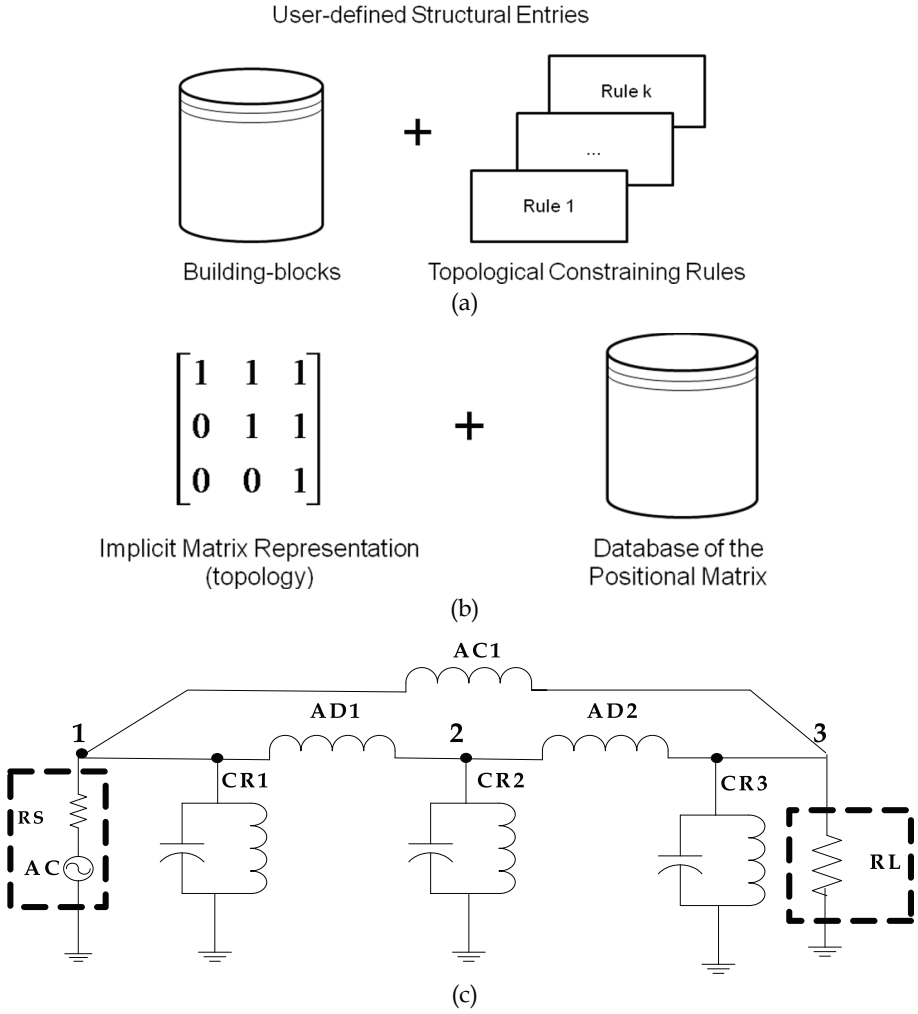


Fig. 6. Structural entries: (a) specialist database, (b) representation of the solution, and (c) a circuit composed from the structural entries.

2.3 Population initialization

The population is initialized with circuits composed by building-blocks, randomly selected from the database previously mentioned. They are connected according to the Positional Matrix previously completed, using the rules of topological constraints. Initially, only a building-block is connected to a possible pair of nodes. At the same time a building-block is selected, values are assigned to its parameters. This selection is random and held in a range of values predetermined by the user. The rules of structural constraints can be set by the user, if he has expert knowledge. Once defined, the types of connections allowed between the circuit elements are established: connections in series, parallel, cascade, mixed, number of paths between source and load, n

umber of paths between parts of the circuit, if the source and the load will be directly coupled, what types of coupling that are allowed, etc. The definition of coupling rules can improve the quality of the initial population. In case the user does not have this knowledge, the algorithm performs a default initialization. It generates a matrix associated with a valid circuit of arbitrary size - for this, all entries are filled with high probability. Soon after, the algorithm checks whether the generated matrix corresponds to a connected circuit and, if it is wrong, a repairing procedure is performed.

2.4 Evaluation functions

In this method, two objective-functions are defined in order to allow a trade-off relation: (a) the circuit performance, which is evaluated using a frequency-domain circuit simulation method; and (b) the circuit size, given by the number of two-port building-blocks. The circuit simulator computes the frequency responses (the scattering parameters) over a set of frequencies uniformly distributed in the range defined by the user. Then, the algorithm calculates the sum of the squared deviations between the computed aggregate responses and the desired responses (sum squared error), as in (1). The desired response is provided as scattering parameters masks for the absolute values of the transmission coefficient $|S_{21}|$ and reflection coefficient $|S_{11}|$, given in dB.

$$SSE = \sum_{j=1}^{k} \left[\left(\left| S_{21}(f_j) \right| - \left| S_{21}^{*}(f_j) \right| \right)^2 + \left(\left| S_{11}(f_j) \right| - \left| S_{11}^{*}(f_j) \right| \right)^2 \right] \quad (1)$$

In (1), k is the number of evaluation frequencies, $\left|S_{21}^{*}(f_j)\right|$ is the constraining value of the response mask of the respective scattering parameter at f_j. The difference $\left|S_{21}(f_j)\right| - \left|S_{21}^{*}(f_j)\right|$ in (1) is set to zero if the mask is not violated by the value $\left|S_{21}(f_j)\right|$. The same criterion is applied to $|S_{11}|$.

2.5 Evolution schemes

The population is randomly initialized with circuits (individuals or chromosomes) that use the template circuit shown in Fig. 2, which is composed by two-port circuit elements (genes) randomly selected from the previously defined database. In order to generate high-performance small circuits, a bi-objective selection approach – the crowded-comparison operator, extracted from the NSGA-II (Deb et al., 2002) – is applied in this method at two points: to extract the elite (non-dominated chromosomes) of the current population, as well as the elite of the offspring. The two evaluation functions (objective-functions) previously defined are taken into account. The elite individuals of the population, i.e. the Pareto front, are found by applying this classification method. The selection scheme used here is the well-known binary tournament method (Michalewicz & Fogel, 2004).

The proposed approach provides the balance between performance and size of the solutions, and, consequently, makes it possible to naturally reduce the tendency of the process for producing larger circuits as the population evolves. Additionally, it allows the extraction of new building-blocks (as desired concerning the building-block hypothesis (Goldberg, 1989)) derived by the evolution process. They can be used in the next stage to produce the competitive circuits with some degree of structural redundancy.

A local search process assists the Evolutionary Algorithm for fitness improvement of candidate circuits, refining their parameters in order to avoid good topologies with non-

optimized parameter values to be prematurely discarded. The evaluation criterion to accept new parameters for a given topology is mono-objective, based on the performance function (1). This process takes place in two points of the evolution cycle. After the classification process, the local search method is applied to each elite individual. Also, the local search procedure is carried out after the crossover/mutation procedure. Doing so, the topology space is explored and, subsequently, the parameters of the new topologies are improved. As a result, offspring solutions will be able to fairly compete with the current elite set for composing the elite of the next generation. The Simulated Annealing technique (Michalewicz & Fogel, 2004) with few iterations and predefined temperature values was adopted. Only a low computational effort is necessary for each local search (circuit parameters tuning).

2.6 Bidimensional topology crossover operator

Only one crossover operator is proposed. Fig. 7 sketches this operator. Each crossover operation generates only one offspring. The crossover occurs as follows.

1. Given two reduced matrices, a cut point in parent matrix 1 is randomly chosen, such that four regions are defined, as shown in Fig 7(a).
2. After that, a square sub-matrix in parent matrix 2 is arbitrarily defined, as shown in Fig. 7(b) and Fig. 7(c). Fig. 7(e) e Fig. 7(f) illustrates the offspring composition. The blocks R_1, R_2, R_3 and R_4 in the offspring matrix are from the parent matrix 1, the block R_5 is from the parent matrix 2, and the block R_6, is randomly selected from the corresponding block in parent matrix 1 or from 2. Two types of offspring composition are possible and likely to occur: cut-splice, as in Fig. 7(c); cut-overlap, as in Fig. 7(d).

Then, the proposed crossover operator can explore the knowledge content of the body of the parents, and can also promote the diversity of structures. Since selection of the cut point is independent for each Parent Matrix, it is obvious that the length of the produced offspring matrix can vary during the evolution process. Then, circuits with different sizes and complexities evolve together by exchanging their genetic material.

2.7 Bidimensional topology and value mutation operators

Four types of likely topology mutation were defined. The circuit mutation is performed via one of the following operations:

1. adding a randomly selected building-block, without position restriction in the Positional Matrix;
2. deleting a randomly selected building-block, given that the circuit remains connected;
3. deleting a randomly selected building-block from the diagonal matrix, by removing a row/column, given that the circuit remains connected;
4. inserting an arbitrary building-block in cascade into the diagonal of the Positional Matrix, by adding a row/column.

All the parameters of the two-port circuit elements of the circuit may undergo mutation. When a parameter is mutated, a new parameter value is randomly generated through a uniform distribution bounded by the predefined range of possible values.

3. Experiment and results

Several different two-port filters were synthesized, presenting several complexity levels. The proposed method successfully produced filters that complied with the rigorous specifications. The number of circuit evaluations for the entire synthesis process was not

large. A simple bandpass and a type of dual band-pass filter, which offers a considerable difficulty degree, illustrate the application of the proposed method in this section. In recent years, dual-band filters have become extremely important components for wireless communication devices at microwave frequencies (Chen & Hsu, 2006, Hu et al., 2004, Koza et al., 1996, Grimbleby, 2000, Zebulum et al., 1998, Nishino & Itoh, 2002, Lai & Jeng, 2006). Its design is a hard problem, as reported in previous works (see (Lai & Jeng, 2006, Tsai & Hsue, 2004, Zhang, 2007), for example).

The crossover probability was set to 100%, the topology mutation probability was set to 20%, and the parameters mutation probability was set to 5%. The circuit performance was analyzed at 80 discrete frequencies. In all 10 runs, the proposed algorithm achieved results that accomplished the specifications.

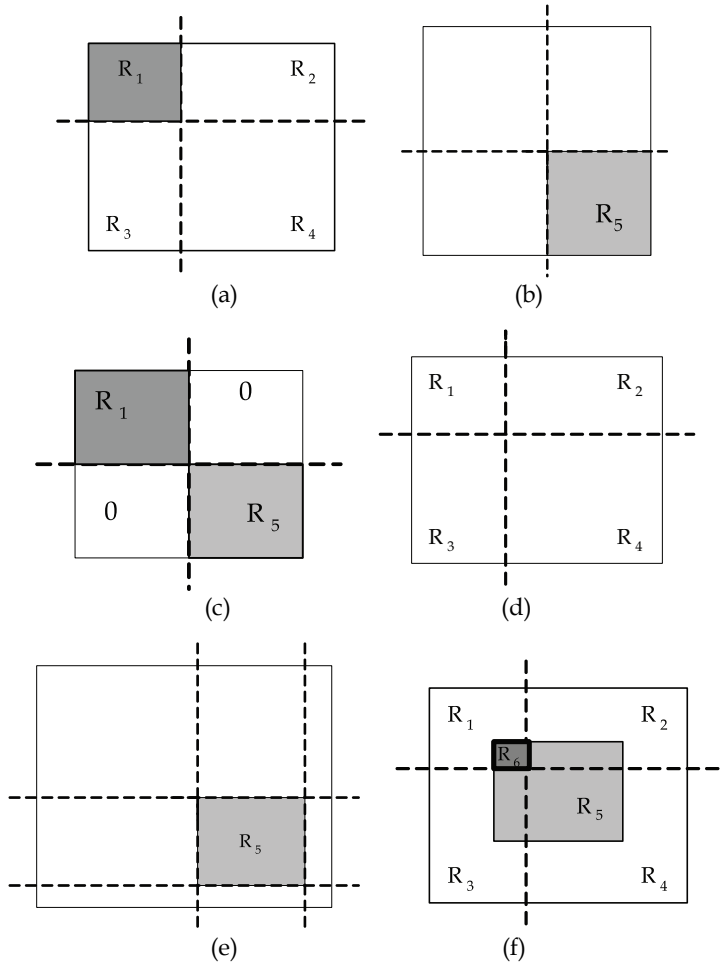


Fig. 7. Crossover operator: (a) parent matrix 1, (b) parent matrix 2, (c) cut-splice operation, (d) parent matrix 1, (e) parent matrix 2, and (f) cut-overlap operation.

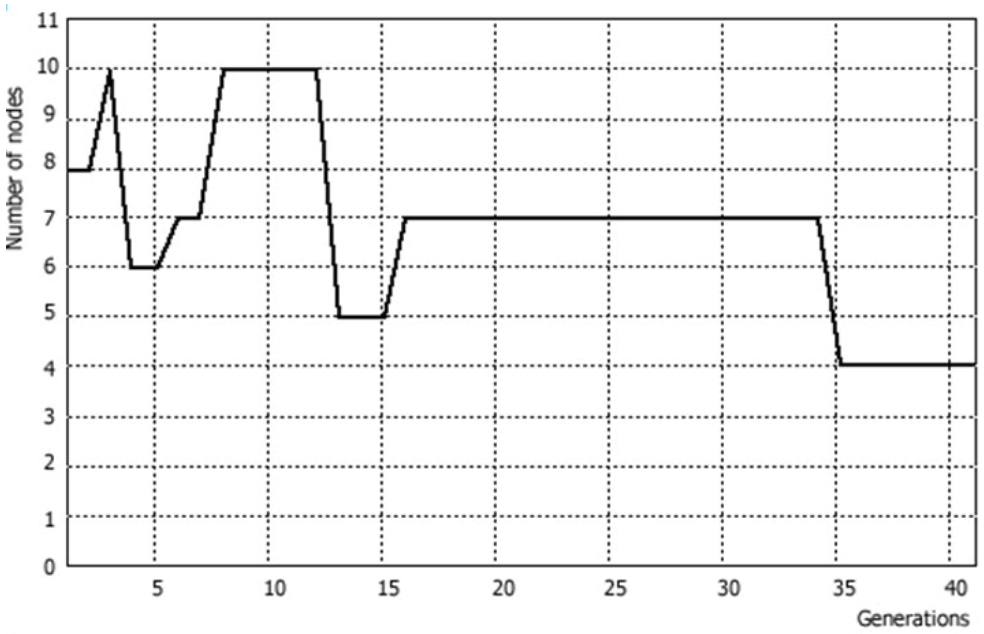


Fig. 8. Evolution of number of nodes (size of the circuit) of the best solution.

3.1 Narrowband filter

A simple bandpass filter was generated in this experiment. The best solution has a circuit with 4 nodes and 12 components. The solution was obtained after 41 generations with populations of 30 circuits, i.e. with 16,000 circuit evaluations. It can be observed in Fig. 8 that the size of the circuit varies during the evolution process, which means that changes occur until the process reaches a structure that meets the specifications. One can also notice that the proposed method achieves a solution topology, as shown in Fig. 9, where building-blocks naturally appear due to the evolution process, e.g. the sub-circuit composed by an inductor and a capacitor in parallel that appears regularly on the circuit. The frequency response of the circuit is shown in Fig. 10. Koza (Koza et al., 1996), using Genetic Programming with populations of 640,000 individuals and 199 generations (127,360,000 circuit evaluations) achieved a circuit with 38 components. Grimblebly (Grimblebly, 2000),

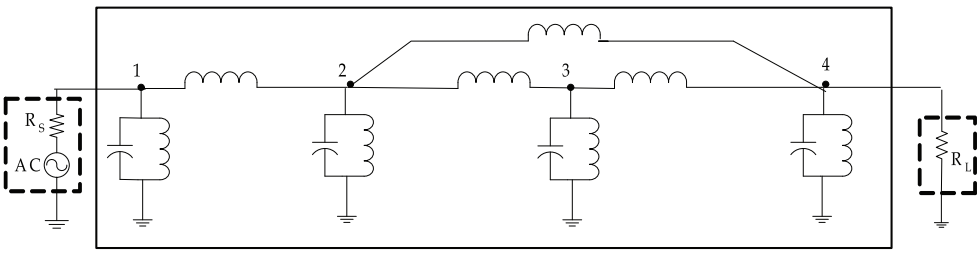


Fig. 9. Best topology obtained.

using a hybrid Genetic Algorithm, obtained a circuit with 4 nodes, but did not mention the computational effort required to reach such a solution. Shin (Shin & Histoshi, 2003), using a multi-stage Genetic Algorithm obtained a solution with a population of 2,000 individuals after 400 generations, which represents 800,000 circuit evaluations. The authors did not present the obtained circuit structure.

3.2 Dual-band filter

In this experiment, a filter for dual-band systems was synthesized. The same filter was synthesized in (Lai & Jeng, 2006); (Tsai & Hsue, 2004) with the following specifications: the return losses (reflection coefficient inside the pass-bands) within 3.4─3.6 and 5.4─5.6 GHz > 10 dB, and the rejections (transmission coefficient outside the pass-bands) within 2.0─3.0, 4.0─5.0, and 6.0─7.0GHz > 20 dB.

Table 1 and Fig. 11 present the building-blocks and connection types. Besides that, during the evolution process, as a topology constraining rule, only common junctions in microstrip circuits were allowed (step-, tee-, and cross junctions) (Tsai & Hsue, 2004).

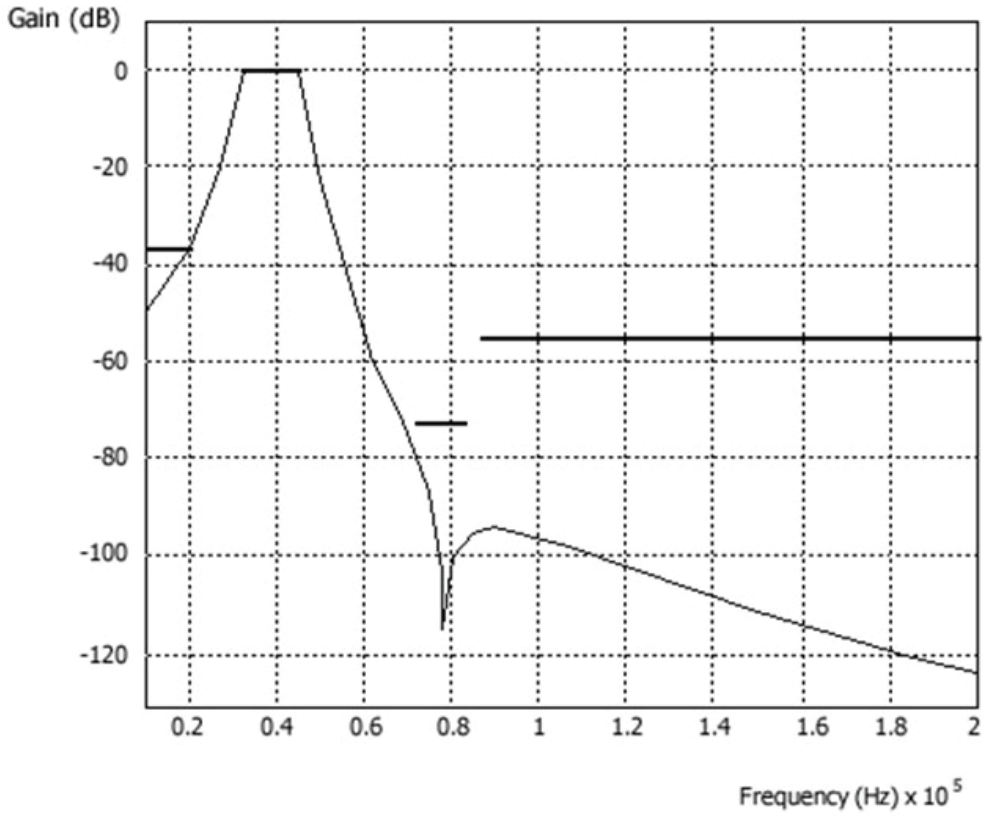


Fig. 10. Frequency response of the best solution.

Two-port circuit building blocks	Connection types	Physical parameters	Electrical parameters (lower and upper bounds)	
			Z_{01}, as a function of L *(length)*	Θ_1 (at 4 GHz), as a function of W *(width)*
TL	Cascade or Parallel	L, W	40 – 110	30 – 100
Sh-TL-OC	Cascade	L, W	40 – 110	20 – 160
Sh-TL-SC	Cascade	L, W	40 – 110	20 – 160
Sh-TL2-OC	Cascade	L_1, W_1, L_2, W_2	40 – 110	20 – 120
Sh-TL2-SC	Cascade	L_1, W_1, L_2, W_2	40 – 110	20 – 120

Table 1. Building-blocks and connection types.

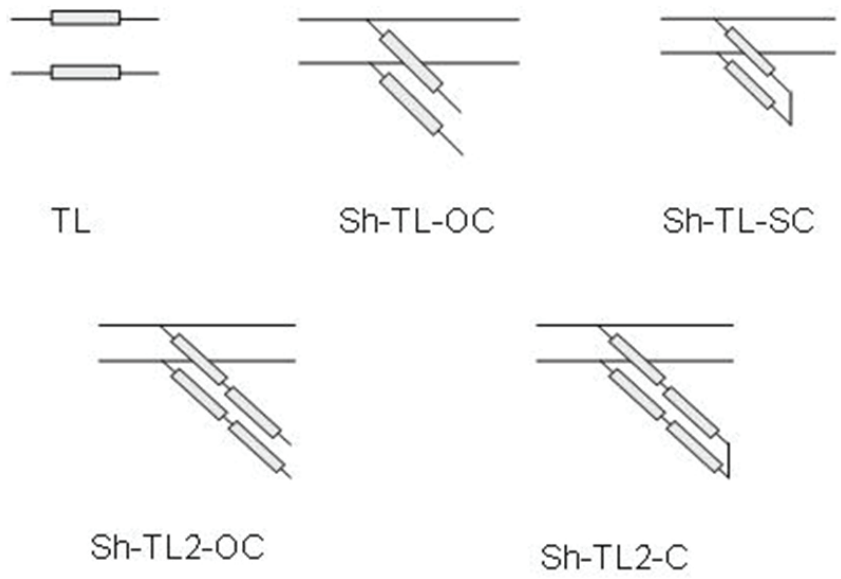


Fig. 11. Building-blocks of the experiment.

The best topology obtained with the proposed method has 11 two-port circuit elements, achieved after 20,285 circuit evaluations. It is a very compact topology and matches the specifications, as shown in Fig. 12. Besides this high-quality solution, the designer has a set of trade-off solutions available into the elite population (Pareto front). For instance, another good solution in the Pareto front has 8 circuit elements, although the frequency response was slightly worse. It can be compared with the result presented in (Lai & Jeng, 2006), which uses a mono-objective hybrid encoded Genetic Algorithm. In (Lai & Jeng, 2006), the best solution was composed by 10 circuit elements, and was achieved after 300 generation, with a population size of 200, or 60,000 circuit evaluations. Results as good as the ones in (Lai & Jeng, 2006) were achieved with a lower number of circuit evaluations and moreover, the proposed method produced filters smaller than those presented in (Tsai & Hsue, 2004).

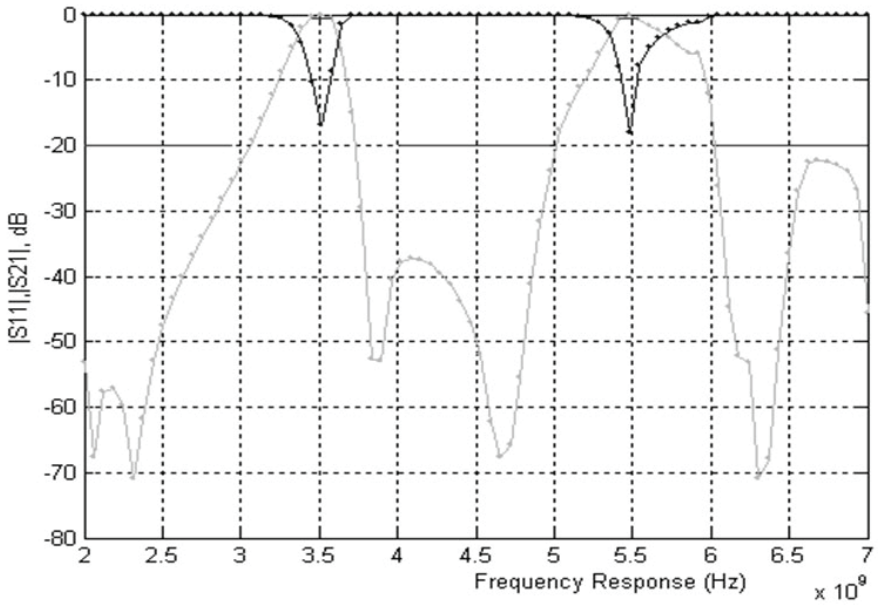


Fig. 12. The frequency responses of one Pareto solution. The thick black line represents the user-defined mask. The gray line is the transmission response $|S_{21}|$. The black line is the reflection response $|S_{11}|$.

4. Conclusions

The circuit design to meet increasingly stringent specifications is a demand of modern applications. The evolutionary methodologies applied to the problem of synthesis of analog circuits have been producing competitive solutions with the solutions found by conventional techniques developed by experts. However, it is a difficult problem, requiring a balance between optimizing topologies and parameters and also presents several open issues. The hybrid evolutionary method presented in this chapter has evolved over time (Dantas, 2006a, 2006b, 2007). The latest proposal, with representation for circuit elements of two ports, using the Positional Matrix, is appropriate to obtain arbitrary topologies and parameters of circuits using small populations, few generations and with a low computational cost. The proposed method has proved robust and flexible. We performed several tests for filters with bandwidth of various sizes, with the desired response with differing degrees of asymmetry, multi-bands, broadband, all with promising results. This indicates that the method is able to generate various compact topologies that meet the design specifications and shows characteristics such as regularity and controlled structural complexity. The circuit elements used, provided by the user, were reorganized during the evolution process into new blocks, appearing with some degree of redundancy in the formation of the final circuit. This means that the evolutionary strategy developed is able to keep the size of the circuit under control, and is still effective in maintaining an elite of better building-blocks to compose the next generation of circuits.

Another important aspect is the choice of a bi-objective approach, which provides a trade-off between performance and size of circuits. Miniaturization is a requirement of current

applications, depending on the development of new materials and manufacturing technologies. On the other hand, after the optimization process, the designer has at his disposal a set of solutions in the Pareto front. He can thus use their expert knowledge to choose the best solution, taking into account the application at his hand.

In future works, we intend to work with multiple objective functions, using the classifying process of the NSGA-II method, but making a preference-based articulation along the evolutionary process. In the synthesis of a multi-band filter, for example, each band may correspond to an objective-function and to each of them could be assigned a preference level.

5. References

Antoniou, A. & Lu, W. (2007). *Practical Optimization: Algorithms and Engineering Applications*, 1th Ed, Springer, ISBN 0387711066.

Cameron, R.J.; Faugere, J.C.; Rouillier, F & Seyfert, F. (2007). Exhaustive Approach to the Coupling Matrix Synthesis Problem And Application to the Design of High Degree Asymmetric Filters. *International Journal of RF and Microwave Computer-Aided Engineering - Special Issue on RF and Microwave Filters, Modeling and Design*, Vol. 17, No. 1, January 2007, 4-12, ISSN 10964290.

Chen, C. & Hsu, C. (2006). Design of a UWB Low Insertion Loss Bandpass Filter with Spurious Response Suppression. *Microwave Journal*, Vol. 49, No. 2, February 2006, 112-116.

Dantas, M.; Brito, L. & Carvalho, P.. (2006a). Biobjective Hybrid Evolutionary Algorithm Applied to Resonator Filters of Arbitrary Topology. *13th IEEE International Conference on Electronics, Circuits and Systems*, Nice, September 2006, Vol. 1, 1-5.

Dantas, M.; Brito, L. & Carvalho, P.. (2006b). Multi-objective Memetic Algorithm Applied to the Automated Synthesis of Analog Circuits. *Lecture Notes in Computer Science*, Vol. 4140/2006, 258-267.

Dantas, M., Brito, L., Carvalho, P., Abdalla., H. (2007). Design of Microwave Filters Topologies using a Hybrid Evolutionary Algorithm. *Journal of Microwaves and Optoelectronics*, Vol. 6, 295-309, ISSN 21791074.

Dastidar, T. R.; Chakrabarti, P. P. & Ray, P. (2005). A Synthesis System for Analog Circuit Based on Evolutionary Search and Topological Reuse. *IEEE Transactions on Evolutionary Computation*, April 2005, Vol. 9, No. 2, 211-224.

Deb, K.; Pratap, A.; Agarwl, S. & Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, April 2002, Vol. 6, No. 2, 182-197.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st Ed., Addison-Wesley Professional, ISBN 0201157675.

Grimbleby, J. B. (2000). Automatic Analogue Circuit Synthesis using Genetic Algorithms. *IEEE Proceedings: Circuits, Devices and Systems*, Vol. 147, No. 6, 319-323, December 2000, ISSN 1350-2409.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems*, The MIT Press, ISBN 0262581116.

Hou, H., Chang, S. & Su, Y. (2005). Practical Passive Filter Synthesis using Genetic Programming. *IEICE Trans. Electron.*, Vol. E88, No. 6, June 2005, 1180-1185, ISSN 09168524.

Hsu, M. & Huang, J. (2005). Annealing algorithm applied in optimum design of 2.4 GHz and 5.2 GHz dual-wideband microstrip line filters. *IEICE Trans. Electron.*, Vol. E88-C, No. 1, January 2005, 47-56, ISSN 09168524.

Hu, J., Goodman, E. & Rosenberg, R. (2004). Robust and Efficient Genetic Algorithms with Hierarchical Niching and a Sustainable Evolutionary Computation Model. *Proc. of the Genetic and Evolutionary Computation Conference*, *Lecture Notes in Computer Science, Springer, GECCO-2004*, Seattle, June 2004, Part I, 1220-1232.

Im, C., Jung, H. & Kim, Y. (2003). Hybrid Genetic Algorithm for Electromagnetic Topology Optimization. *Transactions on Magnetics*, September 2003, Vol. 39, No. 5, 2163–2169, ISSN 00189464.

Jun, D., Lee, H., Kim, D., Lee, S. & Nam, E. (2005). A Narrow Bandwidth Microstrip Band-Pass Filter with Symmetrical Frequency Characteristics. *ETRI Journal*, Vol. 27, No. 5, October (2005), 643-646.

Koza, J., Andre, F. & Keane, M. (1996). Automated Design for Both Topology and Components Values of Electrical Circuits using Genetic Programming. *Genetic Programming 1996: Proceedings of the First Annual Conference*, 123-131, July 1996, Stanford University, Cambridge MA..

Lai, M. & Jeng, S. (2006). Compact Microstrip Dual-Band Bandpass Filters, Design Using Genetic-Algorithm Techniques. *IEEE Transactions on Microwave Theory and Techniques*, Vol. 54, No. 1, , January 2006, 160-168, ISSN 00189480.

Lenoir, P.; Bila, S.; Seyfert, F.; Baillarget, D. & Verdeyme, S. (2006). Synthesis and Design of Asymmetrical Dual-band Bandpass Filters Based on Network Simplification. *IEEE Transactions on Microwave Theory and Techniques*, Vol. 54, No. 7, July 2006, 3090-3097, ISSN 00189480.

Levy, R.; Fellow, L. & Petre, P. (2001). Design of CT and CQ Filters Using Approximation and Optimization. *IEEE Transactions on Microwave Theory and Techniques*, Vol. 49, No. 12, December 2001, 2350-2356, ISSN 00189480.

Mesquita, A.; Salazarand F. & Canazio, P. (2002). Chromosome Representation through Adjacency Matrix in Evolutionary Circuit Synthesis. *NASA/Conference on Evolvable Hardware*, 102-112, ISBN 0769517188, July 2002.

Michalewicz, Z. & Fogel, D. (2002). *How to Solve It: Modern Heuristics*. 2nd Ed., Springer, ISBN 9783540224945.

Nishino, T. & Itoh, T. (2002). Evolutionary Generation of Microwave Line-Segment Circuit by genetic Algorithms. *IEEE Transactions on Microwave Theory and Techniques*, Vol. 50, No. 9, September 2002, 2048-2055, ISSN 00189480.

Shin, A. & Histoshi, I. (2003). Variable length Chromosomes for Analog Evolvable Hardware. In: *Advances in Evolutionary Computing: Theory and Applications*, A. Ghosh and S. Tsutsui, 643-662, Springer-Verlag, ISBN 3540433309.

Tsai, L. & Hsue, C. (2004). Dual-Band Bandpass Filters using Equal-Length Coupled-Serial-Shunted Lines and Z-Transform Technique. *IEEE Transactions on Microwave Theory and Techniques*, Vol. 52, No. 4, April 2004, 1111-1117, ISSN 00189480.

Uhm, M.; Lee, J.; Yom, I. & Kim, J. (2006). General Coupling Matrix Synthesis Method for Microwave Resonator Filters of Arbitrary Topology. *ETRI Journal*, Vol. 28, N. 2, April 206, 223-226.

Wang, H.; Zhang, A. & Fang, D. (2005). The Design of Cross-Coupled Microstrip Filter Exploiting Aggressive Space Mapping Technique. *Microwave Conference – APMC 2005 Proceedings*, Vol. 5, 1-3, December 2005.

Zebulum, R.; Pacheco, M. & Vellasco, M. (1998). Comparison of Different Evolutionary Methodologies Applied to Electronic Filter Design. *Proceedings of the IEEE International World Congress on Computational Intelligence, ICEC'98*, 434-439, Alaska, May 1998.

Zhang, H. (2007). *Compact, Reconfigurable, and dual-band microwave circuits*. Doctorate Thesis, Dep. of Electronic and Computer Engineering, 1-187, Hong Kong.

22

Feature Extraction from High-Resolution Remotely Sensed Imagery using Evolutionary Computation

Henrique Momm and Greg Easson
University of Mississippi Geoinformatics Centre (UMGC)
United States of America

1. Introduction

Feature extraction, in the context of remote sensing, can be defined as image processing techniques to identify and to classify mutual relationships or mutual meaning between image regions (Baatz et al., 2000). The aggregation of image pixels forming image regions and their relationship to other image regions are interpreted and used as cues in the information retrieval process (Quackenbush, 2004). A common approach is to create hierarchical structures of image regions in which fine-scale image regions constitute portions of other coarse-scale image regions (Niemeyer & Canty, 2001). Feature extraction differs from traditional pixel-based remote sensing image classification algorithms in which each, individual pixel (or pixel vector in the case of images with more than one channel) is individually evaluated and assigned to one class (Lillesand & Kiefer, 2000). The difference between low-level information extraction techniques using traditional pixel-based classification methods and high-level information extracted by a human analyst is often referred to as the "*semantic gap*" (Smeulders et al., 2000). Human analysts use a complex combination of different image cues such as colour (spectral), texture, shape (geometry of image regions), and context (relationship between image regions). However, human analysis of large areas and multiple images is costly and time consuming (Munyati, 2000).

As the volume of available remotely sensed imagery increases by many orders of magnitude, one of the challenges faced by many organizations and institutions is converting large quantities of images into actionable information and intelligence. Because human analysis of large areas and sometimes over multiple periods of time is costly and time consuming, scientists have recognized the importance of developing more sophisticated semi-automated or automated feature extraction techniques to improve the information extraction process. The challenge resides in multifaceted problems where the relationship between image's regions is too complex to be solved by explicit programming (hard computation) and/or these problems require the system to adapt and evolve when image conditions change. This provision is particularly important in remote sensing applications due to changing factors such as variation in sensor spatial and spectral resolutions, change in environmental conditions between images, and specificity of the feature of interest.

The use of stochastic algorithms to address these complex feature extraction problems, are now being investigated as a possible alternative; due to their properties of deriving

solutions from a small set of positive and negative examples through an optimized combinatorial search rather than being explicit programmed (Fogel, 2000; Mitchell, 1997). Evolutionary algorithms (also referred to as evolutionary computation) have been used to solve problems in different domains, including remote sensing applications.

In this chapter, the use of evolutionary algorithms, in the form of genetic programming, to aid the feature extraction process from high-resolution satellite imagery was evaluated. A novel framework involving genetic programming, standard image processing methods, and clustering algorithms is described. The proposed system was designed to support routine feature extraction procedures from satellite imagery and is composed of two modes: development and operational. In the development mode, a single and representative image in conjunction with human analyst input are used to train the system to develop the candidate solutions. The operational mode applies the developed candidate solutions to unforeseen images in an automated fashion, thus expediting the information extraction process. In this study, the objective was to quantitatively access the generalization capability of the proposed system to imagery variations in physical and environmental factors such as distinct features with similar spectral signatures, variations in sensor's resolution, and environmental condition changes between scenes. The proposed methodology uses a biologically-inspired framework to extract and combine in non-linear way, image derived information such as colour (spectral characteristics) and shape (image region geometrical properties). The accuracy of the framework was quantitatively assessed through a cross-evaluation procedure where a set of different image chips is used to develop candidate solutions in one scene (development mode) and then test those solutions in the remaining unforeseen scenes (operational mode).

2. Background

2.1 Remote sensing and remote sensing spectral indices

Remote sensing can be defined as the science of deriving information about a feature, an object, or a phenomenon from a distance by analyzing the energy reflected or emitted by the feature (Aronoff, 2005; Lillesand & Kiefer, 2000). The main energy detected by remote sensing systems is electromagnetic energy. Remote sensing uses sensors to measure the amount of electromagnetic energy exiting an object or a geographic area. Remote sensing sensors are characterized by different resolutions such as spatial (relative ground sampling distance of one pixel), spectral (number of electromagnetic regions sampled), radiometric, and temporal (revisit time).

Because objects and/or features at the Earth's surface interact differently with the electromagnetic energy based on their molecular composition, differences in the amount and properties of electromagnetic radiation becomes a valuable source of information. Through the use of multiple parts of the electromagnetic spectrum, represented by multiple channels in remote sensing images, it is possible to generate spectral signatures and/or data transformations to aid information retrieval.

Spectral band indices are the most common spectral transformations used in remote sensing. These spectral indices apply a pixel-to-pixel operation to create a new value for individual pixels according to some pre-defined function of the spectral values (Momm et al., 2006). After the transformation, some features and/or spectral properties become more discernable when compared to the original data (Figure 1).

The challenge resides in the development of such spectral indices. The use of existing indices in new environments or the development of new spectral indices constitutes a time consuming and complex problem (Momm et al., 2007). Different features with similar spectral signatures add to the complexity of creating such indices. Spectral indices level of complexity varies according to the relationship between feature's spectral responses to different parts of the electromagnetic spectrum.

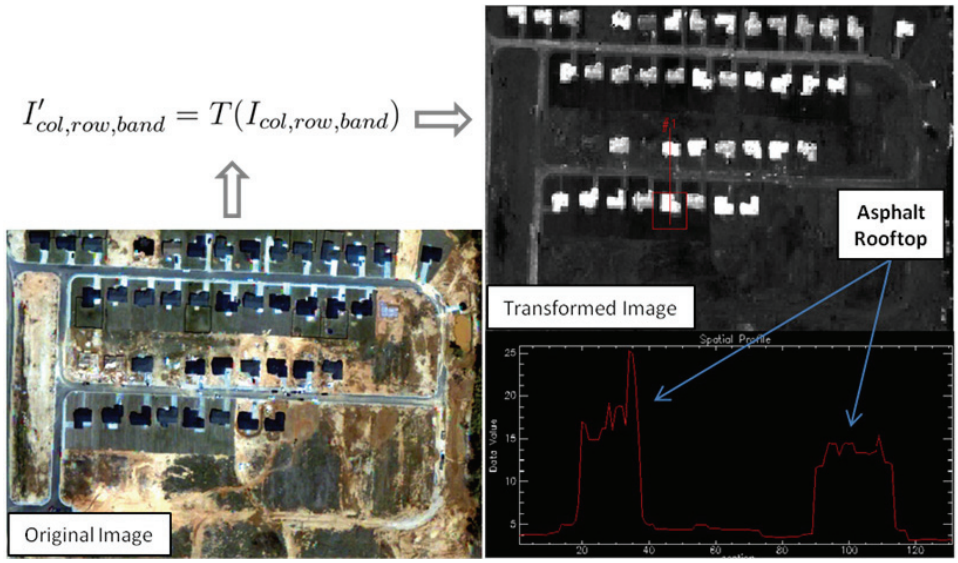


Fig. 1. Illustration of the use of spectral indices to transform the original multi-spectral image for enhanced information extraction. Example shows a spectral profile of the transformed image highlighting asphalt-based residential rooftops.

2.2 Evolutionary algorithms for remote sensing feature extraction

Easson and Momm (Easson & Momm, 2010) have provided a detailed survey of the use of evolutionary algorithms to extract information from remotely sensed data. In their review, the different applications were classified into four categories according to the general research objective: image enhancement, image classification, modelling, and feature extraction. Their literature investigation also revealed that the majority of applications are based on genetic algorithms (GA) and genetic programming (GP).

In image enhancement categories the applications described used GA and GP as an optimization tool to improve some image processing problem by defining which basic image processing operation, or sequence of operations, to use to solve the problem. The objective of image classification algorithms is to automatically (or semi-automatically) categorize all pixels in an image into classes (Lillesand & Kiefer, 2000) based on multi-dimensional spectral similarities of electromagnetic measurements at various wavelengths. The use of evolutionary algorithms to aid satellite image classification is the most common problem addressed and more than 15 publications were identified. In the modelling category, evolutionary algorithms were used to optimize the search for model's parameters or to define new models designed to obtain measurements from remotely sensed imagery.

Specifically for feature extraction applications, literature investigation indicates that this field of research is relatively new and unexplored. Daida and others (Daida et al., 1995; 1996) used genetic programming to identify pressure ridges in Arctic ice through the use of synthetic aperture RADAR images. In this work, a set of texture-based filters (convolution functions) were considered and genetic programming was used to select the most appropriate filter (or combination of filters) to highlight pressure ridges. Similarly, Howard and Roberts (Howard & Roberts, 1999) used a combination of image regions statistics, texture-based filters, and genetic programming to develop a vehicle and ship detector. In this work a two step process was used, object location and object classification.
Recent contributions have employed evolutionary algorithms in the task of deriving ontology rules describing characteristics and relationships between image regions (Durand et al., 2007). The ultimate goal is to develop tools to partially replicate the human ability to interpret images (Easson & Momm, 2010). Forestier and others (Forestier et al., 2008) researched the use of genetic algorithms to optimize the search for ontology rules to segment satellite imagery. Candidate solutions, composed of non-linear combinations of the primitive rules developed by genetic algorithms, where then compared to human-derived ontology rules. The definition of ontology rules for feature extraction from remotely sensed data is a complex and time consuming task and the use of evolutionary algorithms to optimize the search and definition of such rules are now the subject of ongoing research (Forestier et al., 2008; Momm et al., 2009; Puissant et al., 2007).

3. Evolutionary framework

3.1 Framework description

The proposed framework develops spectral indices, in the form of mathematical expressions of the original image's channels, to create a transformed image; which maximizes the performance of standard classification algorithms to separate the target feature from the remaining image background (Momm at al., 2009). The system works in a learn-from-examples approach where positive and negative samples are used by the genetic programming algorithm to evolve candidate solutions through an optimized iterative search.
In the development mode, the system requires three inputs: original image, parameters controlling the run, and reference image (Figure 2). The original image consists of a representative multi-spectral image containing the feature to be extracted. Success of machine learning algorithms are dependent on the quality of the training set; and therefore, when designing applications involving feature extraction it is necessary to understand the sensor's limitations (spectral, spatial, and radiometric resolutions) and contrast them with the feature's spectral and spatial characteristics. The parameters controlling the run involve the definition of the terminal set (image's spectral channels), function set (list of basic mathematical functions used as the building blocks to evolve candidate solutions), population size, number of generations, percentage of crossover, stopping criteria, and restarting threshold (measure to maintain diversity during the evolutionary process). Reference data consists of human classified set of positive and negative samples.
During the initial generation, genetic programming randomly generates a set of candidate solutions (mathematical expressions) referred to as population. This set of candidate solutions are then individually applied to the original multi-spectral image resulting in a new set of transformed images; which are individually clustered and compared to the reference image for fitness computation. If either of the stopping criteria are met (fitness threshold or maximum number of iterations) the system sorts the candidate solutions by

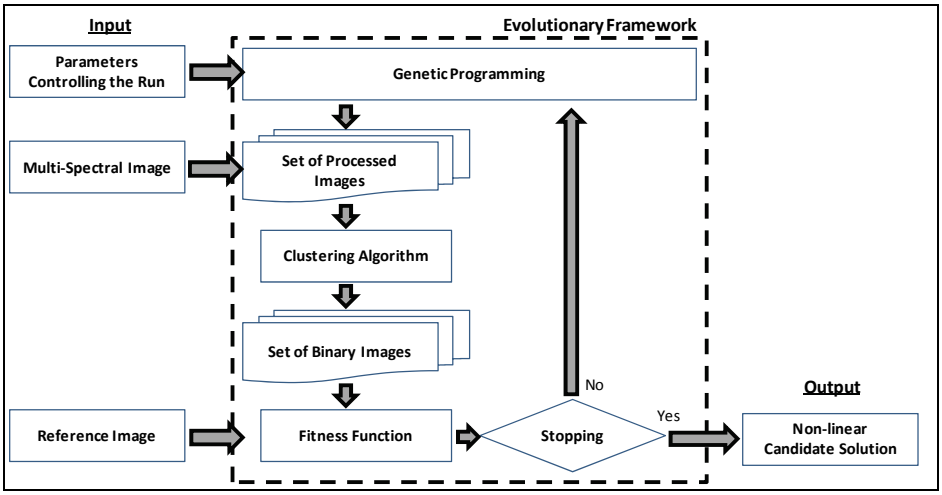


Fig. 2. Simplified flowchart illustrating data/parameters input, output, and main internal components of the evolutionary framework.

fitness values and then outputs the most fit candidate solution. On the other hand, if the stopping criteria are not met, the system performs genetic operations (cross over and restarting) with the top most fit individuals to create a new population and the entire process is iteratively repeated until the stopping criteria are met.

3.2 Fitness function

Cohen's kappa coefficient of agreement was selected as the statistical measurement of fitness for each candidate solution (Cohen, 1960). When comparing the binary image obtained by clustering of the transformed image to the user-provided reference data, kappa is preferred over simple measure of percent of agreement because it corrects for the amount of agreement due to chance. Kappa statistics can be computed as:

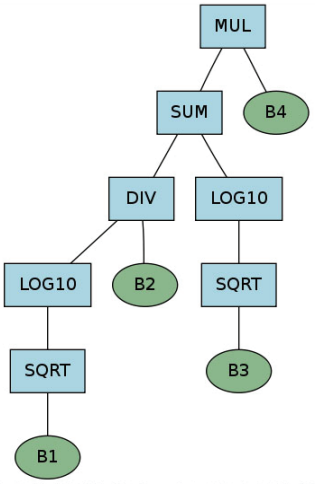
$$k = \frac{N\left(\sum_{i=1}^{r} x_{ii} - \sum_{i=1}^{r}\left(x_{i+} * x_{+i}\right)\right)}{N^2 - \sum_{i=1}^{r}\left(x_i * x_{+i}\right)} \tag{1}$$

In this equation, after computing the contingency table (Jensen, 1996), r represents the number of rows, X_{ii} the sum of values in the major diagonal, X_{i+} the sum of observations in row i, X_{+i} the sum of observations in column i, and N the total number of observations.

3.3 Multi-stage implementation and candidate solution representation

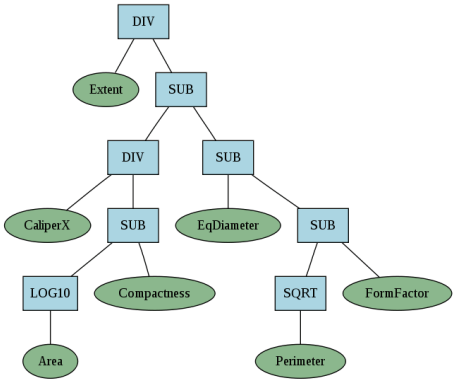
The objective of using a sequence of steps to extract the desired information from imagery is based on the premise that complex problems can be partitioned into a series of easier-to-solve smaller problems. In theory, machine learning algorithms can master smaller tasks and when combined, the set of specialized algorithms can outperform an algorithm designed to solve the overall problem. Following this concept, the initial stages are designed to address spectral characteristics while in the latest stage geometric properties of group of connected pixels (image objects) are considered. Each subsequent stage uses as input the results of the previous stage.

The initial steps address the identification of pixels with similar spectral characteristics to the feature of interest. Ontology rules are not considered since, after the mathematical transformation of the image, pixels are individually analyzed and classified. Transformation functions, also referred to as spectral indices, use the image spectral channels as arguments (Figure 3).



(((LOG10 (SQRT (B1)) / B2) + LOG10 (SQRT (B3))) * B4)

Fig. 3. Example of genetic programming candidate solution representation as a hierarchical tree structure (internal) and as a mathematical expression (external) used in the spectral pixel classification stages of the feature extraction process.



(Extent / ((CaliperX / (LOG10 (Area) - Compactness)) - (EqDiameter - (SQRT (Perimeter) - FormFactor))))

Fig. 4. Example of genetic programming candidate solution representation as a hierarchical tree structure (internal) and as a mathematical expression (external) used in the geometric stages of the feature extraction process. Arguments are geometric properties of image objects.

In the final stages, the classified image resulted from previous stage is processed to identify groups of connected pixels (segmentation), label each group of connected pixels with a unique identifier, and computation of multiple geometric descriptors (Momm et al., 2010).

The input for the geometric stage consists of an raster grid image with the number of image object as the number rows and the number of shape descriptors as the number of columns. Candidate solutions in the geometrical stages use as arguments the geometrical shape descriptors (Figure 4).

4. Feature extraction experiment

The overall research objective is to develop a system to be used in routine operational situations by extracting specific information from sets of imagery with minimum human interaction possible. The ideal system should be trained using a small and representative scene and once the solutions are developed, these solutions are then used in a multitude of unforeseen scenes in an automated fashion. In this experiment, the generalization ability of the evolutionary framework is assessed when candidate solutions are applied to different images with changing environmental conditions and remote sensing parameters. The aim of this study is to use its outcome as guidance for future applications by identifying the limitations and strengths of the proposed system.

The problem selected in the evaluation of the evolutionary framework was the identification of residential single family rooftops from high spatial resolution imagery. There are some challenges in the development of algorithms to obtain such information. The limited spectral resolution presented by the current high spatial resolution satellite sensors combined with the spectral similarities between asphalt-based roofing material and asphalt pavement limits the use of pixel-by-pixel classification algorithms. To overcome the spectral similarities limitations, a geometric classification of the spectrally classified material is introduced mimicking the human analyst classification approach. Human's advanced interpretation ability takes into consideration not only rooftop colour information (spectral information) but also our knowledge of rooftop geometry.

Our approach divided the task of identifying single family residential buildings (through rooftop) into three stages (Figure 5). In the first stage, the evolutionary framework is used to evolve spectral transformation to spectrally separate the image pixels into two classes,asphalt-based material and background. Using the results from the first stage, the evolutionary framework is used to evolve a new set of spectral transformation to further separate the pixels previously identified as asphalt-material into either rooftop class or other classe. The third stage obtains geometric properties of each group of connected pixels

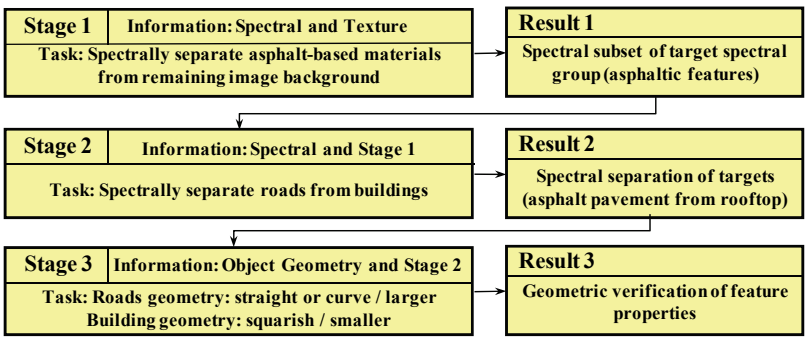


Fig. 5. Flowchart of the multi-stage approach for single family rooftop detection to access the evolutionary framework's ability to generalize as remote sensing and physical conditions change.

identified as single family rooftop and the evolutionary framework is once again used to evolve a third set of mathematical transformations to distinguish single family residential rooftop from other features (such as commercial buildings, pavement, etc) based on geometric properties (Momm et al., 2010).
Examples of the resulting thematic maps from each stage are displayed on Figure 6. The resulting map of stage 1 (step 1 in Figure 6) illustrates the separation between asphalt-based materials (in red colour) and the remaining background (green colour).
The map resulting from stage 1 is fused with the original multi-spectral image to create the input image for stage 2. Green colour pixels in Figure 6 step 1, are used as a filter to mask out pixels from the original multi-spectral image leaving only the pixels marked with red. A new multi-spectral image is created with the same original four channels but pixel can have as values either the original scaled radiance values (red pixels from map created in stage 1) or no data (green pixels from map created in stage 2). The results from stage 2 further discriminate asphalt-based materials into rooftops and others (step 2 in Figure 5). In the middle map, black colour indicates pixels not considered (masked out), red colour indicates target material, and green colour non rooftop materials.
In the final stage, the group of connected pixels resulting from stage 2 (red colour in the map created in stage 2) are further filtered based on geometric properties such that smaller, larger, and elongated image objects differing from single family residential building were removed (step 3 in Figure 6).

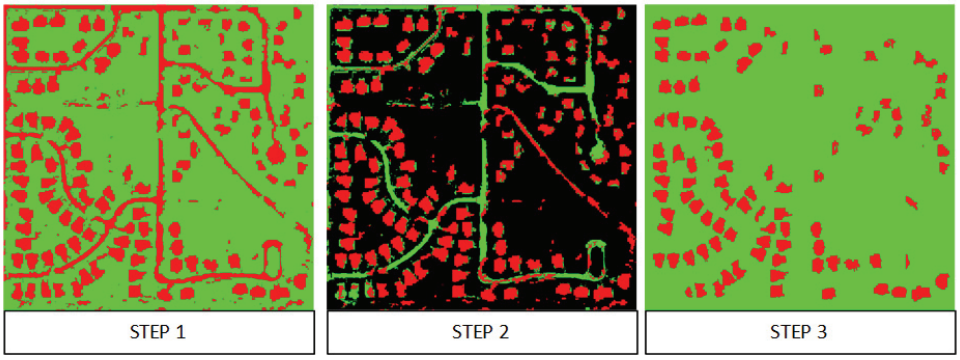


Fig. 6. Illustration of the outcomes produced by the evolutionary framework for each step considered. Step 1 outputs a binary image containing asphalt-based material and background. Step 2 uses the results of step 1 to generate another image discriminating asphalt rooftop from other asphalt-based material. Step 3 uses the output from step 2 containing group of connected pixels and filter them based on geometric properties.

4.1 Data description and preparation steps

Three scenes were used in this experiment, two obtained with the IKONOS sensor and one with the QuickBird sensor (Table 1). The two IKONOS scenes were acquired three years apart during early fall while the QuickBird scene was acquired during the summer. The 2005 imagery was immediately acquired after hurricane Katrina. Trees in the region investigated (south part of Mississippi in the United States of America) does not lose their leaves during the winter; however, there are differences in rain patterns and day light illumination (Table 1).

Scene	Sensor	Acquisition Date	Spatial Resolution (meters)	Spectral Resolution (η meters)	Scan Azimuth.	Sun Elevation.	Sun Azimuth.
1	IKONOS-2	2002-OCT-06	Pan: 1.0 Multi: 4.0	480,550, 665,805	179.97	50.82	154.94
2	IKONOS-2	2005-SEP-06	Pan: 1.0 Multi: 4.0	480,550, 665,805	90.00	63.24	107.5
3	QuickBird	2002-JUL-06	Pan: 0.7 Multi: 2.8	485, 560, 660,830	25.55	70.22	144.20

Table 1. Imagery used in the evaluation of the evolutionary framework in the task of single family residential rooftop extraction.

The scenes also differ in the sensor's spatial and spectral resolution. QuickBird has a nominal spatial resolution of 0.7 meters for the pan-chromatic image and 2.8 meters for the multi-spectral image while IKONOS has 1.0 meter and 4.0 meters for pan-chromatic and multi-spectral images respectively. Both sensors record four spectral channels (blue, green, red, and infra-red) with similar nominal central wavelengths. The largest differences are in the infra-red channel.

All image scenes were provided as scaled radiance at the sensor. An enhanced image was generated by fusing the high spatial resolution pan-chromatic image to the multi-spectral image using the Gram-Schmidt technique (Laben & Brower, 2000). These images were further subset for the generation of image chips (Table 2) to cope with the large computational cost

CHIP Identification	Nominal GSD (meters)	Number of Samples	Number of Lines	Number of Bands	Role
QB02R2	0.7	623	614	4	Training and Testing
QB02R5	0.7	517	677	4	Training and Testing
IK03R1	1.0	242	233	4	Training and Testing
IK02R2	1.0	436	425	4	Training and Testing
IK05R1	1.0	242	233	4	Training and Testing
IK05R2	1.0	436	425	4	Training and Testing
QB02C1	2.8	581	459	4	Testing
QB02C2	2.8	348	475	4	Testing
QB02C3	2.8	646	400	4	Testing
IK05C1	1.0	866	478	4	Testing

Table 2. Description of the image chips characteristics and primary role in the cross-validation process of the evolutionary framework.

involved during developing mode (training). Image chips QB02C1, QB02C2, and QB02C3 were produced using the original multi-spectral image before the resolution enhancement procedure. Each image chip covers areas with different morphological characteristics, environmental conditions, and level of pre-processing (Figure 6 and 7). A summary of the environmental and physical property differences between image chips can be listed as follows:

- Level of oxidation of the asphaltic material. Asphalt-based pavement and roofing material are subject to chemical oxidation over time by prolonged exposure and reaction with atmospheric oxygen leading to changes in electromagnetic reflectance properties. Roofing material in image chips QB02R2 and QB02R5 present contrasting levels of oxidation, thus indicating the presence of younger housing rooftop in the QB02R2 image chip.

- Level of tree coverage of rooftops: Tree canopy coverage of rooftops varies at each residential subdivision. This poses a challenge for the geometric stage where the rectangular shape of residential rooftops may be altered by tree cover.
- Rooftop integrity: Image chips IK05C1, IK05R1, and IK05R2 cover locations impacted by hurricane winds leading to varying levels of rooftop damage ranging from missing shingles to flattened rooftops.

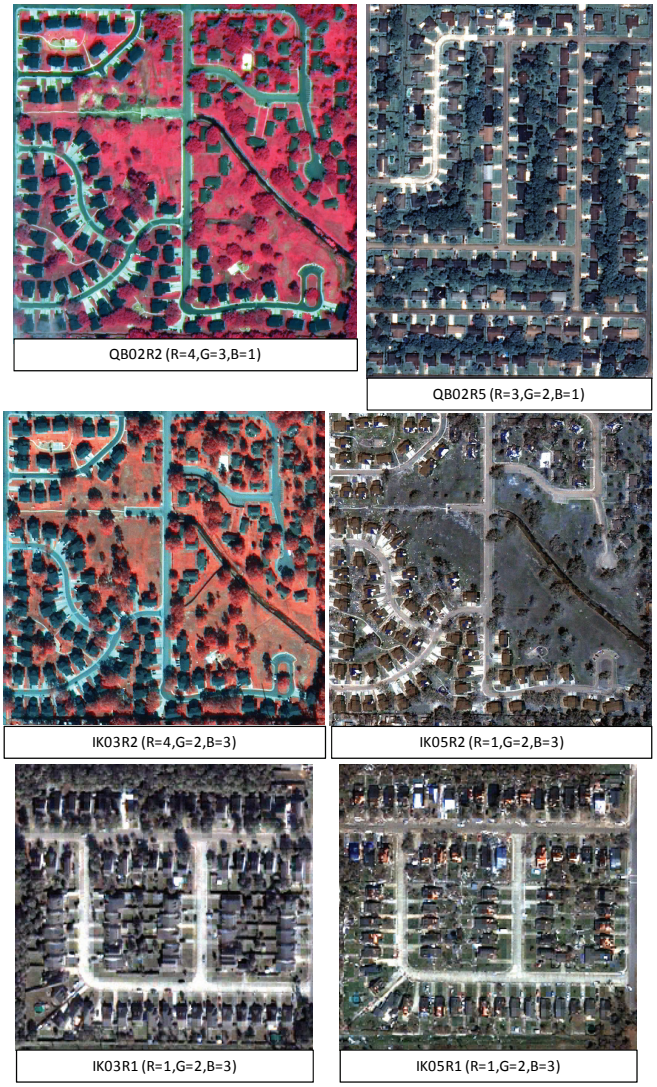


Fig. 6. Image chips used in the evaluation of the evolutionary framework in the task of identifying single family residential rooftops through training and testing procedure. Differences in sensors, acquisition dates, and level of pre-processing were exploited to access the generalization ability of the system to changing conditions.

Reference data was generated by manual classification of individual pixels into one of the three land use covers: rooftop, roads, and other (Figure 8). In the first stage (asphalt material versus background) the land use covers rooftop and roads were combined to form another class, asphalt-based material. The reference data used as input for stage two uses only rooftop and road classes.

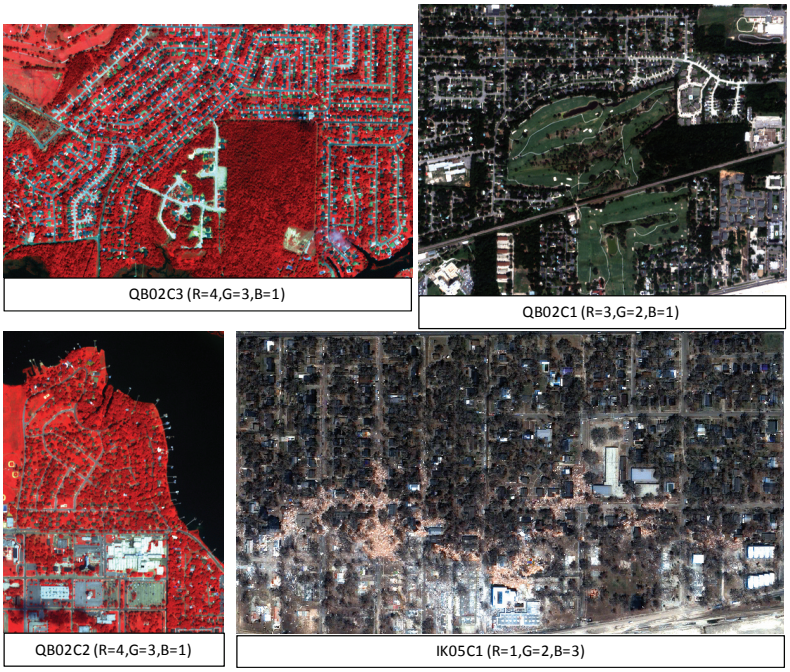


Fig. 7. Additional image chips used in the evaluation of candidate solutions developed by the evolutionary framework for identifying single family residential rooftops.

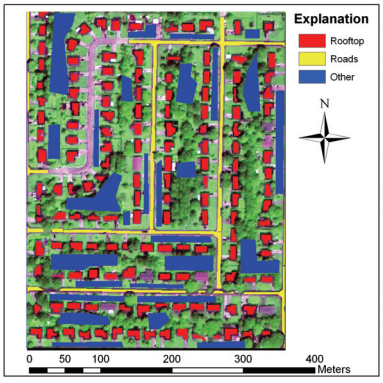


Fig. 8. Example of coloured polygons representing the reference datasets obtained by manual classification of individual image pixels. Background shows QB02R5 with spectral band combination 1-4-1.

The reference data for stage 3 (geometric properties) used the set of resulting images from stage 2. These thematic images were analyzed by human analyst and individual image objects were classified as either residential single family rooftop or other.

4.2 Genetic programming parameters

Two configurations of genetic programming parameters were considered (Table 3). For the first two stages, dealing with spectral information, the terminal set was composed of the four available spectral bands. This configuration used a reduced number of generations and a smaller population size. These were required to cope with the increased computational overhead generated by the utilization of images as arguments in the candidate solutions. During the evolutionary process, calculation of fitness values are formed by a sequence of mathematical expressions where the attributes consist of images that must be processed thousands or even millions of times depending on the population size and the number of generations. Conversely, the number of image objects is many orders of magnitudes smaller than the number of pixels, allowing for the larger population sizes and number of generations. The terminal set in the third stage contains ten geometric descriptors (Table 4).
In both scenarios, constant numbers were not considered as part of the terminal set to promote adaptation and generalization. Constant numbers, selected during the evolutionary process as being part of the solution, are often specific to the training image and thus considered a threat to the generalization ability of the system when the same solution is applied to a different image. It is possible that, in the testing image, the constant number defined during training has a different meaning.

Parameter	Spectral Information	Geometric Information
1. Terminal Set	Image spectral bands	Object's shape descriptors
2. Function Set	Summation (SUM) Subtraction (SUB) Safe Division (DIV) Multiplication (MUL) Safe Square Root (SQRT) Safe logarithm (LOG) Absolute value (ABS)	Summation (SUM) Subtraction (SUB) Safe Division (DIV) Multiplication (MUL) Safe Square Root (SQRT) Safe logarithm (LOG) Absolute value (ABS) Threshold (*if a>b then a else b*) Greater than (*if a>b then 1 else -1*) Lower than (*if a<b then 1 else -1*)
3. Fitness Function	Kappa Coefficient of Agreement	Kappa Coefficient of Agreement
4. Population Size	40	200
5. Generations	70	250
6. Crossover	30%	30%
7. Stopping Criteria	71 or $K^{hat} > 0.975$	251 or $K^{hat} > 0.975$
8. Restarting Threshold	5 generations	5 generations

Table 3. Genetic programming parameters used in the multi-step feature extraction experiment. During the spectral information steps, smaller population size and number of generations were used to cope with the computational cost inherent from using multi-spectral images in the terminal set.

Population diversity was controlled by restarting procedure (Momm & Easson, 2010; Momm et al., 2008) rather than traditional mutation operations. The basic principle of restarting is the introduction of new genetic material into the evolutionary process after a certain number of iterations without change in fitness value of the *most fit* individual of the population.

Shape Descriptors			
1	Area	6	$FormFactor = \frac{4 * \pi * Area}{Perimeter^2}$
2	Caliper X	7	$Roundness = \frac{4 * Area}{\pi * \left(MaxDiameter^2 \right)}$
3	Caliper Y	8	$AspectRatio = \frac{Caliper_x}{Caliper_y}$
4	Perimeter	9	$Campactness = \frac{\sqrt{\frac{4 * Area}{\pi}}}{MaxDiameter}$
5	$EquivalentDiameter = \sqrt{\frac{4 * Area}{\pi}}$	10	$Extent = \frac{Area}{Extent_x * Extent_y}$

Table 4. Shape descriptors of group of connected pixels used in the experiment of identifying residential rooftops from high spatial resolution satellite imagery.

4.3 Cross-evaluation

The evaluation of the system was performed by developing solutions in one image chip and then applying those solutions to the remaining image chips in the pool. Results generated by the system were then quantitatively compared to human classified reference data. For each stage six training-testing configurations were considered resulting in 18 different scenarios. This approach was adopted to verify the system's robustness to environmental and physical condition changes between images.

5. Experimental results and discussion

The accuracy results for each scenario considered were expressed as overall accuracy and Cohen's kappa coefficient of agreement. Kappa values range from -1.0 to 1.0. Negative values mean agreement less than random chance of agreement while positive values are a result of greater than random chance of agreement.

Accuracy results for each scenario are plotted in Figures 8, 9, and 10. The image chips used to develop solutions are identified by a shadowed area in the plots. The remaining points in each plot are accuracy results yielded from using the candidate solutions developed using the image chip in the shadowed area to the other image chips. For example, in the upper left plot in Figure 8, a non-linear spectral transformation was developed to spectrally identify asphalt-based materials using the evolutionary framework with the image chip QB02R2 and its correspondent reference dataset as input. The spectral transformation developed, was then applied to the remaining nine image chips resulting in nine new transformed images. Each transformed image was then clustered into a two-class thematic map and compared to its correspondent reference data for fitness computation (overall accuracy and kappa statistic).

Analysis of the accuracy results for stage 1 (Figure 8), identification of asphalt-based materials, indicates that solutions developed and tested using the sensor QuickBird produced an overall consistent pattern of accuracy results despite the differences in pre-processing between image chips. These findings were expected due to the smaller level of difficulty of this task. Conversely, training and testing results between image chips produced from IKONOS 2003 and 2005 imagery did not agree. This could be attributed to the differences in shade length and orientation between these scenes caused by distinct sun elevation and azimuth angles (Table 1).

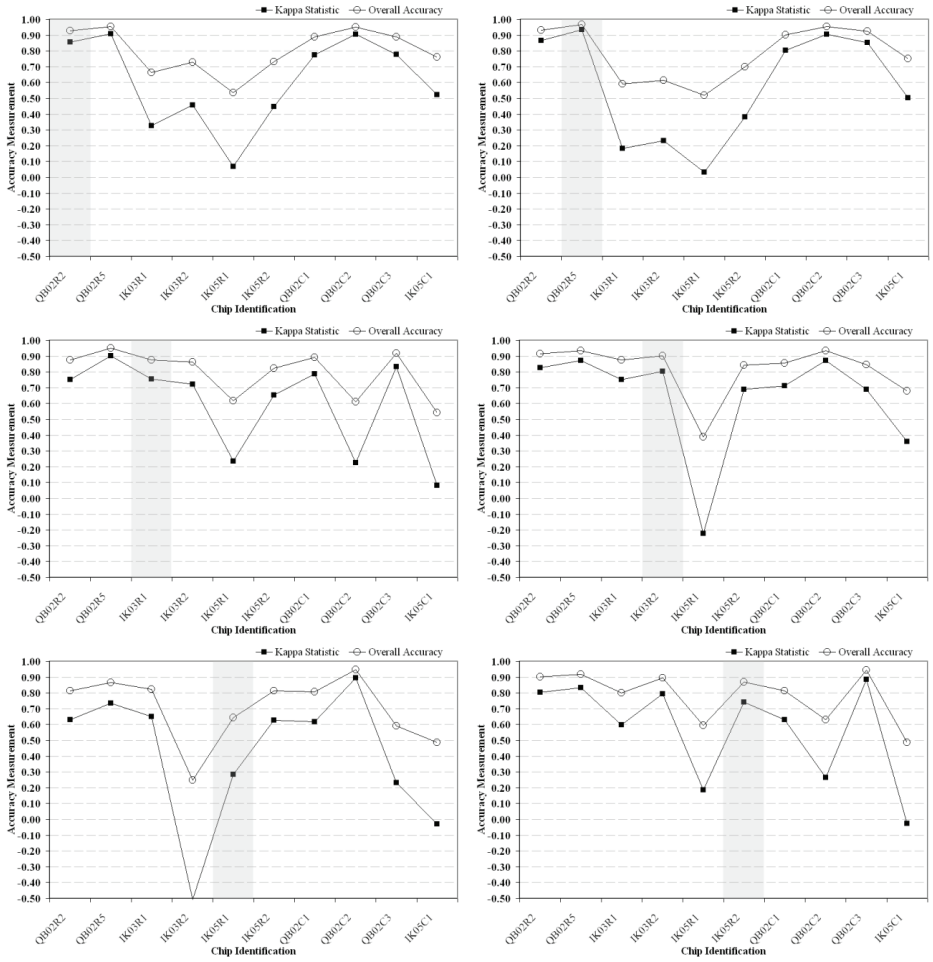


Fig. 8. Accuracy results of step 1 in the cross-evaluation of candidate solutions for imagery classification developed using the evolutionary framework. In step 1 candidate solutions were developed to spectrally classify individual pixels as either asphaltic material or background. The shadowed vertical bar represents the image chip used for training (developing the candidate solutions) and the remaining points are the testing results when using the candidate solution developed using the training image.

The thematic maps, produced in stage 1, were used to create the necessary input files for stage 2. Image chips derived from the QuickBird sensor repeated the good generalization performance previously displayed in stage 1 (Figure 9). Results also indicated that age of roofing material and roads had little or no effect in the QuickBird-based image chips. The image chip IK05R1 and IK03R2 resulted in the poorest performance.

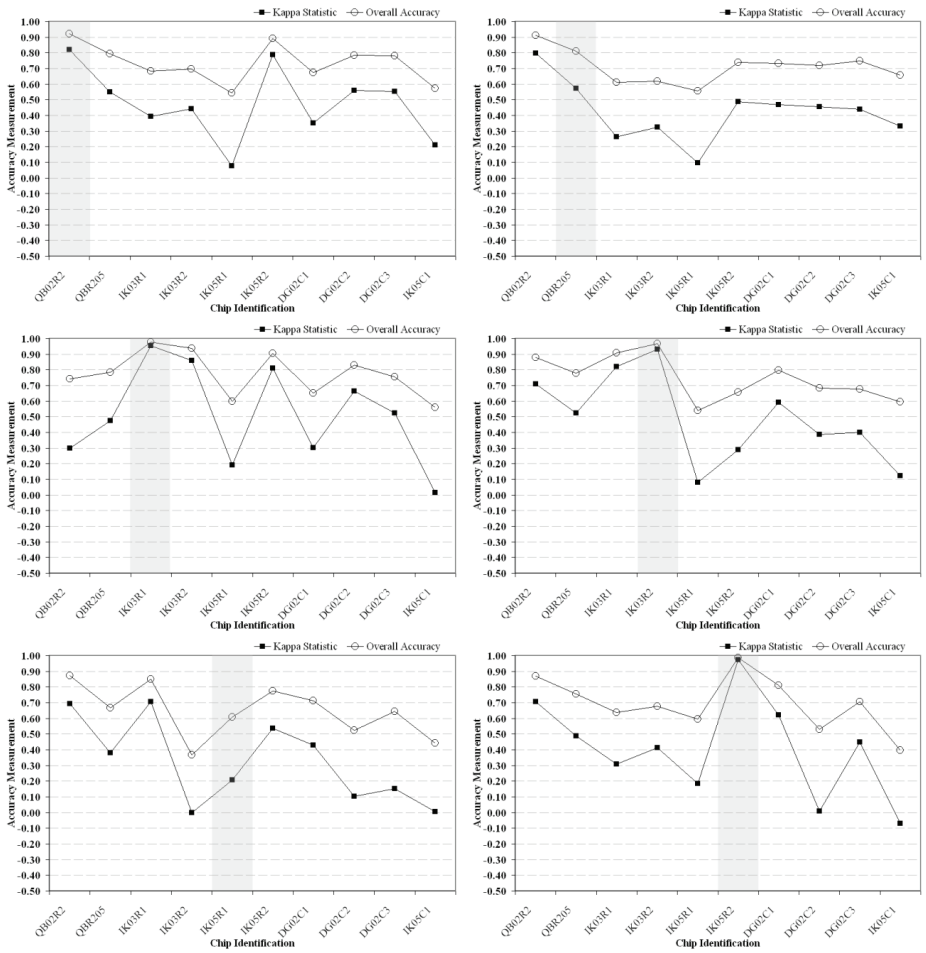


Fig. 9. Accuracy results of step 2 in the cross-evaluation of candidate solutions for imagery classification developed using the evolutionary framework. In step 2 candidate solutions were developed to spectrally classify individual pixels selected in step 1 as either asphaltic rooftop or other asphalt-based material. The shadowed vertical bar represents the image chip used for training (developing the candidate solutions) and the remaining points are the testing results when using the candidate solution developed using the training image.

The highest variability was found in stage 3, image object classification based on geometric descriptors (Figure 10), where no apparent pattern could be identified. Kappa statistics values were found in the "very good to excellent" agreement beyond the random chance of agreement range (>0.75), according to Landis and Kock (Landis & Kock, 1977), only for the image chips used in the training phase. With exception of isolated cases, results indicated a weak generalization capability of the system.

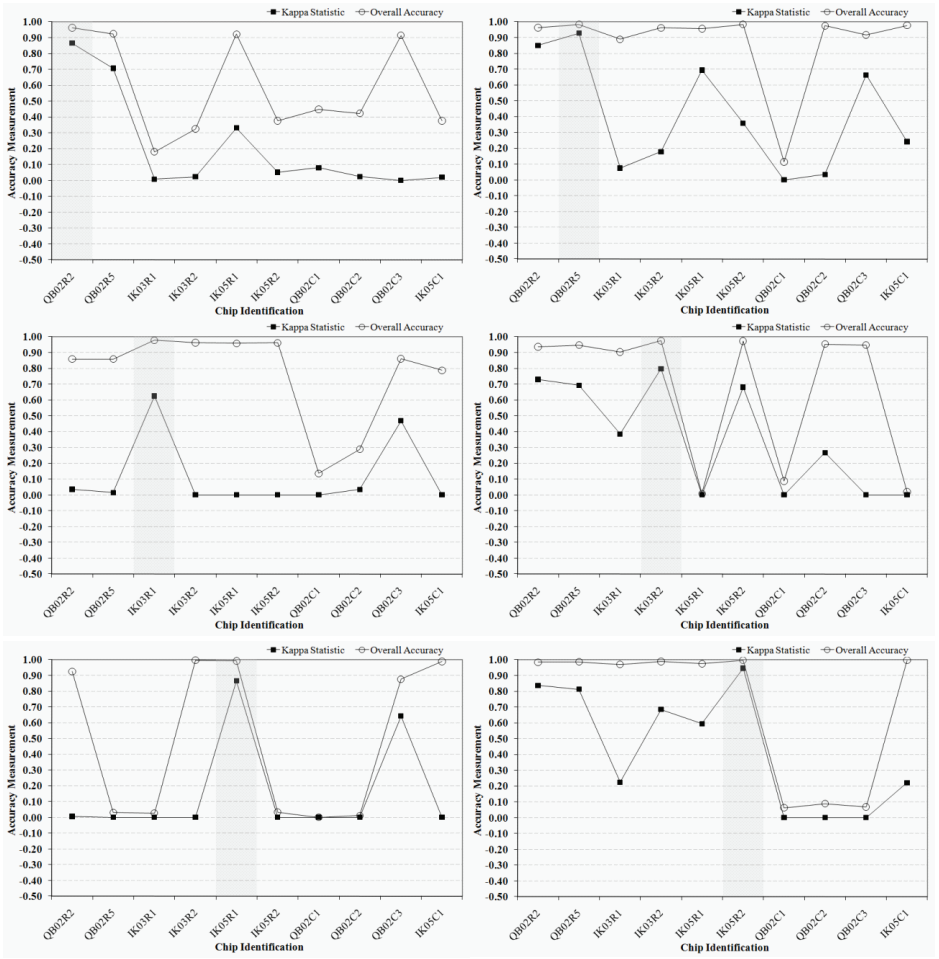


Fig. 10. Accuracy results of step 3 in the cross-evaluation of candidate solutions for imagery classification developed using the evolutionary framework. In step 3 candidate solutions were developed to geometrically identify individual group of connected pixels, classified in step 2, as single family residential rooftop. The shadowed vertical bar represents the image chip used for training (developing the candidate solutions) and the remaining points are the testing results when using the candidate solution developed using the training image.

The poor generalization ability found in Stage 3 could be partially attributed to the large variability in rooftops shapes (Figure 11) causing a direct impact on the shape descriptors. Image chips presented different pixel sizes as result of differences in sensors spatial resolution and image pre-processing procedures. For example, image object shown in Figure 11 boxes 1 and 4 have a nominal spatial resolution of 0.7 meters, boxes 3 and 6 1.0 meter, and boxes 2 and 5 2.8 meters. Additional variations in image object shapes were caused by partially coverage of rooftops by tree canopy (box 4 and 5) and roof damage caused by hurricane winds (box 3 missing shingles and box 6 half of the roofing material was removed).

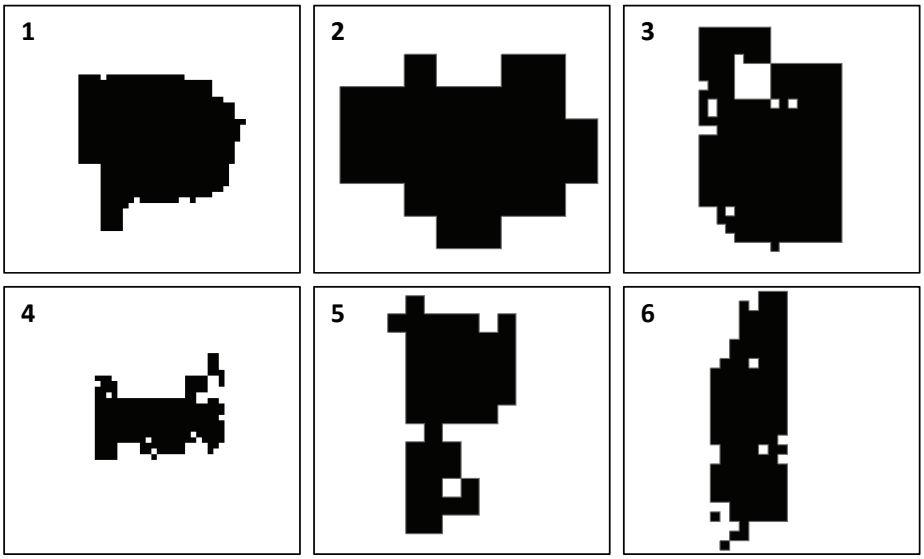


Fig. 11. Image objects representing single family residential rooftop extracted using the evolutionary framework. Image objects displayed illustrate the large variability in geometric properties due to factors such as sensor spatial resolution, rooftop partially covered by tree canopies (boxes 4 and 5), and damaged rooftops (boxes 3 and 6).

6. Conclusions

In this chapter we evaluated the robustness of an evolutionary framework in the task of feature extraction from remotely sensed imagery. The proposed system integrated standard imagery processing algorithms with genetic programming to evolve non-linear mathematical transformations to convert the original imagery into transformed images to aid in the discrimination of the material/feature of interest. The task selected was the identification of residential single-family rooftops from several image chips produced from scenes acquired with different sensors and at different dates and locations. Robustness was quantitatively assessed by training the system in one image chip and testing the evolved solutions in the remaining image chips.

The overall task of identifying rooftops from several image chips was addressed by dividing it into three sub-stages: two focused on spectral characteristics and one focused on geometrical characteristics. The multi-stage approach permitted the breakdown of a

complex problem into three simpler and smaller problems. In each sub-task a more specialized solution was evolved by the genetic programming algorithm as its performance could be assessed specifically for that sub-task. Because the results from one stage were used as input for the subsequent stage, the success of individual stages becomes significant to the overall success of the system. Additionally, the multi-stage approach helped identify possible limitations and areas of improvement of the system.

Although environmental and physical factors, such as environmental conditions, date of acquisition of the scenes, sensor resolutions, and others, influenced the robustness of the system in all stages; the main discrepancy of the results were found in the third stage (geometric properties). In the first two stages (focused on spectral information), results indicated a good agreement between sensors and a small impact of seasonal differences, illumination (radiance received at the sensor), and level of pre-processing of the scenes. The limited generalization ability demonstrated by the third stage can be partially attributed to the geometric properties dependency on sensor's spatial resolution (pixel size), type of subdivision (rooftop geometry), tree canopy coverage of rooftop, and level of rooftop damage. The complexity and size of the search space when these properties are combined limited the ability of genetic programming to evolve general solutions.

Once the development stage is completed, the operational stage has a small computational cost allowing it to be applied in a large number of scenes. The evolutionary framework was able to evolve useful non-linear transformations that provides tools to expedite the information extraction from large amounts of data, despite the limited generalization capability demonstrated by the geometrical stage. For improved generalization of geometrical stages, we advise the use of images collected with similar sensor's spatial characteristics and identification of features with similar shape properties in both training and testing images.

Future work includes the addition of more stages to investigate ontology (relationship between image objects). This relationship is inherent to the human's perception of how features should look like. For instance, the human analyst knows that a house may be connected to the road by a concrete driveway and that houses occur within a certain distance of roads. The same concept could be carried out to produce computer programs to replicate our spatial relationship perception ability. Topological functions such as "connected", "compact", "continuity", "close", "contained", and others could be defined and implemented to be used in the subsequent stages. Evolutionary algorithms could be used as the optimization tool to generate the most appropriate ontology representation of the feature of interest, using the same optimized learn-from-examples schema used herein.

7. Acknowledgements

This research was funded by the United States Department of Homeland Security-sponsored Southeast Regional Research Initiative (SERRI) at the United States Department of Energy's Oak Ridge National Laboratory.

8. References

Aronoff, S. 2005. Remote Sensing for GIS Managers, ESRI.

Baatz, M., Benz, U., Dehghani, S., Heynen, M., Höltje, A., Hofmann, P., Lingenfelder, I., Sohlbach, M. M. M., Weber, M. & Willhauck, G. (2000). *eCognition Professional User Guide 4*, Defines Imaging, Munich, Germany.

Cohen, J. 1960. A coefficient of agreement for nominal scales, *Educational and Psychological Measurement* 20(1):37–46.

Daida, J. M., Hommes, J. D., Ross, S. J. & Vesecky, J. F. (1995). Extracting curvilinear features from synthetic aperture radar images of arctic ice: algorithm discovery using the genetic programming paradigm, *Proceedings of the 1995 International Geoscience and Remote Sensing Symposium. Part 1 (of 3), Jul 10-14 1995*, Vol. 1 of *International Geoscience and Remote Sensing Symposium (IGARSS)*, Firenze, Italy, pp. 673–675.

Daida, J. M., Onstott, R. G., Bersano-Begey, T. F., Ross, S. J. & Vesecky, J. F. (1996). Ice roughness classification and ers sar imagery of arctic sea ice: evaluation of feature-extraction algorithms by genetic programming, *Proceedings of the 1996 International Geoscience and Remote Sensing Symposium. Part 3 (of 4), May 28-31 1996*, Vol. 3 of *International Geoscience and Remote Sensing Symposium (IGARSS)*, IEEE, Piscataway, NJ, USA, Lincoln, NE, USA, pp. 1520–1522.

Durand, N., Derivaux, S., Forestier, G., Wemmert, C., Gancarski, P., Boussaid, O. & Puissant, A. (2007). Ontology-based object recognition for remote sensing image interpretation, *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, pp. 472–479.

Easson, G. & Momm, H. G. (2010). Evolutionary computation for remote sensing applications, *Geography Compass* 4(3): 172–192.

Fogel, D. B. (2000). Evolutionary Computation: Toward a New Philosophy of Machine Learning, second edn, IEEE Press, New York, NY, USA.

Forestier, G., Derivaux, S., Wemmert, C. & Gançarski, P. (2008). An evolutionary approach for ontology driven image interpretation, *in* M. Giacobini, A. Brabazon, S. Cagnoni, G. Di Caro, R. Drechsler, A. Ekárt, A. Esparcia-Alcázar, M. Farooq, A. Fink, J. McCormack, M. O'Neill, J. Romero, F. Rothlauf, G. Squillero, A. Uyar S. Yang (eds), *Applications of Evolutionary Computing*, Vol. 4974 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 295–304.

Howard, D. & Roberts, S. C. (1999). Evolving object detectors for infrared imagery: a comparison of texture analysis against simple statistics, *Evolutionary Algorithms in Engineering and Computer Science*, Jyvaskyla, Finland, pp. 79–86.

Jensen, J. R. (1996). *Introductory Digital Image Processing: a Remote Sensing Perspective*, Prentice Hall, Upper Saddle River, NJ, USA.

Laben, C. & Brower, B. (2000). Process for enhancing the spatial resolution of multispectral imagery using pan-sharpening. US Patent 6,011,875.

Landis, J. & Kock, G. G. (1977). The measurement of observer agreement for categorical data, *Biometrics* 33(1): 159–174.

Lillesand, T. M. & Kiefer, R. W. (2000). *Remote Sensing and Image Interpretation*, John Wiley & Sons, Inc., New York, NY, USA.

Mitchell, T. M. (1997). *Machine Learning*, The McGraw-Hill Company, Inc., Singapore.

Momm, H., Easson, G. & Kuszmaul, J. (2009). Evaluation of the use of spectral and textural information by an evolutionary algorithm for multi-spectral imagery classification, *Computers, Environment and Urban Systems* 33(6): 463 – 471. Spatial Data Mining-Methods and Applications.

Momm, H. G. & Easson, G. (2010). Population restarting: a study case of feature extraction from remotely sensed imagery using textural information, *GECCO '10: Proceedings*

of the 12th annual conference on Genetic and evolutionary computation, ACM, New York, NY, USA, pp. 973–974.

Momm, H. G., Easson, G. & Kuszmaul, J. (2007). Integration of logistic regression and genetic programming to model coastal louisiana land loss using remote sensing, *ASPRS*, Tampa, FL, USA.

Momm, H. G., Easson, G. & Kuszmaul, J. (2008). Uncertainty analysis of an evolutionary algorithm to develop remote sensing spectral indices, *in* J. T. Astola, K. O. Egiazarian E. R. Dougherty (eds), *Image Processing: Algorithms and Systems VI*, Vol. 6812, SPIE, pp. 68120A.1–68120A.9.

Momm, H. G., Easson, G. & Wilkins, D. (2006). Genetic programming as a preprocessing tool to aid multi-temporal imagery classification, *Proceedings of the ASPRS 2006 Annual Conference*, Reno, Nevada, USA.

Momm, H. G., Gunter, B. & Easson, G. (2010). Improved feature extraction from high-resolution remotely sensed imagery using object geometry, *in* S. S. Shen P. E. Lewis (eds), *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Vol. 7695, SPIE, p. 76951C.

Munyati, C. (2000). Wetland change detection on the kafue flats, zambia, by classification of a multitemporal remote sensing image dataset, *International Journal of Remote Sensing* 21(9): 1787–1806.

Niemeyer, I. & Canty, M. J. 2001. Knowledge-based interpretation of satellite data by object-based and multi-scale image analysis in the context of nuclear verifcation, *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, IGARSS*, Sydney, Australia.

Puissant, A., Durand, N., Sheeren, D., Weber, C. & Gançarski, P. (2007). Urban ontology for semantic interpretation of multi-source images, *2nd Workshop on ontologies for urban development: conceptual models for practitioners*, Turin, Italy.

Quackenbush, L. J. (2004). A review of techniques for extracting linear features from imagery, *Photogrammetric Engineering & Remote Sensing* 70(12): 1383–1392.

Smeulders, A., Worring, M., Santini, S., Gupta, A. & Jain, R. 2000. Content-based image retrieval at the end of the early years, IEEE Transactions on pattern analysis and machine intelligence pp. 1349–1380.

23

Evolutionary Feature Subset Selection for Pattern Recognition Applications

G.A. Papakostas, D.E. Koulouriotis, A.S. Polydoros and V.D. Tourassis
Democritus University of Thrace, Department of Production Engineering and Management
Greece

1. Introduction

A crucial part of a typical pattern recognition system is the extraction of the appropriate information that uniquely describes the patterns under processing. This information has the form of vectors and their contents are called *features*, which are constructed by specific extraction methods (*Feature Extraction Methods - FEMs*). The length of the extracted feature vectors may take high dimension by incorporating many features for each pattern, although this huge information may be redundant and in a lot of cases this extra information corrupts the separability of the patterns under recognition.
Therefore the need of an additional pre-processing method that reduces the feature vectors' dimension, by selecting the most appropriate features, subject to some performance indices (class separability, high classification error etc.) is necessary. This procedure is called *dimensionality reduction* or *feature subset selection* and has attracted the attention of the scientific community for the last thirty years (Molina et al., 2002).
This chapter is focused on the usage of evolutionary methods in selecting the appropriate feature subset from a pool of features, in a way the resulted subset increases the recognition rates in several benchmark pattern recognition problems. A simple genetic algorithm is used to examine the usefulness of a predefined feature set of some benchmark problems from the literature and some useful conclusions about the ability of these features to recognize the patterns are drawn.
Moreover, the dependency of the resulted feature subsets, as far as their classification abilities are concerned, on the form of the fitness function used to measure the appropriateness of the candidate solutions, constructed by the genetic algorithm, is studied in this chapter. Three fitness functions with different properties are examined and their performance is compared to each other, for a set of pattern recognition problems.

2. Feature subset selection

Feature subset selection plays an important role in any pattern recognition system where the knowledge about the problem under consideration has to be modelled by appropriate data derived by the problem's environment. Since these data may have very high dimension the presence of irrelevant or redundant information causes significant disorders in the whole recognition system.

First of all, the usage of massive data collections reduces significantly the training and evaluating performance of the *mining* or *classifying* stage of the recognition procedure. Therefore there is a need to keep these data as little as possible without loosing useful information about the problem to solve.
On the other hand the presence of imprecise features can cause the misrepresentation of the knowledge which affects the generalization capabilities of the decision making module.
From the above it is obvious that there is a need of a mechanism that analyses the entire data collection and forms the *optimal* feature subset, according to the following proposition, in terms of some performance indices.

Proposition: A feature subset is called "optimal" if it has the lowest dimension that gives the highest recognition rate simultaneously.

Many algorithms that attempt to find this optimal feature subset, in many disciplines, have been proposed in the past. Generally, there are three main categories (Liu & Yu, 2005) of feature selection methods: 1) *wrapper* (Talavera, 2005) methods, where a search mechanism evaluates candidate feature subsets by applying them to a specific *classification* model, 2) *filter* (Marono et al., 2007) methods, where the candidate subsets are evaluated without the presence of the mining model (they are independent of the classification model), instead the internal data properties/characteristics (dependency, correlation etc.) are measured and 3) *hybrid* (Das, 2001; Jashki et al., 2009) methods which make use of both *filter* and *wrapper* mechanisms by collaborating them in different steps.
Ideally, an optimal subset has to be efficient, independent of the presence or not of the classification stage, since the internal characteristics of the features in a pool, determine their irrelevance and redundancy. However, due to the fact that a pattern recognition procedure constitutes a multi-step procedure, where its stage might affect each other, the operational behaviour of the classifying device (classifier) has to be considered. Therefore, while the *filter* methods are converged quite quickly, their resulted feature subsets may not work appropriately when applied on the classifier. On the other hand when *wrapper* methods are applied, the convergence to an optimal subset is slow and it highly depends on the structure of the classifier.
A special case of feature selection methodologies are these methods which are making use of an evolutionary algorithm (Genetic Algorithms, Particle Swarm Intelligence, Evolutionary Strategies, etc.) as an optimization procedure with several different objective functions. Evolutionary feature subset selection has proved to be an effective selection tool, since the ability of the evolutionary algorithms to search in parallel many candidate solutions of the problem (Raymer et al., 2000; Papakostas et al. 2003, 2010; Uncu & Turkşen, 2007), guarantees their convergence to a near optimum solution subject to a performance index.
In this chapter a Simple Genetic Algorithm (SGA), without having any advanced mechanism to prevent possible premature convergence to a problem solution is used, in order to optimize specific performance indices called *objective functions*. The presented algorithm is examined under three different configurations regarding the used objective function, the nature of which gives to the algorithm the characterization of *filter* or *wrapper*.

3. Genetic Algorithms (GAs)

Genetic Algorithms (GAs) have played a major role in many applications of the engineering science, since they constitute a powerful tool for optimization. A simple genetic algorithm is

a stochastic method that performs searching in wide search spaces, depending on some probability values. For these reasons it has the ability to converge to the global minimum or maximum, depending on the specific application and to skip possible local minima or maxima, respectively.

The main idea in which GAs are based, was first inspired by (Holland, 2001). He tried to find a method to mimic the evolutionary process that characterizes the evolution of living organisms. This theory is based on the mechanism qualified by the survival of the fittest individuals over a population. In fact, there are some specific procedures taking place until the predominance of the fittest individual.

In the sequel, terminology in the field of genetic methods for optimization and searching purposes is given (Coley, 2001):

- *Individual (Chromosome)* is a solution of a problem satisfying the constraints and demands of the system in which it belongs.
- *Population* is a set of candidate solutions of the problem (chromosomes), which contains the final solution.
- *Fitness* is a real number value that characterizes any solution and indicates how proper the solution for the problem under consideration is.
- *Selection* is an operator applied to the current population, in a manner similar to the one of natural selection found in biological systems. The fitter individuals are promoted to the next population and poorer individuals are discarded.
- *Crossover* is the second operator that follows the *Selection*. This operator allows solutions to exchange information, in the same way the living organisms use in order to reproduce themselves. Specifically two solutions are selected to exchange their sub-strings from a single point and after, according to a predefined probability Pc. The resulting offsprings carry some information from their parents. In this way new individuals are produced and new candidate solutions are tested in order to find the one that satisfies the appropriate objective.
- *Mutation* is the third operator applied to an individual. According to this operation a single bit of an individual binary string can be flipped with respect to a predefined probability Pm.
- *Elitism* is the procedure according to which, the fittest individual of each generation is ensured to be maintained in the next generation.

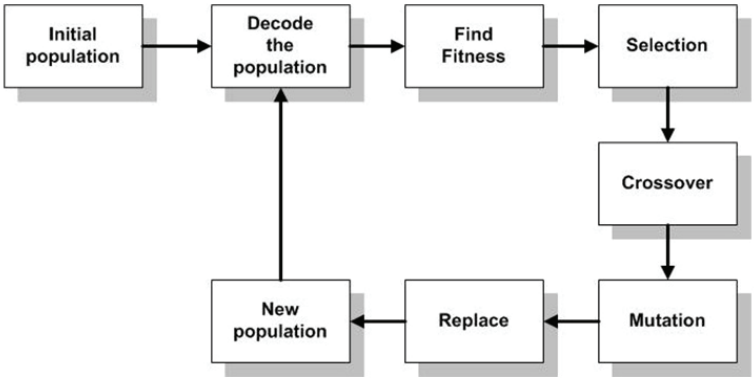


Fig. 1. Block diagram of a simple genetic algorithm.

After the application of these operators to the current population, a new population is formed and the generational counter is increased by one. This process will continue until a predefined number of generations is attained or some form of convergence criterion is met.
A Simple Genetic Algorithm (SGA), which uses some of the operations discussed above, is presented in the above Fig.1.
The usage of the SGA, depicted in Fig.1, in selecting the suitable feature subset for several benchmark datasets, for different objective functions is studied in the next section.

3.2 GA-based selection

The most computational blocks of the above Fig.1 are independent of the application where the GA is applied. Only the coding/decoding of the population and the fitness calculation depend on the problem under solution.
The application of the GA as feature subset selection algorithm involves the appropriate representation of the problem solution as chromosome structure of the algorithm's population. Since the main goal of this work is to investigate the ability of each feature to describe the classes, the GA is repeatedly applied for all possible number of features from 1 to the maximum number. Therefore the algorithm's chromosomes take variable length equal to the predefined number of features needed.
In this way the general form of the *m* chromosomes of a population, where *n* features are searched, is as follows:

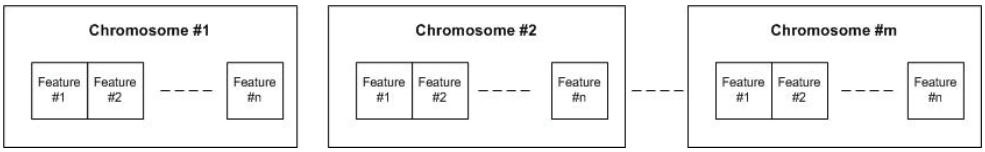


Fig. 2. Block diagram of chromosomes structure.

In the above chromosome representation, *n* is equal to the number of the features being searched. For example, if the best 2 features are needed *n* is equal to 2 and each feature in the chromosome is labelled with a value (binary or real) lying inside the feature range of the pool. Moreover, appropriate handling to avoid multiple copies of the same feature to be included in the same chromosome, is required.
The next processing stage, after the chromosome coding, is the determination of the objective (also defines the fitness of the candidate solutions) function, which is application dependent and constitutes the representation of the problem being optimized. The correct definition of the objective function is an important procedure since it has to fully describe the desired behaviour of the system.
For pattern recognition purposes a simple objective function is the *classification rate* yielded when a set of features are selected. In order to compute the classification rate, a specific classifier is needed, so this version of GA-based selection belongs to the *wrappers* algorithms category. This first objective function has the following form:

$$\underset{\#1}{\textit{Objective Function}} \qquad ObjFunc_1 = \frac{Number\ of\ incorrectly\ classified\ samples}{Total\ number\ of\ samples} \tag{1}$$

The *Minimum Distance* classifier (Kuncheva, 2004) is used to compute the classification performance of the candidate feature subsets. This classifier operates by measuring the distance of each sample from the patterns that represent the classes' centroid. The sample is decided to belong to the specific class having the less distance from its pattern.
Since the performance of the classifier is highly dependent on the specific metric used to measure the distance of the samples from the classes, five well-known distances from the literature (Papakostas et al., 2010), the *Euclidean*, *Logarithmic*, *Correlation Coefficient*, *Discrimination Cost* and *Hausdorff* distancrs are selected and presented in the following:

Euclidean Distance

$$d_1(\mathbf{p},\mathbf{s}) = \sqrt{\sum_{i=1}^{n}(p_i - s_i)^2} \tag{2}$$

Logarithmic Magnitude Distance

$$d_2(\mathbf{p},\mathbf{s}) = \sqrt{\sum_{i=1}^{n}\left(\log|p_i| - \log|s_i|\right)^2} \tag{3}$$

Correlation Coefficient

$$d_3(\mathbf{p},\mathbf{s}) = \frac{\sum_{i=1}^{n} p_i s_i}{\left|\sum_{i=1}^{n}(p_i)^2\right|^{1/2}\left|\sum_{i=1}^{n}(s_i)^2\right|^{1/2}} \tag{4}$$

Discrimination Cost Function

$$d_4(\mathbf{p},\mathbf{s}) = \sum_{i=1}^{n}\left[\frac{\max(p_{vi}, s_{ki})}{\min(p_{vi}, s_{ki})} - 1\right]^2 \tag{5}$$

Hausdorff Distance

$$\begin{aligned} d_5(\mathbf{p},\mathbf{s}) &= \max\left(h(\mathbf{p},\mathbf{s}), h(\mathbf{s},\mathbf{p})\right) \\ h(\mathbf{p},\mathbf{s}) &= \max_{p\in\mathbf{p}} \min_{s\in\mathbf{s}} \|p - s\| \end{aligned} \tag{6}$$

The above formulas measure the distance between two vectors $\mathbf{p} = [p_1, p_2, p_3, ..., p_n]$ $\mathbf{s} = [s_1, s_2, s_3, ..., s_n]$, which are defined in the $\mathbb{R}^n$ space.
It has to be remarked that the above measures tend to 0 for the case of two equal vectors, except d_3 which gives 1, since it counts the similarity of the two vectors.
Finally, when the GA-based feature selection method is used, these measures are treated as objective functions aimed to being minimized (d_1, d_2, d_4, d_5) or maximized (d_3). Noted that a maximization problem can be transformed to a minimization one, by minimizing the opposite objective function (-*F* instead *F*).
The second examined objective function describes the internal relationships of the feature vectors describing each class and is based on the *Pearson Correlation Coefficient* (Wikipedia). This function measures the *within-class* and *between-class* correlation of the feature vectors belonging to the same class and the feature vectors of different classes respectively. In a similar way as in the case of *Fisher Criterion*, the objective is the maximization of the following quantity.

Objective Function #2

$$ObjFunc_2 = \frac{r_w}{r_b} \tag{7}$$

where r_w is the within-class Pearson correlation coefficient defined as

Within-class correlation

$$r_w = \frac{1}{C_{\max}} \sum_{c=1}^{C_{\max}} r_c \tag{8}$$

with

$$r_c = \frac{1}{N_c} \sum_{i=1}^{N_c} r_{i,c} \tag{9a}$$

$$r_{i,c} = \frac{1}{N_c - 1} \sum_{j=1, i \neq j}^{N_c} r_{ij} \tag{9b}$$

and r_b is the between-class Pearson correlation coefficient defined as

Between-class correlation

$$r_b = \frac{1}{(C_{\max} - 1)^2} \sum_{i=1}^{C_{\max}} \sum_{j=1, i \neq j}^{C_{\max}} r_i^j \tag{10}$$

with

$$r_c^d = \frac{1}{N_d} \sum_{i=1}^{N_d} r_{i,c}^d \tag{11a}$$

$$r_{i,c}^d = \frac{1}{N_c} \sum_{j=1}^{N_c} r_{ij} \tag{11b}$$

In the above equations (8)-(11), C_{max} is the number of classes and N_C the number of samples belonging to the class C.

The maximization of the $ObjFunc_2$, indicate the existence of the appropriate feature vectors which guarantee high correlation between the vectors that describe the same class and low correlation between the vectors of different classes. The GA-based selection procedure which uses the function defined in (7) as the objective being optimized, constitutes a *filter* selection method, since it is independent from the classifier device used to take the final decision.

The third objective function which is studied in this investigation, is a hybrid function formed by the combination of the previous two functions $ObjFunc_1$ and $ObjFunc_2$, according to the following weighted combination rule.

Objective Function #3

$$ObjFunc_1 = w_1 \times ObjFunc_1 + w_2 \times ObjFunc_2 \tag{12}$$

where the weights w_1 and w_2, controls the importance of each objective function regarding their ability to describe the problem under process.

The definition of (12) corresponds to a generalized formula of an objective function, where the functions of (1) and (7) are special cases derived from (12), by setting w_2=0 and w_1=0, respectively.

Although, a separate study on the appropriate selection of the w_1, w_2 weights is needed in order to improve the overall feature selection procedure, the same value of 0.5 is selected for the experiments, by giving the same degree of importance to the two objective functions.

It is important to notice that the GA-based selection scheme has the advantage to permit the usage of non differentiable functions as objective functions, in contrast to the gradient-based optimization methodologies working only with differentiable error functions.

4. Experimental study

For experimental purposes, several well-known benchmark datasets from the pattern classification research field are selected, where the usefulness of their default number of features are examined, according to the GA-based selection schemes presented in the previous section.
The experimental benchmarks are widely used in the literature and are selected from the UCI repository (UCI-Machine Learning Repository), with their properties being summarized in the following Table 1.

Dataset	Features	Instances	Classes
Iris	4	150	3
Wine	13	178	3
Pima Indians Diabetes	8	768	2
Thyroid	5	215	3
Parkinson	22	195	2
Hepatitis	19	155	2
Glass	9	214	6

Table 1. Properties of benchmark datasets.

In all the experiments the GA operates with the configuration shown in Table 2, although each classification problem needs its own configuration in order to achieve the best performance. However, this common GA configuration does not affect significantly the selection procedure, since even taking into account a suboptimal solution, significant conclusions can be drawn, which can further be improved by appropriate algorithm's calibration.

Parameter	Value
Population Size	10
Variables Range	[1,n] n: number of features
Maximum Generations	100
Elitism	YES, 2 chromosomes
Crossover Points	2 points
Crossover Probability	0.8
Mutation Probability	0.001
Selection Method	Uniform Selection

Table 2. Simple Genetic Algorithm settings.

Three feature selection experiments are arranged, where the GA optimization scheme used to select the best features by using the three objective functions defined in (1), (7) and (12). For a predefined desired number of features, the algorithm returns the best features' combination for each one of the datasets and the corresponding formed vectors are presented in the following sections.

4.1 Experiment 1 – 1st objective function *(a wrapper case)*

In the first experiment the objective function of equation (1) is applied, as a fitness measure of each candidate feature subset, while the five metric distances (2)-(6) is used as minimum distance classification module. Since the first objective function measures the classification performance of the candidates feature subsets, there is a need to define the representative patterns that best characterize the classes' distributions in each benchmark problem. These patterns correspond to the classes' centers and are decided by taking into account a specific part of the entire dataset, called *training set*. In fact three different data collections are used in this experiment, the 25%, 50% and 75% randomly selected samples of each dataset, while the rest samples, called *testing set*, in each case are used for evaluation purposes. Moreover, each execution of the GA-based selection has been repeated 10 times in order to extract more statistically corrected results and the corresponding mean values are summarized in the following Tables (3)-(9) (for the case of 50% training data samples).

Iris Dataset					
Metric	$\mathbf{d_1}$	$\mathbf{d_2}$	$\mathbf{d_3}$	$\mathbf{d_4}$	$\mathbf{d_5}$
Best Feature Subset	3,4	1,4	4	1,4	3,4
Objective Value	0.973	0.973	0.960	0.973	0.973
All Features Objective Value	0.893	0.973	0.866	0.973	0.893

Table 3. Selection results for the Iris dataset.

Wine Dataset					
Metric	$\mathbf{d_1}$	$\mathbf{d_2}$	$\mathbf{d_3}$	$\mathbf{d_4}$	$\mathbf{d_5}$
Best Feature Subset	1,2,3,7,9,10,12	3,4,5,7,10,11, 12,13	2,6,7,10,12	1,3,4,5,7,10, 11,12,13	1,2,3,7,9,10,12
Objective Value	0.933	0.988	0.900	0.988	0.933
All Features Objective Value	0.700	0.933	0.711	0.933	0.700

Table 4. Selection results for the Wine dataset.

Pima Indians Diabetes Dataset					
Metric	$\mathbf{d_1}$	$\mathbf{d_2}$	$\mathbf{d_3}$	$\mathbf{d_4}$	$\mathbf{d_5}$
Best Feature Subset	1,3,5,7	1,5,7	1,7	1,5,7	1,3,5,7
Objective Value	0.750	0.760	0.742	0.763	0.750
All Features Objective Value	0.679	0.555	0.713	0.554	0.679

Table 5. Selection results for the Pima Indians Diabetes dataset.

Thyroid Dataset					
Metric	**d_1**	**d_2**	**d_3**	**d_4**	**d_5**
Best Feature Subset	2,3	2,3,4	3,4,5	2,3,4	2,3
Objective Value	0.906	0.943	0.915	0.943	0.906
All Features Objective Value	0.850	0.846	0.710	0.887	0.850

Table 6. Selection results for the Thyroid dataset.

Parkinson Dataset					
Metric	**d_1**	**d_2**	**d_3**	**d_4**	**d_5**
Best Feature Subset	17,20,22	1,16,21,22	17,20,22	1,16,21,22	17,20,22
Objective Value	0.855	0.855	0.855	0.855	0.855
All Features Objective Value	0.721	0.690	0.701	0.690	0.721

Table 7. Selection results for the Parkinson dataset.

Hepatitis Dataset					
Metric	**d_1**	**d_2**	**d_3**	**d_4**	**d_5**
Best Feature Subset	5,10,12,14,17,19	1,10,12	2,12,13,14	1,3,12	3,5,10,12,14,17,19
Objective Value	0.870	0.860	0.870	0.860	0.870
All Features Objective Value	0.545	0.652	0.584	0.652	0.545

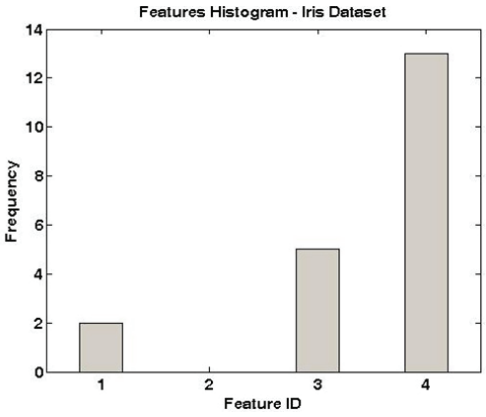
Table 8. Selection results for the Hepatitis dataset.

Glass Dataset					
Metric	**d_1**	**d_2**	**d_3**	**d_4**	**d_5**
Best Feature Subset	2,4,8,9	2,4,7	2,3,4,7,8	2,4,7	2,4,8,9
Objective Value	0.458	0.458	0.440	0.458	0.458
All Features Objective Value	0.403	0.247	0.357	0.247	0.403

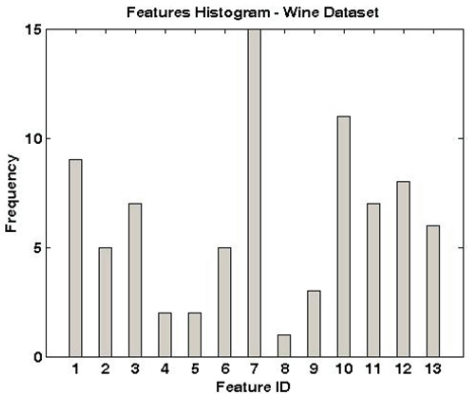
Table 9. Selection results for the Glass dataset.

An important conclusion is drawn from the above tables, about the usefulness and information redundancy of the nominal features describing all the benchmark datasets. In all the cases there is a feature vector with lower dimension and higher objective value than the corresponding vectors consisting of all the features. This means that the classification performance can be significantly improved by using a small feature subset, while the usage of all the features does not guarantee better classification results. Therefore, by applying a classification-driven dimensionality reduction mechanism based on GA selection scheme, only the most essential features are kept.

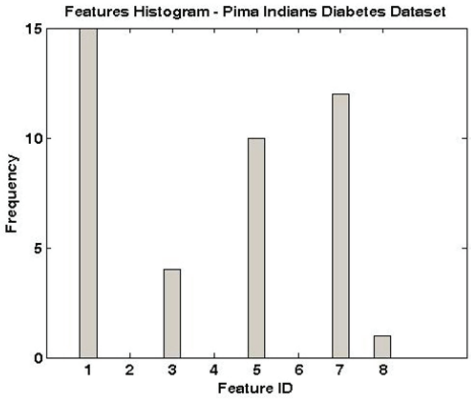
Moreover, the above tables show the ability of each metric distance (d_1)-(d_5) to measure the real distance between the data points and the corresponding classes' centers. Due to the fact that none of the above distance shows significant superior performance over the rest ones, an additional statistical analysis, which counts the frequency of the features including in the best feature subsets for all distances and training sets (25%,50%,70%), is applied and the resulted histograms are illustrated in the following Fig.3.


Features Histogram - Iris Dataset
Frequency
Feature ID

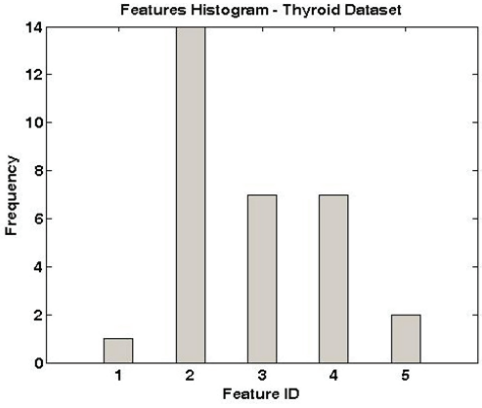
(a)


Features Histogram - Wine Dataset
Frequency
Feature ID

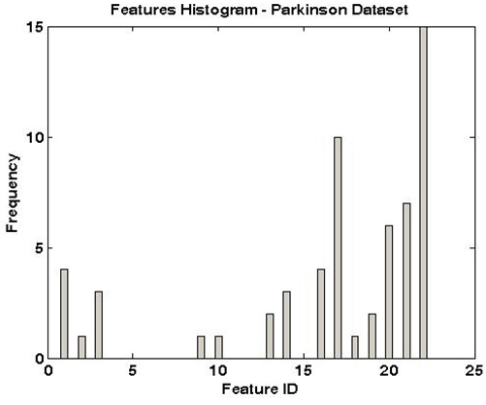
(b)


Features Histogram - Pima Indians Diabetes Dataset
Frequency
Feature ID

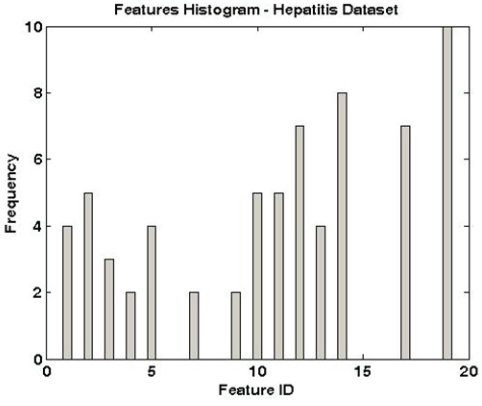
(c)


Features Histogram - Thyroid Dataset
Frequency
Feature ID

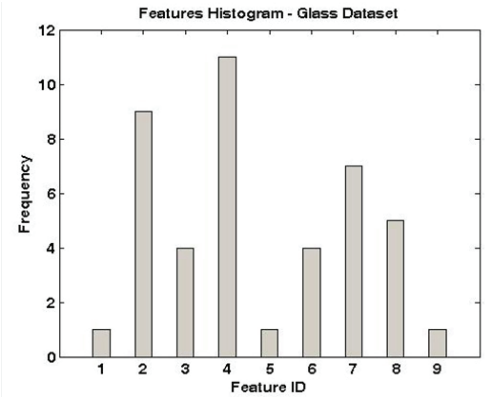
(d)


Features Histogram - Parkinson Dataset
Frequency
Feature ID

(e)


Features Histogram - Hepatitis Dataset
Frequency
Feature ID

(f)



(g)

Fig. 3. Features histograms for (a) Iris, (b) Wine, (c) Pima Indians Diabetes, (d) Thyroid, (e) Parkinson, (f) Hepatitis and (g) Glass datasets.

By analysing the above plots of Fig.3, the suitability of each feature in the case of all datasets, can be studied. Through these plots, the statistically most efficient feature subsets for all metric distances are constructed, as a supplementary to the GA-based feature selection mechanism. The final features subsets for each dataset are summarized in the following Table 10. It has to be noted that the usefulness of these features subsets will be studied later by using them to solve the same pattern recognition problems, with a typical feedforward neural network classifier used as the decision module.

	Datasets						
	Iris	**Wine**	**Pima Indians Diabetes**	**Thyroid**	**Parkinson**	**Hepatitis**	**Glass**
Best Feature Subset	3,4	1,2,3,6,7,10, 11,13	1,3,5,7	2,3,4	1,2,3,9,10,13, 14,16,17,18,19, 20,21,22	1,2,3,4,5,7,9, 10,11,12,13,14, 17,19	2,4,7,8

Table 10. Resulted feature subsets from the analysis of the histograms of Fig.3.

4.2 Experiment 2 – 2nd objective function *(a filter case)*

The only difference between the 2nd experiment and the 1st one is the usage of a different objective function for the evaluation of the candidate solutions fitness. The used objective function is defined in (7) and it measures the correlation degree between the data points belonging to the same class and to different classes simultaneously. This *filter* type of selection is executed in the absence of any classification module and therefore it is interesting to study the classification performance of the selected features when applied to a traditional neural network classifier.

The selected features subsets along with the performance of the entire features sets are presented in Table 11, as follows.

	Datasets						
	Iris	**Wine**	**Pima Indians Diabetes**	**Thyroid**	**Parkinson**	**Hepatitis**	**Glass**
Best Feature Subset	2,3,4	2,3,6,7 9,11,12	3,4,5,6,8	2,3,5	4,8,11,15	2,4,5,6,7,8,9, 10,11,12,13,19	1,3,4,6,8,9
Objective Value	0.765	0.623	1.507	0.895	1.273	0.835	0.683
All Features Objective Value	1.210	1.522	1.930	1.502	2.026	2.030	1.259

Table 11. Selection results using $ObjFunc_2$ for the case of all benchmark datasets.

A careful study of the above table can lead to common conclusions with that of the previous experiment. The selection procedure by using the objective function of (7), forms feature vectors of lower dimension and higher objective value, as compared with the nominal ones. This result highlights the fact that all the features are not of the same importance but moreover the counting of some features may degrade the over classification performance.
Furthermore, there are some overlaps between the subsets derived by the two experiments, something which reinforces the importance and appropriateness of the common features.
What is of major importance is the investigation of the classification capabilities of the formed subsets by using a specific classifier, in order to study the independency of the selection procedure to the applied classification structure.

4.3 Experiment 3 – 3rd objective function *(a hybrid case)*

For the purposes of the 3rd experiment, a hybrid objective function (12) that combines the two functions $ObjFunc_1$ and $ObjFunc_2$ is used. The operation of the GA in this case corresponds to a multi-objective optimization, where the weights are set both to 0.5, while an additional procedure to find the best values of them can be performed.
It has to be noted that in order to evaluate the $ObjFunc_1$, all the metric distances are used and the same statistical analysis is performed as in the case of the 1st experiment. For space saving

	Datasets						
	Iris	**Wine**	**Pima Indians Diabetes**	**Thyroid**	**Parkinson**	**Hepatitis**	**Glass**
Best Feature Subset	2,3,4	2,3,6,7,9,11,12	3,4,6,7,8	2,3,4,5	10,20,22	2,3,4,5,6,8,9, 10,11,13,19	1,3,4,6,8
Objective Value	0.402	0.411	0.928	0.503	0.782	0.545	0.620
All Features Objective Value	0.658	0.911	1.125	0.825	1.152	1.242	0.927

Table 12. Selection results using $ObjFunc_3$ (with d_1) for the case of all benchmark datasets.

	Datasets						
	Iris	**Wine**	**Pima Indians Diabetes**	**Thyroid**	**Parkinson**	**Hepatitis**	**Glass**
Best Feature Subset	2,3,4	2,3,6,7,11	3,4,5,6,8	2,3,5	10,20,22	2,3,4,5,6,8, 9,10,11,19	1,3,4,6,8,9

Table 13. Resulted feature subsets from the analysis of the corresponding features' histograms.

reasons, only the selection results of distance d_1 is presented in Table 12, while the selection results analysing the corresponding features' histograms, are summarized in Table 13.
A first look to the above selection results, leads to the conclusion that the usage of the hybrid objective function gives in some cases the same features subsets with the $ObjFunc_2$, meaning that this measurement mostly influences it, while there are cases where the formed subsets are smaller than the other two experiments. Therefore, by combining the two objective functions novel features subsets can be found that optimize both the classification rate and the correlation degrees of the feature vectors being used.
However, the study of the selected features subsets obtained by the three experiments, gives information only about the utility of the features, regarding the objective value used to evaluate them and their classification capabilities have to be investigated on the presence of the classifier module.

4.4 Feature subsets verification – A Neural Network Classifier case

As already mentioned in the previous sections, the features subsets formed by applying the GA-based selection scheme, are optimal as far as their performance is concerned, in terms of the used objective function. In the case of $ObjFunc_1$ the selection is taking into account the classification capabilities of the subsets relative to a specific classifier structure. It is worthy investigating the performance of the selected subsets under the usage of a totally different classifier module, such as the Neural Network Classifier (NNC), widely used in pattern recognition applications (Papakostas et al., 2008). This need for further study of the global behaviour of the selected features is more important in the case of the subsets derived by applying the $ObjFunc_2$, since this selection procedure takes into account inherent properties of the data samples constituting the pattern classes.
By working on this way, a typical feed-forward neural network classifier is used to verify the classification performance of the features subsets selected through the GA-based procedure, under the three different objective functions configurations.
Before the presentation of the classification configuration and results of the NNC, it is constructive to summarize the features subsets selected by the three different objective functions for all the benchmark datasets, as depicted in Table 14.
A multilayer perceptron is used as the NNC, having a different structure for each benchmark dataset. The used NNC has three layers with one hidden layer and its structure is denoted as *inputs x hidden nodes x outputs*. The number of *inputs* is equal to the number of features used to discriminate the patterns, the number of *hidden nodes* is equal to the nominal features (Table 1, 2nd column) of each dataset and the number of *outputs* is equal to the number of classes describing each dataset (Table 1, 4th column)

	Datasets						
	Iris	**Wine**	**Pima Indians Diabetes**	**Thyroid**	**Parkinson**	**Hepatitis**	**Glass**
ObjFunc$_1$ Histogram based	3,4	1,2,3,6,7,10, 11,13	1,3,5,7	2,3,4	1,2,3,9,10,13, 14,16,17,18,19, 20,21,22	1,2,3,4,5,7,9, 10,11,12,13, 14, 17,19	2,4,7,8
ObjFunc$_2$	2,3,4	2,3,6,7,9,11,12	3,4,5,6,8	2,3,5	4,8,11,15	2,4,5,6,7,8,9, 10,11,12,13, 19	1,3,4,6,8,9
ObjFunc$_3$ Histogram based	2,3,4	2,3,6,7,11	3,4,5,6,8	2,3,5	10,20,22	2,3,4,5,6,8, 9,10,11,19	1,3,4,6,8,9

Table 14. Selected features subsets by applying the three objective functions.

Subsets	**Statistics**	**Datasets**						
		Iris	**Wine**	**Pima Indians Diabetes**	**Thyroid**	**Parkinson**	**Hepatitis**	**Glass**
All Features	*min (%)*	58.53	95.55	68.48	82.24	94.84	79.22	17.11
	max (%)	98.78	100	80.46	100	98.96	94.80	78.37
	mean (%)	90.73	98.44	77.57	97.28	97.21	89.22	58.28
	std (%)	14.37	1.58	3.81	5.33	1.29	4.82	18.15
ObjFunc$_1$ Histogram based	*min (%)*	97.56	96.66	77.34	70.09	94.84	83.11	34.23
	max (%)	98.78	100	80.72	94.39	100	96.10	78.37
	mean (%)	97.68	98.88	78.90	85.79	97.42	89.61	59.45
	std (%)	0.69	1.04	1.02	9.37	1.70	3.67	16.21
ObjFunc$_2$	*min (%)*	50	37.77	65.10	70.09	67.01	85.71	21.62
	max (%)	100	92.22	71.09	99.06	79.38	92.27	76.57
	mean (%)	92.92	82.55	67.70	85.79	74.22	88.31	53.24
	std (%)	15.13	17.67	2.81	12.94	3.26	2.53	18.80
ObjFunc$_3$ Histogram based	*min (%)*	50	63.33	65.10	70.09	75.25	79.22	21.62
	max (%)	100	94.44	71.09	99.06	92.78	85.71	76.57
	mean (%)	92.92	83	67.70	85.79	86.90	81.29	53.24
	std (%)	15.13	12.62	2.81	12.94	6.57	2.81	18.80

Table 15. Classification results of the neural classifier for the entire features subsets.

Each experiment is executed 10 times in order to ensure its statistical accuracy, and the corresponding statistics (minimum (min), maximum (max), mean (mean) and standard deviation (std)), in terms of classification rate (%), by applying the feature subsets of Table 14, on the NNC are summarized in the above Table 15.

The results show the superiority of the feature subsets selected by ObjFunc$_1$, over the two other selection methods. However, the most important observation is the outperformance of these features subsets as compared with the performance of all the nominal features, which in all the cases (except for the case of Thyroid dataset) give lowest classification rates.

Another significant result that comes from the comparison of Table 15 and Tables 3-9, is the improvement (Iris: 97.30% to 97.68%, Wine: 98.88% remains the same, Pima: 76.30% to 78.90%, Parkinson: 85.55% to 97.42, Hepatitis: 87.00% to 89.61%, Glass: 45.80% to 59.45%) of the classification abilities of the subsets, when the NNC is applied as the classifier module. Therefore while a minimum distance classifier is used to select the best feature subsets, the appropriateness of the selected features is further enforced by applying a more sophisticated classifier structure in the recognition procedure.

5. Conclusion

The issue of selecting the most appropriate features describing the classes of different patterns constituting a pattern recognition application is concerned in this chapter. The presented selection procedure is based on the usage of an evolutionary algorithm, such as a Genetic Algorithm, in order to find a global optimal solution, by giving the necessary feature subsets that better separate the classes.

The advantage of the evolutionary optimization methods to enable the application of any objective function, without the need of being differentiable, gives a great flexibility in choosing this function that better describes the problem in hand.

By examining three different configurations of the GA-based selection scheme, regarding the usage of the objective function applied to measure the fitness of possible candidate solutions, some useful outcomes are obtained. The *wrapper* version of the selection method, which takes into account the classifier type used to classify the patterns, present the better performance over the other two different alternatives, but most of all the selected subsets perform better than the nominal benchmarks' features, even for the case of a different applied classifier structure.

Therefore, it is important to highlight the necessity of applying a selection procedure before the classification stage, in order to reduce the dimensionality of the features' vectors driven by the classification separability point of view. Moreover, there is a need to find suitable objective functions without the performance of $ObjFunc_1$, but with the speed of $ObjFunc_2$ independent of the applied classifier structure.

6. References

Coley, D.A. (2001). An introduction ot genetic algorithms for scientists and engineers. World Scientific Publishing.

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. *Proceedings of 18th International Conference on Machine Learning*, pp. 74-81.

Holland, J.H. (2001). *Adaptation in natural and artificial systems*. 6th Ed., MIT Press.

Jashki, M.A.; Makki, M.; Bagheri, E. & Ghorbani, A. (2009). An iterative hybrid filter-wrapper approach to feature selection for document clustering. *Advances in Artificial Intelligence*, LNS Vol. 5549, pp. 74-85.

Kuncheva, L.I. (2004). *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience Publiching.

Liu, H. & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 4, pp. 491-502.

Marono, N.S.; Betanzos, A.A. & Sanroman, M.T. (2007). Filter methods for feature selection – a comparative study. *Intelligent Data Engineering and Automated Learning*, LNS Vol. 4881, pp. 178-187.

Molina, L.C.; Belanche, L. & Nebot, A. (2002). Feature selection algorithms: a survey and experimental evaluation. *Data Mining, 2002. ICDM 2002. Proceedings of IEEE International Conference on Data Minig (ICDM '02)*, pp. 306 – 313.

Papakostas, G.A.; Boutalis, Y.S. & Mertzios, B.G. (2003). Evolutionary selection of Zernike moment sets in image processing, *Proceedings of 10th International Workshop on Systems, Signals and Image Processing (IWSSIP'03)*, September 2003, Prague – Czech Republic.

Papakostas, G.A. ; Boutalis, Y.S. ; Samartzidis, S.T. ; Karras, D.A. & Mertzios, B.G. (2008). Two-stage hybrid tuning algorithm for training neural networks in image vision applications. *International Journal of Signal and Imaging Systems Engineering*, Vol. 1, No. 1, pp. 58-67.

Papakostas, G.A.; Karakasis, E.G. & Koulouriotis, D.E. (2010). Novel moment invariants for improved classification performance in computer vision applications. *Pattern Recognition*, Vol. 43, No. 1, pp. 58-68.

Raymer, M.L.; Punch, W.F.; Goodman, E.D.; Kuhn, L.A. & Jain, A.K. (2000). Dimensionality reduction using genetic algorithms. IEEE Transactions on Evolutionary Computation, Vol. 4, No. 2, pp. 164-171.

Talavera, L. (2005). An evaluation of filter and wrapper methods for feature selection in categorical clustering. *Advances in Intelligent Data Analysis VI*, LNS Vol. 3646, pp. 440-451.

UCI-Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets.html

Uncu, O. & Turkşen, I.B. (2007). A novel feature selection approach: Combining feature wrappers and filters. *Information Sciences*, Vol. 177, No. 2, pp. 449-466.

Wikipedia http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

24

A Spot Modeling Evolutionary Algorithm for Segmenting Microarray Images

Eleni Zacharia and Dimitris Maroulis
Department of Informatics and Telecommunications, University of Athens
Greece

1. Introduction

cDNA microarrays is one of the most fundamental and powerful tools in biotechnology. Despite its relatively late discovery in 1995, it has since been utilized in many biomedical applications such as cancer research, infectious disease diagnosis and treatment, toxicology research, pharmacology research, and agricultural development. The reason for its broad use is that it enables scientists to analyze simultaneously the expression levels of thousands of genes over different samples (Leung et al., 2003).

More precisely, the process of a microarray experiment (Campbell et al., 2007) starts with the selection of a set of DNA probes that are of particular interest. A robot places the selected DNA probes on a glass slide, creating an invisible array of DNA dots. Two distinct populations of mRNAs (messenger RNAs) are then isolated from a control sample (i.e a cell developed under normal conditions) and a test sample (i.e. a cell developed under a specific treatment). The mRNA populations are reversely transcribed into cDNA (complementary DNA) populations which in turn are colored with separate fluorescent dyes of different wavelengths (i.e. Cy3 and Cy5). The dyed cDNA populations are mixed with purified water and the solution is placed on the glass slide in order for the cDNA populations to be hybridized with the slide's DNA dots. Finally, the hybridized glass slide is fluorescently scanned twice; one scan for each dye's wavelength. Hence, two digital images are produced, one for each population of mRNA. Each digital image contains a number of spots (corresponding to the DNA-cDNA dots) of various fluorescence intensities. Given that the intensity of each spot is proportional to the hybridization level of the cDNAs and the DNA dots, the gene expression information is obtained by analyzing the digital images.

As stated by Yang et al (Yang et. al, 2002), the process of analyzing a microarray image can be divided into three main phases, namely: "Gridding", "Spot-Segmentation" and "Spot-Intensity extraction". During the 1st phase, the microarray image is segmented into numerous compartments, each containing one individual spot and background. During the 2nd phase each compartment is individually segmented into a spot area and a background area, while during the 3rd phase the brightness of each spot is calculated. The expression-levels of the genes in these spots are a direct result of their individual brightness.

Amongst the stages of the microarray-image analysis, spot-segmentation remains the most challenging one. Ideally, the existing spots inside the microarray image are aligned in 2D array layouts. These 'ideal spots' also have a circular 2D shape with fixed diameters, while

their intensity peaks at their central region and declines at regions further from their centre. In reality however, microarray images have poor quality due to the existence of noise and/or artifacts as well as due to uneven background (Wang et. al, 2003). Additionally, many spots are rather different to the ideal ones as they vary in size, shape and position due to imperfect sample-preparation and hybridization processes (Tu et. al, 2002). Last but not least, some spots are so poorly contrasted that are not clearly visible (Chen et al 2006).

As a result, a number of spot-segmentation techniques have been developed, some of which have been incorporated into commercial software programs. The fixed circle segmentation algorithm [implemented by the ScanAlyze software program (Eisen, 1999)] or the adaptive circle segmentation algorithm [implemented by the Dapple software program (Buhler et. al, 2000)] assumes that microarray spots are circular. However, this assumption is in fact invalid since a spot's morphology – as previously mentioned - is not always a circle. Moreover, both of these techniques require input parameters in order to define the spot's diameter. The adaptive shape-segmentation [implemented by the Spot software program(Buckley, 2000)] has been suggested in order to deal with the various shapes of the spots. The algorithm can segment regions of irregular shapes by implementing a watershed algorithm. However, a drawback in this method is that its performance is based on the appropriately specified number and locations of the starting points (seeds). Chen et al (Chen et al, 1997) suggested a thresholding method based on the statistical Mann-Whitney test. A disadvantage of this method is that its performance relies on the appropriate choice of background samples. Clustering algorithms, such as K-means, hybrid K-means and, fuzzy C-means (FCM) have been also applied in order to determine which pixels belong to the spot area and which ones to the background area (Bozinov et. al 2002), (Rahnenfuhrer et. al, 2003), (Nagarajan et. al 2003). Nevertheless, these methods become inaccurate, when the spots are poorly contrasted or when the spots are very close to each other. In the latter case, instead of segmenting the real spot, these methods may segment portions of neighboring spots. Another segmentation method, based on the clustering of pixels' values, is the model-based segmentation algorithm, proposed by Li et al (Li et. al, 2005). A disadvantage of this method is that it may over-segment the microarray spots since the number of clusters is determined automatically. Finally, there are segmentation methods based on active contours and multiple snakes (Ho et al, 2008),(Srinark et al 2004), (Srinark et al 2001). These methods give inaccurate results when the compartment is contaminated with noise and artifacts. The Markov Random Fields method (MRF) (Demirkaya et. al, 2005) utilizes the neighboring information, along with the intensity information, based on an MRF modeling of the compartment. However, one major drawback of this method is that it requires an initial classification of the pixels which in turn affects the final results. The segmentation method included in the Matarray toolbox of Matlab (Wang et al, 2001) combines both spatial and intensity information. A disadvantage of this method is that it requires input parameters in order to segment the spots.

All aforementioned techniques require human intervention in order to define input parameters or to correct the segmentation results. This apparent lack of automation can be disadvantageous during microarray image analysis. Indeed, human intervention may inevitably modify the actual results of the microarray experiment and lead to erroneous biological conclusions. Therefore, the necessity of an accurate and automatic spot-segmentation technique becomes obvious.

In this chapter, the spot-segmentation stage of the microarray-image analysis is expressed as an optimization problem which is subsequently solved by using genetic algorithms and fuzzy logic. In particular, a genetic algorithm (GA) represents the real-spots of the cDNA

microarray image with spot-models, in a 3D space. The segmentation of the real-spots is conducted by drawing the contours of the spot-models. It should be noted that the spot-model presented in this chapter can be used for the representation of all types of real microarray spots such as peak-shaped, volcano-shaped and doughnut-shaped spots. Consequently, the proposed method can segment all possible types of microarray spots. Moreover, the genetic algorithm has been further developed in order to be noise-resistant and yield more accurate results. It adopts the Fuzzy Logic so as to take into account the uncertainties that exist in the pixels' intensities due to noise, artifacts and uneven background. Contrary to existing software systems, the proposed spot-segmentation method is fully automatic as it does not require any input parameters; it is also noise resistant and yields excellent results even under the following adverse conditions: i) the appearance of various spot-shapes, such as peak-shaped, volcano-shaped and doughnut-shaped spots, ii) the appearance of spots of diverse intensities, such as low-intensity spots or saturated spots and iii) the appearance of various spot-sizes. Last but not least, it outperforms other image analysis software programs as well as other well-known published techniques.

2. Genetic algorithms

Genetic Algorithms (GAs) are powerful, stochastic, non-linear optimization tools based on the principles of natural selection and evolution (Golderbg, 1989). Compared to traditional search and optimization tools (such as Blind Search Algorithms), GAs demonstrate superior performance, given that they are robust optimizers, suitable for solving problems for which there is little or no a priori knowledge of the underlying processes.

Given a specific optimization problem, a typical GA searches for the optimal solution as follows: Firstly, it creates a finite number of potential solutions encoded as alpha-numerical sequences called Chromosomes. These Chromosomes constitute an initial Population Pop_1. Subsequently, the GA produces a new Population Pop_2 according to the following: The Chromosomes constituting the Pop_1 are evaluated using a Fitness Function. Thereafter, the GA evolves the Population Pop_1 into a new Population Pop_2 using the three Genetic Operators: Reproduction, Crossover, and Mutation. This Evolutionary Cycle from one Population to the next (Pop_1 to Pop_2, Pop_2 to Pop_3 and so forth) continues until a specific termination criterion is satisfied. Subsequently, the essential elements of the GA are: Chromosome representation, Chromosome evaluation, the Evolutionary cycle, and the Termination criteria.

A Chromosome is often represented as a simple alpha-numerical sequence which encodes the values of variables defining a possible solution to the optimization problem at hand. Although a traditional GA uses a binary number in order to encode these variables, in the present application, a Real-Coded Genetic Algorithm (RCGA), which uses real values, is applied. The reason is that real-coded Chromosomes exhibit various advantages over binary-coded Chromosomes as they can use large or unknown domains for the variables they encode. On the other hand, assuming that the Chromosome has a fixed length, binary implementations cannot increase the domain without sacrificing precision (Herrera et al., 1998).

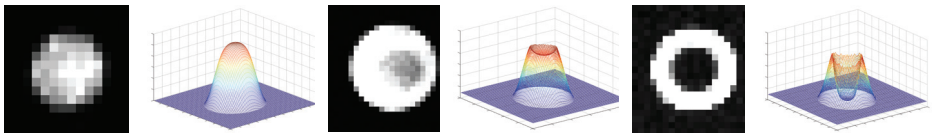
The evaluation of the Chromosome is based on a Fitness Function which assigns to the Chromosome a Fitness Value measuring the quality of the solution that the Chromosome represents. Naturally, the Fitness Function depends on the particular optimization problem at hand and on the Chromosome representation.

Reproduction, Crossover and Mutation are the three Genetic Operators used for the creation of new Chromosomes (Herrera et al., 1998). All of them have been implemented in several, distinct fashions depending on the Chromosome representation.
Common terminating criteria are: (i) A solution that satisfies the defined minimum standards, (ii) The attainment of a maximum number of Populations, (iii) The attainment of a fixed number of Populations for which the Fitness Value of the best Chromosome remains the same, and (v) Combinations of the above (Hayes, 2006).

3. Microarray spots

The following three types of microarray spots can be identified in a microarray image (Kim et al. 2007):

1. Peak-shaped spot (Fig. 1a); this type of spot has an intensity that peaks at its central region and declines at regions further from the centre. In the case when the peak is thin, the spot resembles to a 2D-Gaussian function. In the case when the peak is wide, the spot resembles to a plateau.
2. Volcano-shaped spot (Fig. 1b); this type of spot is defined as the peak-shaped spot having a small hole in the area of its peak. It therefore resembles to a volcano.
3. Doughnut-shaped spot (Fig. 1c); this type of spot has a thin rim of high intensity and a large hole of very low intensity at its central region.



(a) (b) (c)

Fig. 1. Three types of real microarray spots in 2D and 3D dimensions:(a) a peak-shaped spot, (b) a volcano-shaped spot, and (c) a doughnut-shaped spot.

4. Proposed spot-segmentation method

Given that I_{REAL} is one of the compartments of a real microarray image containing one individual spot S_{REAL} and background B_{REAL}, the segmentation procedure aims to the delineation of the boundaries of the spot S_{REAL}. The segmentation procedure is divided into two stages:
1st Stage: The compartment I_{REAL} of the microarray image is optimally represented by a 3D compartment-model I_{MODEL}.
2nd Stage: The boundaries of the microarray spot S_{REAL} are depicted by drawing the contour of the spot-model S_{MODEL}.

4.1 Morphological models for a microarray spot and its compartment

Due to the aforementioned common spots' characteristics, a microarray compartment can be represented by a 3D compartment-model, in which the third dimension represents the intensity. More precisely, a microarray spot can be represented using: i) a 3D-curve representing the main-body S_{MB} of the spot-model, and ii) a 3D-curve representing the inner-dip S_{ID} of the spot-model.

4.1.1 The spot-model and its components

The main-body and the inner-dip 3D curves have opposite orientation and they resemble the 3D Gaussian or plateau curve (Fig.2). More precisely, the main body of the spot-model $S_{MB}(x,y)$ is defined by the following equation:

$$S_{MB}(x,y) = h_{MB} \cdot \left[erf\left(a_{MB} + r_{MB}(x,y)\right) + erf\left(a_{MB} - r_{MB}(x,y)\right) \right], \quad (1)$$

Where

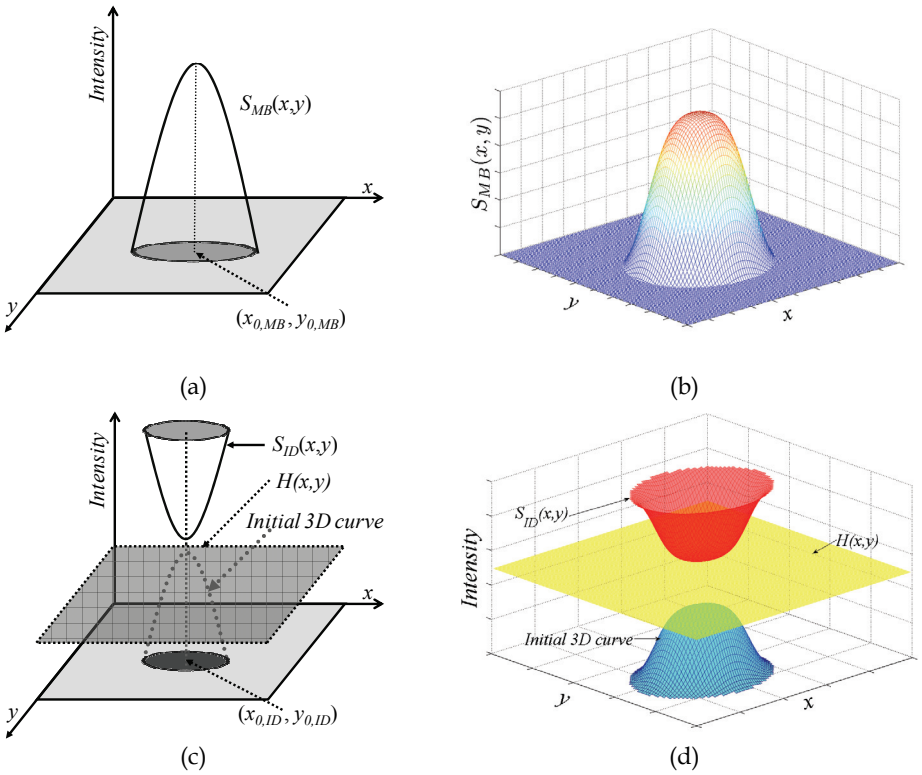


Fig. 2. Components of the spot-model: (a,b) The main-body $S_{MB}(x,y)$ of the spot-model,(c,d) 3D representation of the inner-dip $S_{ID}(x,y)$ 3D-curve, the initial 3D-curve and the horizontal surface $H(x,y)$.

$$r_{MB}(x,y) = \sqrt{\frac{(x - x_{0,MB})^2}{D_{x,MB}} + \frac{(y - y_{0,MB})^2}{D_{y,MB}}}. \quad (2)$$

h_{MB} controls the height of the main body of the spot-model. $erf(z)$ denotes the error function encountered in integrating the normal distribution. $(x_{0,MB}, y_{0,MB})$ are the coordinates of the center of the main body of the spot-model on the 2D plane. $D_{x,MB}$ and

$D_{y,MB}$ control the slope of the 3D curve at two main directions (x and y) of the 2D plane, while a_{MB} controls the shape of the 3D curve. For $a_{MB} \to 0$, $S_{MB}(x,y)$ resembles a two-dimensional Gaussian function, while for $a_{MB} \to \infty$, $S_{MB}(x,y)$ resembles a plateau or saturated spot. A more detailed illustration of $S_{MB}(x,y)$ is depicted in Fig. 3. It is worth pointing out that $S_{MB}(x,y) \in [0, S_{MB}(x_{0,MB}, y_{0,MB})]$.
The inner dip of the spot-model $S_{ID}(x,y)$ is defined as a symmetrical 3D curve - in respect to an horizontal surface $H(x,y)$ - to an 'initial 3D curve' which derives from eq. (1), and whose maximum value appears in its center $(x_{0,ID}, y_{0,ID})$.

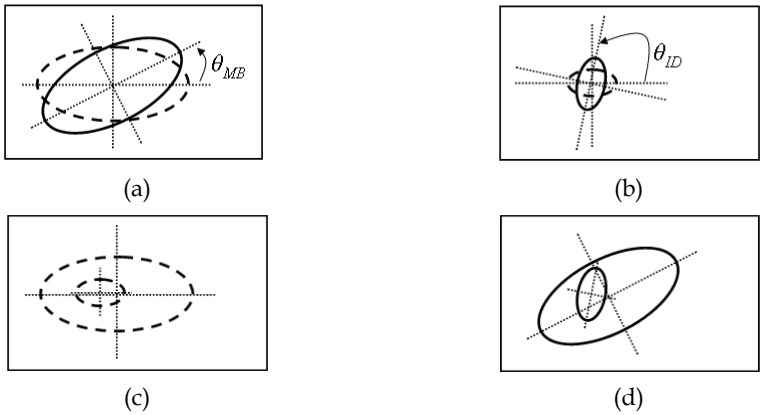


Fig. 3. Illustration of the rotations: (a) The dashed curve represents the contour of the main body of the spot-model before the rotation, while the solid curve represents the contour of the main body of the spot-model after the rotation, (b) The dashed curve represents the contour of the inner dip of the spot-model before the rotation, while the solid curve represents the contour of the inner dip of the spot-model after the rotation, (c) The contour of the total spot-model without applying the rotations, and (d) The contour of the total spot-model after applying the rotations.

4.1.2 Total spot-model

The total spot-model $S_{Model}(x,y)$ as a function of x, y is defined by the following mathematical equation:

$$S_{MODEL}(x,y) = Min\left[S_{MB}(x_{\theta,\mathrm{MB}}, y_{\theta,\mathrm{MB}}), S_{ID}(x_{\theta,ID}, y_{\theta,ID}) \right] \tag{3}$$

where $(x_{\theta,\mathrm{MB}}, y_{\theta,\mathrm{MB}})$ are the rotated coordinates of the (x,y) by an angle θ_{MB} around the 3D curve's center $(x_{0,MB}, y_{0,MB})$ of the main body of the spot-model. Likewise, $(x_{\theta,ID}, y_{\theta,ID})$ are the rotated coordinates of the (x,y) by an angle θ_{ID} around the 3D curve's center $(x_{0,ID}, y_{0,ID})$ of the inner dip of the spot-model.
Rotating the two compartments of the spot-model through the angles θ_{MB} and θ_{ID}, permits both the $S_{MB}(x,y)$ and $S_{ID}(x,y)$ 3D curves to have any possible direction on the 2D plane. An example is shown in Fig. 3. A graphical explanation of eq. (3) is depicted in Fig. 4. The resulting total-models are the areas colored with grey. Depending of the distance between the S_{MB} and S_{ID} centers and the height of the S_{ID} 3D curve, the resulting total-model can

resemble a peak-shaped spot (Fig.4a), a volcano-shaped spot (Fig.4b), or a doughnut-shaped spot (Fig.4c).

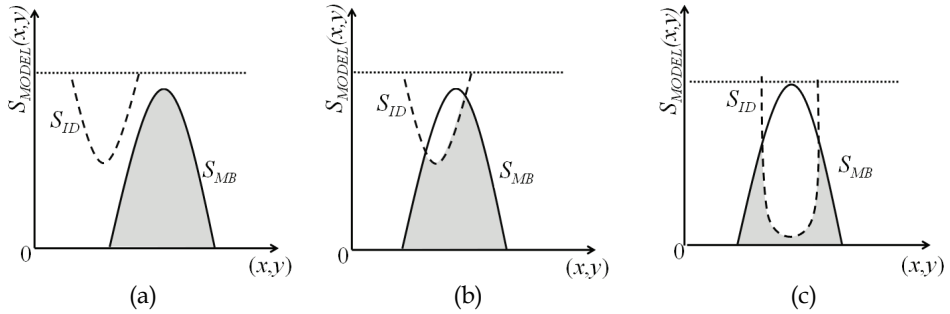


Fig. 4. S_{MB} and S_{ID} components of the morphological models of: (a) a peak-shaped spot, (b) a volcano-shaped spot, and (c) a doughnut-shaped spot. The total morphological models are the grey areas.

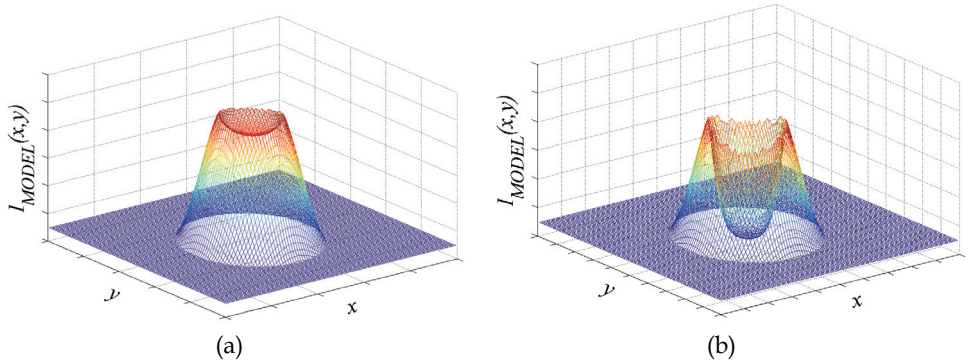


Fig. 5. Examples of the 3D compartment-models containing: (a) a volcano-shaped spot, and (b) a doughnut-shaped spot.

4.1.3 The compartment-model

The compartment-model $I_{MODEL}(x,y)$ as a function of x,y is defined by the following mathematical equation:

$$I_{MODEL}(x,y) = Max\left[B_{AV}, S_{MODEL}(x,y)\right] \tag{4}$$

where B_{AV} denotes the average background intensity of the compartment-model I_{MODEL}. B_{AV} corresponds to a threshold of the lowest values of the $S_{MODEL}(x,y)$. Pixels whose values are lower than B_{AV} belong to background, and their values are set equal to B_{AV}. Thus: $I_{MODEL}(x,y) \in [B_{AV}, S_{MB}(x_{0,MB}\ y_{0,MB})]$.

Two examples of compartment–models $I_{MODEL}(x,y)$ are depicted in Fig. 5. The first compartment-model contains a volcano-shaped spot-model while the second compartment-model contains a doughnut-shaped spot-model.

4.2 Optimum 3D representation and definition of real-spot contour

The first stage of the segmentation procedure is regarded as an optimization problem of modeling a microarray compartment and it is tackled by using the proposed genetic algorithm. A genetic algorithm determines the compartment-model which optimally represents the real-one. More precisely, it determines the values of the variables of the compartment-model (eq. 4) so that the resulting compartment-model represents optimally the real-one.

4.2.1 Chromosome representation

A chromosome m represents a specific compartment-model I^m_{MODEL} in a three-dimensional space, where m stands for a specific chromosome. It is therefore a simple numerical sequence which encodes the values of the variables defining the specific compartment-model. It consists of 3 segments: The first segment encodes the value of the average background intensity of the compartment-model B^m_{AV}. The second segment encodes the values of the variables of the main-body S^m_{MB} of the spot-model S^m_{MODEL}, while the third segment encodes the values of the variables of the inner-dip S^m_{ID} of the spot-model S^m_{MODEL}.

4.2.2 Chromosome evaluation

The aim of the genetic algorithm is the maximization of the resemblance between the compartment-model I^m_{MODEL} and the real-one I_{REAL}. In other words, the higher the resemblance of the compartment-model I^m_{MODEL} (represented by the chromosome m) to the real-compartment I_{REAL} is, the higher the value of the fitness function of a chromosome m becomes. Based on the aforementioned remark, the chromosome evaluation contains the following three main objectives:

1. Maximization of the degree of overlap between the area containing the real microarray spot S_{REAL} and the area containing the spot-model S^m_{MODEL} (represented by chromosome m),
2. Maximization of the resemblance between the real microarray spot S_{REAL} and the main body of the spot-model S^m_{MB} (represented by chromosome m). In this case, let I_{MB} be a model-compartment which contains only the main body of the spot-model (instead of the total-spot model). In correspondence with eq. (4), I_{MB} is defined by the following equation:

$$I_{MB}(x,y) = Max\left[B_{AV}, S_{MB}(x,y)\right] \tag{7}$$

   Subsequently, the aforementioned maximization is equivalent to the maximization of the resemblance between the real-compartment I_{REAL} and the model-compartment I^m_{MB} (represented by chromosome m).
3. Maximization of the resemblance between the real-compartment I_{REAL} and the model-compartment I^m_{MODEL} containing the total spot-model S^m_{MODEL}.

It should be noted, however, that since the real-compartment is contaminated with noise and artifacts, its intensity values are noticeably fluctuated – even between two consecutive pixels – resulting in a scabrous 3D-curve that contains many peaks. As a result, pixels belonging to the spot area S_{REAL} may have lower intensity values than the pixels belonging to the background area B_{REAL}. Correspondingly, pixels belonging to the background area B_{REAL} may have higher intensity values than the pixels belonging to the spot area S_{REAL}.

Contrary to the scabrous 3D-curve of the real-compartment, the compartment-model has a smooth 3D-curve. Consequently, some of the points of the 3D-curve of the compartment-model are identical to the points of the real-one while some others interpolate the points of the real-one. The identical points should belong mostly to the region near the spot's contour while the interpolated points should belong mostly to spot areas or background areas.

To deal with the ambiguity and vagueness of the intensity values of pixels – due to noise, artifacts and uneven background – the genetic algorithm adopts the Fuzzy Logic. We set the '*membership degree*' of a pixel p to belong to the background area or to the spot area according to the following two rules of fuzzy logic theory:

1. The smaller the intensity's value $I_{REAL}(p)$ is, the greater the '*membership degree*' that p_r belongs to the background area becomes, and
2. The higher the intensity's value $I_{REAL}(p)$ is, the greater the '*membership degree*' that p_r belongs to the spot area becomes.

Based on the two aforementioned rules, the membership function $\mu_B(p)$ of a pixel p in order to belong to the background area and the membership function $\mu_S(p)$ of a pixel p in order to belong to the spot area are defined by the following equations:

$$\mu_B(p) = \begin{cases} 1, & if \;\; I_{REAL}(p) \le I_B \\ \dfrac{I_F - I_{REAL}(p)}{I_F - I_B}, & if \;\; I_B < I_{REAL}(p) < I_F \\ 0, & if \;\; I_{REAL}(p) \ge I_F \end{cases}. \tag{5}$$

and,

$$\mu_S(p) = \begin{cases} 0, & if \;\; I_{REAL}(p) \le I_B \\ \dfrac{I_{REAL}(p) - I_B}{I_F - I_B}, & if \;\; I_B < I_{REAL}(p) < I_F \\ 1, & if \;\; I_{REAL}(p) \ge I_F \end{cases} \tag{6}$$

where I_B and I_F are two intensity values. More precisely, let I_o be the intensity corresponding to the minimum between the maxima of the two normal distributions which represent the distributions of background pixels and spot pixels (Fig. 6). I_{min} and I_{max} are the minimum and maximum intensity values that appear in the I_{REAL}. Let N_1 be the number of pixels whose intensities' values are less than I_o and, N_2 be the number of pixels whose intensities' values are higher or equal to I_o. I_B is chosen so that $k \cdot N_1$ number of pixels have intensity lower or equal to I_B, where k is a constant ($0 \le k \le 1$). I_F is chosen so that $k \cdot N_2$ number of pixels have intensity higher or equal to I_F.

Fig. 7 represents the membership functions $\mu_B(p)$ and $\mu_S(p)$. It becomes obvious that pixels with intensity lower or equal to I_B belong to the background area ($\mu_B(p) = 1$ and $\mu_S(p) = 0$), while pixels with intensity higher or equal to I_F belong to the spot area ($\mu_B(p) = 0$ and $\mu_S(p) = 1$). Pixels, with intensity between I_B and I_F, have a 'membership degree' $\mu_B(p)$ to belong to the background area and a 'membership degree' $\mu_S(p)$ to belong to spot area ($\mu_B(p) \ne 0$ and $\mu_S(p) \ne 0$).

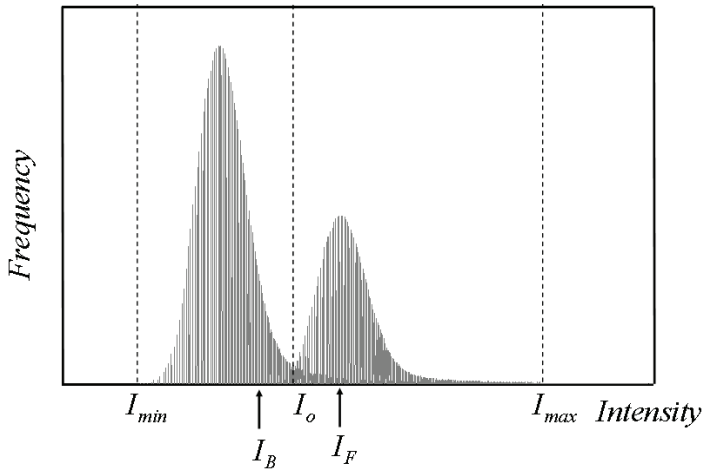


Fig. 6. A typical histogram of a real microarray compartment. The left curve corresponds to background pixels while the right curve corresponds to spot pixels.

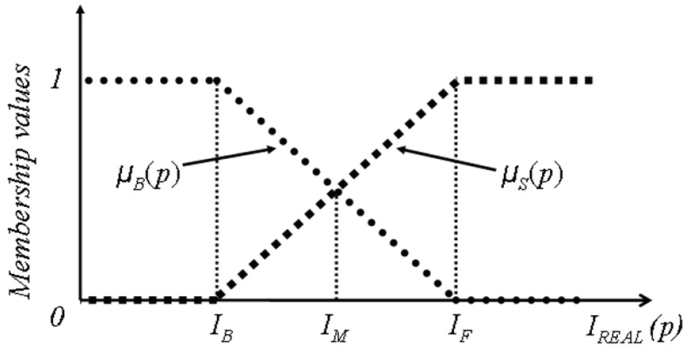


Fig. 7. The membership functions $\mu_B(p)$ and $\mu_S(p)$.

In the next subsections, the three aforementioned objectives for the chromosome evaluation, as well as the way they are combined in order to form the fitness function are apposed in detail.

4.2.2.1 Overlap of the area containing the real microarray apot S_{REAL} and the area containing the spot-model S^m_{MODEL}

Let $\hat{S}_{REAL}$ be the set of pixels whose intensity value is higher or equal to I_M and, $\hat{B}_{REAL}$ be the set of pixels whose intensity value is lower than I_M (Fig. 7). Ideally, $\hat{S}_{REAL}$ contains the pixels belonging to the spot area S_{REAL} while $\hat{B}_{REAL}$ contains the pixels belonging to the background area B_{REAL}. By overlapping the I_{REAL} and the I^m_{MODEL} (Fig. 8) four different regions can be identified: 1) S_A is the set of pixels whose members are the pixels which are located in the area of $\hat{S}_{REAL}$ and in the area of S^m_{MODEL}. 2) S_B is the set of pixels whose members are the pixels which are located in the area of $\hat{S}_{REAL}$ and in the area of B^m_{AV}. 3) S_C is

the set of pixels whose members are the pixels which are located in the area of $\hat{B}_{REAL}$ and in the area of S^m_{MODEL}. 4) S_D is the set of pixels whose members are the pixels which are located in the area of $\hat{B}_{REAL}$ and in the area of B^m_{AV}

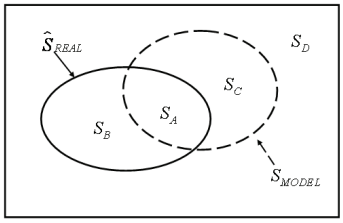


Fig. 8. Overlapping of the I_{REAL} compartment and I^m_{MODEL} compartment. The solid curve represents the area of $\hat{S}_{REAL}$ while the dashed curve represents the area of S^m_{MODEL}.

Using the aforementioned regions, the true positive rate $TP(m)$ and the true negative rate $TN(m)$ can be calculated for the S^m_{MODEL}. Due to the uncertainties existing in the pixel's intensities, the pixels contributing to the calculations are weighted; In the calculation of $TP(m)$, the weight coefficient of a pixel p equals to its corresponding 'membership degree' $\mu_S(p)$, while in the calculation of $TN(m)$ the weight coefficient of a pixel p equals to its corresponding 'membership degree' $\mu_B(p)$.

The higher the $TP(m)$ and the $TN(m)$ are, the higher the overlapping of the S^m_{MODEL} with the $\hat{S}_{REAL}$ is. As a result, the overlap $F_{Overlap}(m)$ of the area containing the real microarray spot S_{REAL} and the area containing the spot-model S^m_{MODEL} is defined by the following equation:

$$F_{Overlap}(m) = TP(m) \cdot TN(m) \tag{8}$$

4.2.2.2 Measure for calculating the error of a model-compartment at a pixel p

Let I^m_C be either the model-compartment I^m_{MODEL} or the model-compartment I^m_{MB}. If the surface of the real-compartment I_{REAL} was smooth, the error of the model-compartment I^m_C at a pixel p should be defined as:

$$E^m_{REAL}(p) = \frac{|I^m_C(p) - I_{REAL}(p)|}{I_{REAL}(p)} \tag{9}$$

However, the surface is not smooth since the real microarray compartment is contaminated with noise. As a result, the error of the model-compartment I^m_C at a pixel p is defined by the following equation:

$$E^m(p) = Min\left[E^m_{REAL}(p), E^m_{MR}(p)\right] \tag{10}$$

where,

$$E^m_{MR}(p) = \frac{|I^m_C(p) - I_{MR}(p)|}{I_{MR}(p)} \tag{11}$$

and,

$$I_{MR}(p)=\begin{cases}\underset{k\in K}{Median}\left[I_{REAL}(k)\right], \text{ if } Max\left[\mu_S(p),\mu_B(p)\right]\geq\lambda_m\\ I_{REAL}(p), otherwise\end{cases} \tag{12}$$

$$K=\left\{k\mid(|k-p|\leq 1)\ \wedge|\mu_S(k)-\mu_S(p)|\leq\lambda_d\right\}. \tag{13}$$

$I_{MR}(p)$ equals either to $I_{REAL}(p)$ or to the median intensity value of a set of pixels $\{K\}$ located in a neighborhood near the pixel p, according to the values of $\mu_S(p)$ or $\mu_B(p)$. λ_m and, λ_d denote two constants ($0\leq\lambda_m,\lambda_d\leq 1$) which control the $I_{MR}(p)$ value of a pixel p.

4.2.2.3 Resemblance between the real-compartment I_{REAL} and the model-compartment I_{MB}^m

The resemblance $R_{MB}(m)$ between the real-compartment I_{REAL} and the model-compartment I_{MB}^m is defined by the following equation:

$$R_{MB}(m)=f_1(m)\cdot f_2(m), \tag{14}$$

where

$$f_1(m)=\frac{\#\left\{p\mid p\in S_1\right\}}{\#\left\{p\mid p\in\hat{S}_{REAL}\right\}}, \tag{15}$$

$$f_2(m)=\frac{\#\left\{p\mid p\in S_2\right\}}{\#\left\{p\mid p\in\hat{B}_{REAL}\right\}} \tag{16}$$

and,

$$S_1=\left\{p\mid E^m(p)\leq E_{MAX}\wedge I_{MB}^m(p)>B_{AV}\right\}, \tag{17}$$

$$S_2=\left\{p\mid E^m(p)\leq E_{MAX}\wedge I_{MB}^m(p)\leq B_{AV}\right\}. \tag{18}$$

The symbol # denotes the number of the elements of the set that is defined by the brackets {}. E_{MAX} is a positive constant which expresses the maximum acceptable error of the model-compartment at a pixel.

S_1 denotes a set of pixels whose members are the pixels p of the compartment-model I_{MB}^m which are located in the area of the main body of the spot-model (S_{MB}) and efficiently represent the corresponding ones of the real compartment I_{REAL} ($E^m(p)\leq E_{MAX}$). Likewise, S_2 denotes a set of pixels whose members are the pixels p of the compartment-model I_{MB}^m which are located in the area of the background (B_{AV}) and efficiently represent the corresponding ones of the real compartment I_{REAL}.

$f_1(m)$ denotes the percentage of the $\hat{S}_{REAL}$ pixels which have been efficiently represented by the compartment-model I_{MB}^m. Likewise, $f_2(m)$ denotes the percentage of the $\hat{B}_{REAL}$ pixels which have been efficiently represented by the compartment-model I_{MB}^m. From eq. (14), the further pixels have been efficiently represented by the I_{MB}^m compartment-model, the higher the value of $R_{MB}(m)$ becomes.

4.2.2.4 Resemblance between the real-compartment I_{REAL} and the model-compartment I^m_{MODEL}

The resemblance between the real-compartment I_{REAL} and the model-compartment I^m_{MODEL} is defined by the following equation:

$$R_{MODEL}(m) = f_3(m) \cdot f_4(m) \tag{19}$$

where

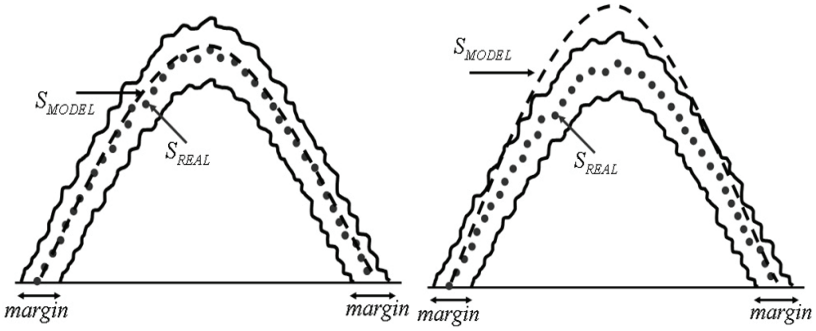
$$f_3(m) = \sum_{p \in I_{REAL}} \mu_S(p) \cdot (1 - w \cdot E^m(p)), \tag{20}$$

$$f_4(m) = \sum_{p \in I_{REAL}} \mu_B(p) \cdot (1 - w \cdot E^m(p)) \tag{21}$$

and,

$$w = \begin{cases} 0.1, & if\ E^m_{REAL}(p) \le E_{MAX} \\ 1, & otherwise \end{cases}, \tag{22}$$

If the value of $E^m_{REAL}(p)$ of a pixel p is less or equal to E_{MAX} (eq. 22), the error between the model-compartment I^m_{MODEL} and the real-compartment I_{REAL} at that pixel is considered negligibly small and thus insignificant. Consequently, the $E^m(p)$ error ((eq. 20) and (eq. 21)) is multiplied by a factor of 0.1 (w=0.1).



(a) (b)

Fig. 9. The dotted curve represents the intensity of a real microarray spot while the dashed curve represents a spot-model. Spot-models whose values fall within the margin, defined by the solid curves, efficiently represent the real microarray spot. (a) An efficient spot-model. (b) An inefficient spot-model.

In fact, the constant E_{MAX} controls an acceptable margin of the error existing between the intensities' values of the compartment-model and the intensities' values of the real compartment. As an example Fig. 9 depicts the margin in the area of the real microarray spot. Spot-models whose values fall within the margin efficiently represent the real microarray spot.

As the value of $f_3(m)$ increases, so does the number of those pixels belonging to $\hat{S}_{REAL}$ which are efficiently represented by the S_{MODEL}. Likewise, as the value of $f_4(m)$ increases, so does the number of those pixels belonging to $\hat{B}_{REAL}$ which are efficiently represented by the B_{AV}. As a result, the greater the number of pixels that are efficiently represented by the I^m_{MODEL} compartment-model is, the higher the value of $R_{MODEL}(m)$ becomes.

4.2.2.5 Fitness function

Each chromosome m in every population is evaluated using a fitness function, $F(m)$, which assigns to it a degree of how appropriate a solution to the optimization problem it is. The higher the value of the fitness function, the more appropriate the chromosome is. The Fitness Function, $F(m)$, of a Chromosome m that encodes a possible solution to the particular optimization problem is defined by the following equation:

$$F(m) = \begin{cases} R_{MB}(m), \ if \ Min\left[f_1(m), f_2(m), R_{MB}(m)\right] \le Th_R \\ 1 + R_{MB}(m), \ else if \ f_3(m) \le 0 \wedge f_4(m) \le 0 \\ R_{MODEL}(m) \cdot F_{Overlap}(m), \ otherwise \end{cases} \tag{23}$$

The Fitness Function $F(m)$ of a Chromosome m equals to $R_{MB}(m)$ (1st case), to $1 + R_{MB}(m)$ (2nd case) or to $R_{MODEL}(m) \cdot F_{Overlap}(m)$ (3rd case), according to the values of $f_1(m)$, $f_2(m)$, $R_{MB}(m)$, $f_3(m)$ and, $f_4(m)$.

If one of the value of $f_1(m)$, $f_2(m)$ and, $R_{MB}(m)$ is less or equal to a threshold Th_R, it means that the model-compartment I^m_{MB} does not resemble at all to the real-compartment I_{REAL} (1st case). Th_R is a threshold which controls the minimum acceptable resemblance of the model-compartment I^m_{MB} with the real-compartment I_{REAL}.

If the values of $f_1(m)$, $f_2(m)$ and, $R_{MB}(m)$ are higher than the threshold Th_R, it means that the model compartment I^m_{MB} resembles to an extent to the real-compartment I_{REAL}. In this case, the fitness function checks the value of $f_3(m)$ and $f_4(m)$. If their values are less than zero, the model compartment I^m_{MODEL} does not resemble at all to the real-compartment I_{REAL}, thus the model-compartment is not an acceptable one (2st case). On the other hand, if their values are higher than zero, it means that the model compartment I^m_{MODEL}, represented by the chromosome m, represents to a degree the real compartment (3rd case). Of course, the higher the value of $R_{MODEL}(m) \cdot F_{Overlap}(m)$ is, the higher the resemblance between the real compartment with the model compartment I^m_{MODEL} becomes.

Using the fitness function $F(m)$, the higher the resemblance of the model-compartment I^m_{MODEL} with the real one is, the higher the value of the fitness function $F(m)$ becomes. This is because the genetic algorithm can assign to the chromosome m of the 3rd case a higher fitness value than to the one of the 2nd case and to the one of the 1st case. For example, the genetic algorithm can progressively assign – from upper left to lower right – a higher fitness value to the chromosomes representing the compartment-models in Fig. 10.

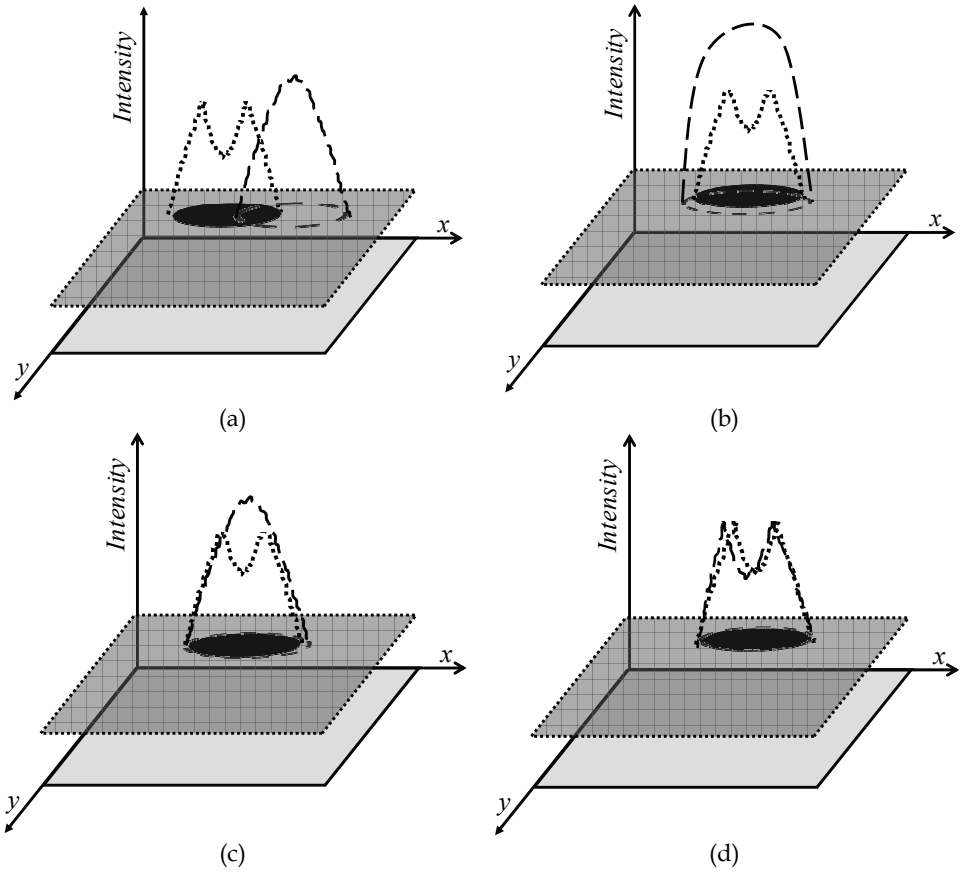


Fig. 10. Overlapping of the real-compartment I_{REAL} and compartment-model I^m_{MODEL}. The dotted curve represents the real spot S_{REAL}, while the dashed curve represents the spot-model S^m_{MODEL}. The chromosome m representing the I^m_{MODEL} in (d) should have progressively higher fitness value than that of (c), (b) and (a).

4.3.3 Evolutionary circle-termination criteria

Let Pop_n be a population of chromosomes, where n stands for the consecutive number of populations. The population consists of N_{pop} chromosomes. A new population Pop_{n+1} of an equal number of chromosomes (N_{pop}) is created through the following stages: (i) Reproduction stage: P_r% of the best chromosomes of the current population Pop_n are carried over to the new population Pop_{n+1}. (ii) Crossover-Mutation stage: The chromosomes needed to complete the new population Pop_{n+1} are produced through iterations of the following: Four chromosomes of the population Pop_n are selected using the tournament selection method (Miller et al., 1995); These chromosomes are subsequently subjected in groups to a crossover operator (according to a P_c% probability) and then to a mutation operator (according to a P_m% probability). The best two of the four resulting chromosomes (those two with the best fitness value) proceed to the new population Pop_{n+1}. It should be noted that the

mutation operator applied is the wavelet-mutation as it exhibits a fine-tune ability as opposed to other mutation operator (Ling et al., 2007). Moreover, the crossover operator applied is the joint application of the BLX-a and the dynamic heuristic crossover as it is the most promising crossover application (Herrera et al., 2005).
New populations are thus produced until at least one of the following two criteria is satisfied: (i) the genetic algorithm is executed up to a maximum number of populations G_{Max}; (ii) the genetic algorithm is executed up to a maximum number of populations G_{Fit} for which the best fitness value has remained unchanged.

5. Results

Several experiments were executed so as to evaluate the performance of the proposed method for spot-segmentation on both synthetic and real cDNA microarray images. Most of the parameters were experimentally adjusted once, and thus they remained unchanged during all the experiments. The constant k of 0.6 was adopted as the most appropriate in order to distinguish: i) the pixels which belong to the background area, ii) the pixels which belong to the spot area, and iii) the pixels which have a 'membership degree' μ_B in order to belong to the background area and a 'membership degree' μ_S in order to belong to the spot area. The constant parameters λ_m of 0.7 and λ_d of 0.2 were adopted as the most appropriate so as to control the $I_{MR}(p)$ value of each pixel p of the real microarray compartment ((eq. 12) and (eq. 13)). A constant E_{MAX} of 0.2 was adopted so as to control the maximum acceptable error of a model-compartment I_C^m at a specific pixel. A threshold Th_R of 0.15 was adopted as the most appropriate one in order to define the minimum acceptable resemblance between a model compartment I_{MB}^m and the real one.
The population size of the genetic algorithm N_{pop} was set to 100. This size is high enough to reduce the possibility of the genetic algorithm prematurely converging to a local solution that would not be an efficient one. Meanwhile, it does not increase the time required for the population to converge to an efficient solution (Achiche et al., 2004). The percentage of each population which was reproduced was relatively small (P_r=10%) as the reproduction was used only for the best chromosomes of the population to be preserved in the next population. In accordance with Miller et al (Miller et al., 2003) the high crossover probability of 80% was chosen (P_c=80%). The mutation probability was experimentally adjusted to 30% (P_m=30%). The termination criterion was satisfied when the genetic algorithm was executed for 500 populations (G_{Max}=1000) or when the best fitness value remained unchanged for 250 populations (G_{Fit}=250).

5.1 Evaluation of the performance using synthetic microarray images

In order to compare the proposed method with preexisting ones objectively, we used an existing dataset of synthetic microarray images for which the ground truth is known. The dataset contains 50 good quality images and 50 low quality images. Each image has been produced by the microarray simulator of Nykter et al (Nykter et al., 2006). It is digitized at 330 x 750 pixels and it contains 1000 spots. Nykter's simulator has been designed to produce synthetic microarray images with realistic characteristics. Consequently, the good quality images have low variability in spot sizes and shapes, while the noise level is reasonable low. On the contrary, the low quality images contain spots whose shape and size vary significantly. In addition, noise level is significantly higher for the low quality images. It

should be noted that this dataset has already been used for the evaluation of other well-known segmentation techniques (see table I), as it is described in (Lehmussola et al., 2006).
During these experiments, the efficiency of the proposed method was analyzed by means of a statistical analysis. The statistical analysis is the one described in (Lehmussola et al., 2006). More precisely, the segmentation accuracy was measured on a pixel level. Firstly, the probability of error PE and the discrepancy distance D for each synthetic spot were calculated. Then, the median probability of error and the median discrepancy distance for both good and low quality images were calculated.
The probability of error PE measures the mis-segmented pixels. It is defined as:

$$PE = P(F) \cdot P(B \mid F) + P(B) \cdot P(F \mid B) \tag{24}$$

where $P(F)$ and $P(B)$ are the a priori probabilities of foreground and background. $P(F \mid B)$ denotes the probability of error in classifying background as foreground, while $P(B \mid F)$ denotes the probability of error in classifying foreground as background.
The discrepancy distance D gives different weights for mis-segmented pixels based on how spatially far they are located from the nearest correct segmentation result.

$$D = \frac{\sqrt{\sum_{i=1}^{N} d^2(i)}}{A} \tag{25}$$

where N is the number of mis-segmented pixels, $d(i)$ is the Euclidian distance from the i-th mis-segmented pixels to the nearest pixel that actually belongs to the mis-segmented class. A is the number of pixels in the image.

Algorithm	Median Value of Probability of error		Median Value of Discrepancy distance	
	Good	Low	Good	Low
Fixed Circle	0.049	0.049	0.027	0.027
Adaptive Circle	0.019	0.192	0.017	0.074
Seeded region growing	0.099	0.114	0.037	0.048
Mann-Whitney	0.165	0.162	0.066	0.074
Hybrid k-means	0.017	0.020	0.016	0.029
Markov random field	0.154	0.053	0.063	0.039
Matarray	0.004	0.031	0.008	0.068
Model-based segmentation	0.094	0.101	0.052	0.067
Proposed method	**0.000**	**0.012**	**0.000**	**0.018**

Table 1. Performance of commonly used and well-known segmentation algorithms as well as of the proposed method

The evaluation results of the proposed method are shown in table I (last row). It becomes obvious that the proposed method can accurately segment the spots of good quality images while it can segment the spots of low quality images quite efficiently. In the same table we have apposed the results of commonly used and well-known segmentation techniques (first

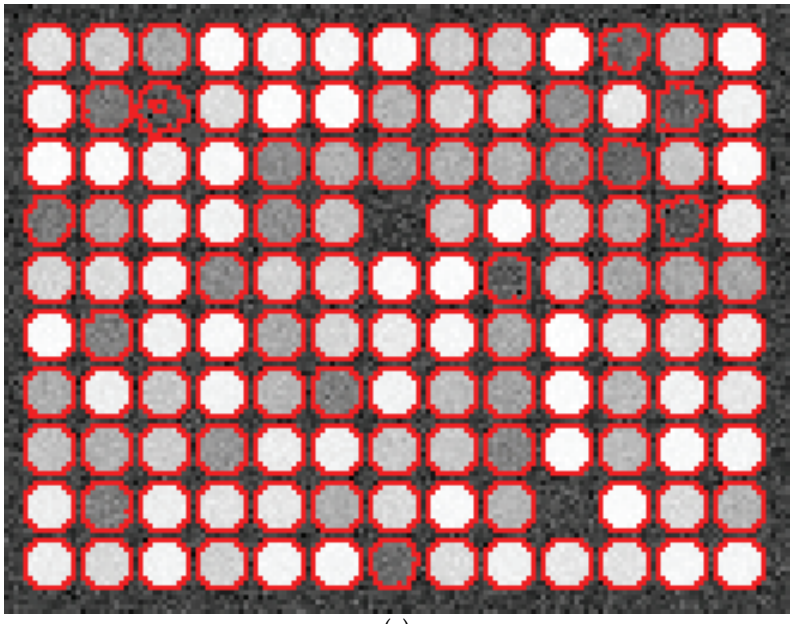
eight rows), as they are reported by Lehmussola et al. Comparing the results of the proposed method with the results of the other software programs, it is obvious that the results of the proposed method are significantly more successful than the ones of the other software programs, indicating the high performance of the proposed method. The significant number of spots which are contained in the used dataset supports these arguments additionally. Indeed, the evaluation of all the methods has been statistically calculated in 50000 artificial microarray spots for which the ground truth is given, meaning that the correct segmentation result is known.
Fig. 11 presents a segmentation result on two blocks of a good quality and a low quality synthetic microarray image. As it is obvious the proposed algorithm has very efficiently segmented the microarray spots.

5.2 Evaluation of the performance using real microarray images

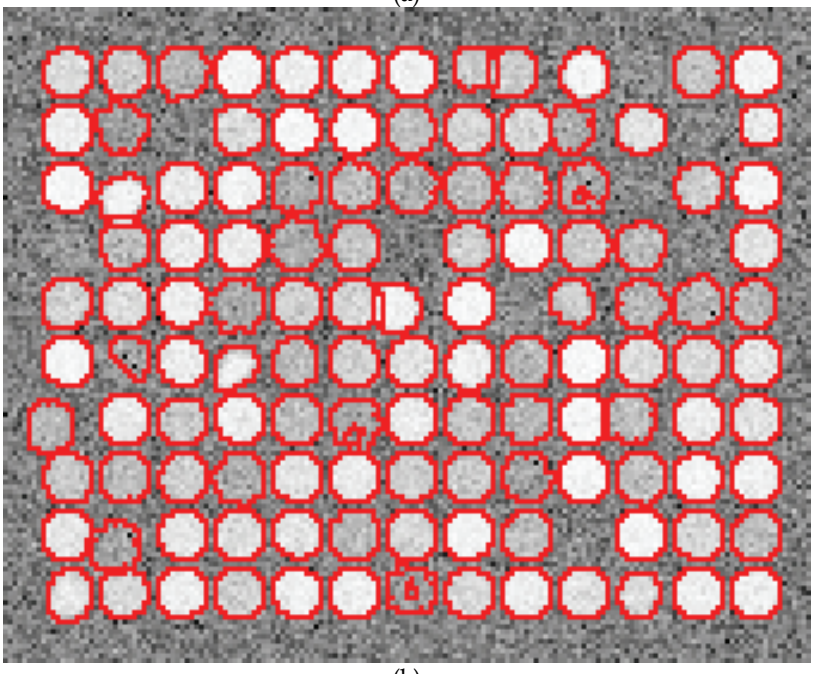
The second dataset contains ten microarray blocks, which have been arbitrarily selected from ten microarray images obtained from the Stanford Microarray Database (SMD) (Standford Microarray Database), which is publicly available. The blocks are digitized at ~ 450 x 450 pixels at 16-bit grey level depth and they are stored in tiff format. Each one of them contains 864 spots. Thus, the blocks contain 8640 spots in total. The microarray images have been produced by comprehensively analyzing the gene expression profiles in 54 specimens of acute lymphoblastic leukemia, 37 positive and 17 negative to BCR-ABL. BCR-ABL is a fusion gene product resulting from translocation between the 9^{th} and the 22^{nd} chromosomes.
Fig. 12 shows the segmentation results of a real-microarray sub-image which is contaminated with noise and contains the three types of microarray spots (peaked-shaped spots, volcano and doughnut-shaped spots). As it is obvious, the proposed method has very efficiently segmented the spots. Moreover, the proposed method has correctly detected the absence of the first spot.

6. Conclusions

Spot-segmentation in microarray images comprises one of the most challenging stages of the microarray image analysis sequence. In this chapter, the segmentation procedure is a result of an optimization problem which is tackled by using a genetic algorithm, which represents, in a three dimensional space, the real-spots of the microarray image with spot-models. In vue of this, fuzzy logic is adopted in order to take into account the uncertainties existing in the pixels' intensities, and which have been caused by noise, artifacts, and uneven background. The segmentation of the real-spots is conducted by drawing the contours of the spot-models.
The proposed approach is noise-resistant and it is efficient under the following adverse conditions: i) the appearance of various spot-shapes, such as peak-shaped spots, volcano-shaped spots and doughnut-shaped spots, ii) the appearance of spots of diverse intensities, such as low intensity spots which are not clearly visible or saturated spots and iii) the appearance of various spot-sizes. Last but not least, it is fully-automatic since it does not require any input parameter or human intervention in order to determine the contours of microarray spots properly. The experimental results over synthetic and real images demonstrate that it is very efficient and effective. Furthermore, it outperforms various existing well-known and broadly used segmentation techniques.



(a)



(b)

Fig. 11. Spot-segmentation result of 2 blocks: a) in a good quality artificial microarray image and, b) in a low quality artificial microarray image.

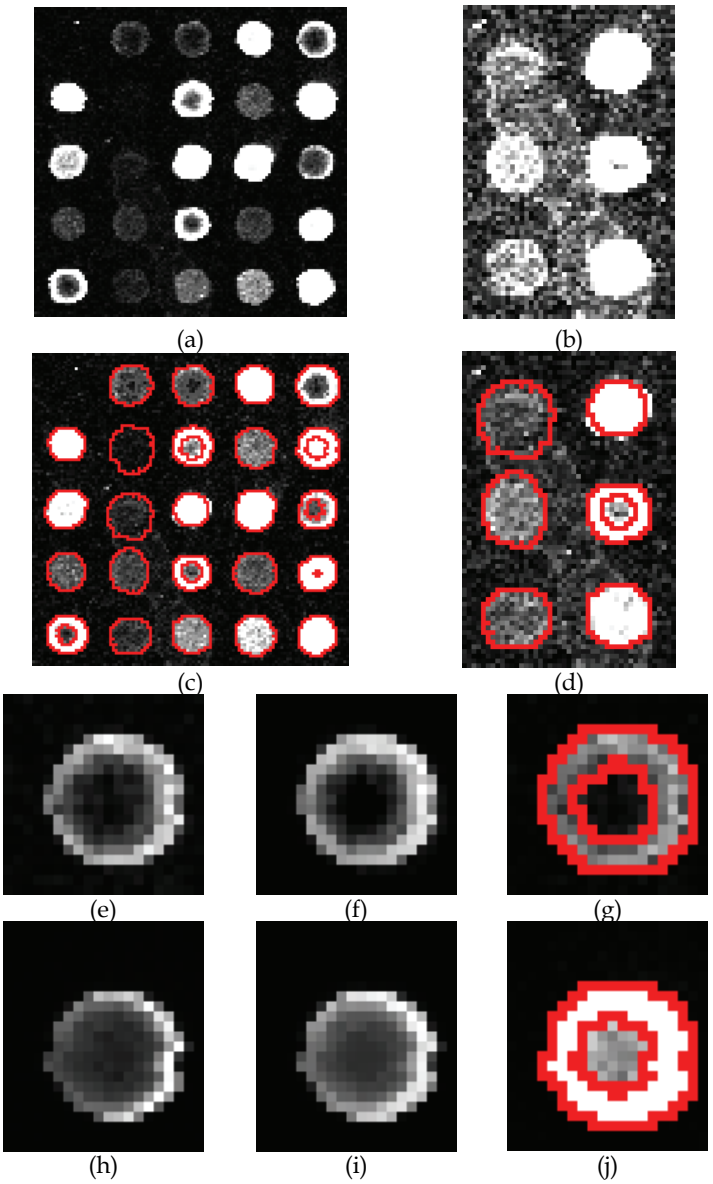


Fig. 12. Spot-segmentation results: (a) A region of a real microarray image containing the tree types of microarray spots in the presence of noise, (b) enlargement of a part of the microarray image which is contaminated with noise, (c,d) the spot-segmentation results of (a,b), (e) enlargement of a doughnut-shaped spot (f) the spot-model representing the doughnut-shaped spot, (g) the segmentation result of the doughnut-shaped spot, (h) enlargement of a volcano-shaped spot (i) the spot-model representing the volcano-shaped spot, and (j) the segmentation result of the volcano-shaped spot.

7. References

Achiche, S. Baron, L. & Balazinski, M. (2004). "Real/binary-like coded versus binary coded genetic algorithms to automatically generate fuzzy knowledge bases: a comparative study," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 4, pp. 313-325, 2004.

Bozinov, D. & Rahnenfuhrer, J. (2002). "Unsupervised technique for robust target separation and analysis of DNA microarray spots through adaptive pixel clustering," *Bioinformatics*, vol. 18, no. 5, pp. 747–756, 2002.

Buckley M.J. (2000). The Spot User's Guide, CSIRO mathematical and information sciences. http://www.cmis.csiro.au/IAP/Spot/ spotmanual.htm

Buhler, J. ; Ideker, T. ; & Haynor, D. "Dapple: improved techniques for finding spots on DNA microarrays," *UW CSE Technical Report UWTR 2000-08-05*, pp. 1-12, Aug. 2000.

Campbell, A.M. & Heyer L.J. (2007). " Discovering Genomics, Proteomics & Bioinformatics," 2nd ed. , Pearson Benjamin Cummings, 2007, pp. 233-238.

Chen, Y.; Dougherty, E.R. &. Bittner, M.L. (1997). "Ratio-based decisions and the quantitative analysis of cDNA microarray images," *Journal Biomedical Optics*, vol. 2 no. 4, pp. 364-374, 1997.

Chen, W.B.; Zhang, C. & Liu, W.L. (2006). "An Automated Gridding and Segmentation Method for cDNA Microarray Image Analysis," in *Proc. 19th IEEE Symp. Computer-Based Medical Systems*, Salt Lake City, 2006, pp. 893-898.

Eisen. M.B. (1999). ScanAlyze. [Online]. Available: http://rana.lbl.gov/EisenSoftware.htm

Demirkaya, O. ; Asyali, M.H. & Shoukri, M.M. (2005). "Segmentation of cDNA microarray spots using markov random field modeling," *Bioinformatics*, vol. 21, no. 13, pp. 2994–3000, Apr. 2005.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization & Machine Learning, Boston: Addison-Wesley, Reading, 1989, ch. 1.

Hayes, C.S.M. (2006). "Generic Properties of the Infinite Population Genetic Algorithm," Ph.D. dissertation, Dept. Mathematics, Montana State Univ., Bozeman, Montana, USA, 2006.

Herrera, F.; Lozano, M.; & Verdegay, J.L. (1998). "Tackling Real Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265–319, Nov. 1998.

Herrera, F.; Lozano, M. & Sanchez, A.M. (2005). "Hybrid crossover operators for real-coded genetic algorithms: An experimental study," *Soft Computing*, vol. 9, no. 4, pp. 280-298, Apr. 2005.

Ho, J. & Hwang, W.L. (2008). "Automatic Microarray Spot Segmentation using a snake-fisher model," IEEE Trans. on Medical Imaging, vol. 27, no. 6, pp. 847-857, Jun. 2008.

Kim H.Y. et al. (2007). "Characterization and simulation of cDNA microarray spots using a novel mathematical model," *BMC Bioinformatics*, vol. 8, pp. 485-496, March 2007.

Lehmussola, A.; Ruusuvuori, P.; & Yli-Harja, O. (2006). "Evaluating the performance of microarray segmentation algorithms," Bioinformatics, vol. 22, no. 23, pp. 2910-2917, Oct. 2006.

Leung, Y.F. & Cavalieri, D. (2003). "Fundamentals of cDNA microarray data analysis," *Trends in Genetics*, vol. 19, no. 11, pp. 649-659, Nov. 2003.

Ling, S.H. & Leung, F.H.F. (2007). "An improved genetic algorithm with average-bound crossover and wavelet mutation operations," *Soft Computing*, vol. 11, no. 1, pp. 7-31, 2007.

Li, Q. ; Fraley, C. ; Bumgarner, R.E. ; Yeung, K.Y. & Raftery, A.E. (2005). "Donuts, scratches and blanks: robust model-based segmentation of microarray images," *Bioinformatics*, vol. 21, no. 12, pp. 2875-2882, Apr. 2005.

Miller, B.L. & Goldberg, D.E. (1995). "Genetic Algorithms, Tournament selection, and the Effects of Noise," *Complex Systems*, vol. 9, no. 3, pp. 193-212, 1995.

Miller, M.T. Jerebko, A.K. Malley, J.D. & Summers, R.M. (2003). "Feature selection for computer-aided polyp detection using genetic algorithms," in *Proc. of SPIE*, Santa Clara, 2003, pp. 102-110.

Nagarajan, R. (2003). "Intensity-based segmentation of microarray images," *IEEE Trans. on Medical Imaging*, vol. 22, no. 7, pp. 882–889, Jul. 2003.

Rahnenfuhrer, J. & Bozinov, D. (2003). "Hybrid clustering for microarray image analysis combining intensity and shape features," *BMC Bioinformatics*, vol. 5, no. 5, p. 47-58, Apr. 2004.

Nykter, M. et al. (2006). "Simulation of microarray data with realistic characteristics," BMC *Bioinformatics*, vol. 7, pp. 349-366, Jul. 2006.

Srinark, T. & Kambhamettu, C. (2004). "A Microarray Image Analysis System Based on Multiple Snakes," *Journal of Biological Systems*, vol. 12, no. 25, Jun. 2004.

Srinark, T. & Kambhamettu, C. (2001). "A framework for Multiple Snakes," in *Proc. Comp. Vision and Pattern Recogn.*, vol. 2, pp. 202-209, 2001.

Standford Microarray Database. [Online]. Available: http://genome-www5.stanford.edu/

Tu, Y.; Stolovltzky, G.; & Kleln, U. (2002). "Quantitative noise analysis for gene expression microarray experiments," in *Proc. of National Academy of Sciences of the United States of America (PNAS)*, vol. 99, no.22, pp. 14031–14036, Oct. 2002.

Wang, X.H. & Istepanian, R.S.H. (2003). "Microarray image enhancement by denoising using stationary wavelet transform," *IEEE Trans. on Nanobioscience*, vol. 2, no. 4, pp 184-189, Dec. 2003.

Wang, X.; Ghosh, S.; & Guo, S-W. (2001). "Quantitative quality control in microarray image processing and data acquisition," *Nucleid Acids Research*, vol. 29, no. 15, 2001.

Yang, Y. H.; Buckley, M. J.; Dudoit, S. & Speed, T. (2002) "Comparison of methods for image analysis on cDNA microarray data," *J. of Comp. & Graphical Statistics*, vol. 11, no. 1, pp. 108–136, 2002.

25

Discretization of a Random Field – a Multiobjective Algorithm Approach

Guang-Yih Sheu
Chang-Jung Christian University
Taiwan

1. Introduction

Discretization of a random field by the generalized polynomial chaos (GPC) begins with selecting a specific type of orthogonal polynomial (e.g. Legendre and Hermite polynomials) (e.g. Ghanem and Spanos, 1991). This selection of a type of orthogonal polynomial can be performed based on the reported experiences (e.g. Xiu and Karniadakis, 2003) or data revealing the distribution of a random field to be discretized. If such data or reported experiences are unavailable, a third way may be generating some pilot tests to study the performance of a specific type of orthogonal polynomial in discretizing this random field. This study tries to develop an evolutionary algorithm-based auxiliary tool for the implementation of such pilot tests. A similar tool (Allaix and Carbone, 2009), which is based on the single-objective evolutionary algorithm, had been developed for constructing the Karhunen-Loève (KL) representation of a random field. Both KL and GPC expansions are two of the popular random field discretization methods (Ghanem and Spanos, 1991). But, the KL expansion should be applied under a prerequisite of knowing the covariance matrix of a random field to be discretized (e.g. Ghanem and Spanos, 1991); while, the GPC expansions can be applied without similar prerequisites (e.g. Xiu and Karniadakis, 2003). Therefore, the development of an auxiliary tool for constructing a GPC representation of a random field would be necessary.

The succeeding research considers the derivation of an GPC representation of a random field as a multi-objective (MO) problem having two goals: (a) limiting the computational efforts spent in applying the resulting GPC representation; and (b) keeping the resulting GPC representation satisfying all accuracy standards (e.g. getting the sufficiently accurate prediction of statistical parameters of a random field). The former goal will be attained by limiting the highest order of polynomial term and total number of uncorrelated random variables used to construct a GPC representation; while the latter goal will be attained by minimizing multiple error estimators. Since there are multiple goals to be attained, a multiple objective evolutionary algorithm (MOEA) is required. Among all available MOEAs, the strength Pareto evolutionary algorithm II (SPEA 2) (Zitzler, et al., 2001) is chosen. The highest order of polynomial term and total number of uncorrelated random variables used to construct an GPC representation are considered as two parameters to be identified.

The remainder of this study is organized into four sections. In Sec. 2, the theoretical backgrounds of GPC expansions (e.g. Xiu and Karniadakis, 2003) are briefly reviewed. In Sec. 3, the SPEA 2 (Zitzler, et al., 2001) is used to construct a parameter identification procedure to identify the aforementioned highest order of orthogonal polynomial and total

number of uncorrelated random variables. Two examples are introduced to study the performance of the resulting works are tested in Sec. 4. The test results are used to give some discussion and conclusion in Sec. 5.

2. Generalized polynomial chaos expansion

2.1 Definition

The GPC (e.g. Xiu and Karniadakis, 2002) is a generalization of the classical Wiener's PC (polynomial chaos). This Wiener's PC is defined as the span of Hermite polynomials of a Gaussian process. A Cameron-Martin theorem states that the Wiener's PC can be used to approximate any functional in $L_2(C)$ and converges in the $L_2(C)$ sense in which C denotes the space of real functions, which are continuous over the interval [0, 1] and vanish at 0. The GPC further provides a mean of expanding second-order random fields having finite variance over a specific interval in terms of orthogonal polynomials. Most physical processes can be simulated by such second-order random fields. As compared to the Wiener's PC, the GPC has better performance in representing some specific types of non-Gaussian inputs.

Suppose θ is an event in the probabilistic space, and u(θ) is a continuous function of θ. The GPC representation of u is equated by (e.g. Xiu and Karniadakis, 2002)

$$\begin{aligned} u = a_0\psi_0(\xi_{i_0}) + \sum_{i_1=1}^{\infty} a_{i_1}\psi_1(\xi_{i_1}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1} a_{i_1 i_2}\psi_2(\xi_{i_1},\xi_{i_2}) \\ + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3}\psi_3(\xi_{i_1},\xi_{i_2},\xi_{i_3}) + ... \end{aligned} \tag{1}$$

where a_0, a_{i_1}, $a_{i_1 i_2}$,.... denote the coefficients to be determined and ψ_n, n = 0, 1, 2… are the polynomial chaos (PC) of order n of multi-dimensional independent random variables $\xi_{i_1}, \xi_{i_2} ... and\ \xi_{i_n}$ having zero mean and unit variance.

For the notational convenience, Eq. (1) is further modified to (e.g. Xiu and Karniadakis, 2002)

$$u = \sum_{i=0}^{\infty} U_i \Psi_i(\boldsymbol{\xi}) \tag{2}$$

where $\boldsymbol{\xi} = (\xi_{i_1}, \xi_{i_2} ... \xi_{i_n})$, $\Psi_0 = 1$, and U_0 is set to the mean value of u. There is a one-to-one correspondence between Ψ_i and ψ_n and between a_0, a_{i_1}, $a_{i_1 i_2}$,.... and U_i. For example, suppose u is as functions of **x** and $\boldsymbol{\xi} = (\xi_{i_1}, \xi_{i_2})$. The relationship between Ψ_i, i = 0-9 and ψ_n, n = 0-3 is given by

$$\begin{aligned} &\Psi_0 = \psi_0(\xi_{i_1})\psi_0(\xi_{i_2}) = 1, \quad \Psi_1 = \psi_1(\xi_{i_1})\psi_0(\xi_{i_2}), \quad \Psi_2 = \psi_0(\xi_{i_1})\psi_1(\xi_{i_2}) \\ &\Psi_3 = \psi_2(\xi_{i_1})\psi_0(\xi_{i_2}), \quad \Psi_4 = \psi_1(\xi_{i_1})\psi_1(\xi_{i_2}), \quad \Psi_5 = \psi_0(\xi_{i_1})\psi_2(\xi_{i_2}) \\ &\Psi_6 = \psi_3(\xi_{i_1})\psi_0(\xi_{i_2}), \quad \Psi_7 = \psi_2(\xi_{i_1})\psi_1(\xi_{i_2}), \quad \Psi_8 = \psi_1(\xi_{i_1})\psi_2(\xi_{i_2}) \\ &\Psi_9 = \psi_0(\xi_{i_1})\psi_3(\xi_{i_2}) \end{aligned} \tag{3}$$

In addition, because Ψ_i, i = 0-∞ and ψ_n, n = 0-∞ are orthogonal polynomials in terms of $\boldsymbol{\xi}$, it can be obtained:

$$\left\langle \Psi_i, \Psi_j \right\rangle = \left\langle \Psi_i^2 \right\rangle \delta_{ij} \text{ and } \left\langle \psi_i, \psi_j \right\rangle = \left\langle \psi_i^2 \right\rangle \delta_{ij}, \quad \forall \mathrm{i,j} = 0\text{-}\infty \tag{4}$$

where δ_{ij} is the Kronecker delta (i.e. $\delta_{ij} = 1$ if i = j and $\delta_{ij} = 0$ if $i \neq j$), $\langle \cdot,\cdot \rangle$ is the ensemble average. If f and g are two orthogonal polynomials of $\boldsymbol{\xi}$, $\langle \cdot,\cdot \rangle$ is computed by (Xiu and Karniadakis, 2002; Xiu and Karniadakis, 2003)

a. Continuous case (i.e. $\xi_{i_1}, \xi_{i_2} ... and\ \xi_{i_n}$ vary continuously over the probabilistic space):

$$\left\langle f(\boldsymbol{\xi}), g(\boldsymbol{\xi}) \right\rangle = \int\int \cdots \int f(\boldsymbol{\xi}) g(\boldsymbol{\xi}) w(\xi_{i_1}) w(\xi_{i_2}) \cdots w(\xi_{i_n}) d\xi_{i_1} d\xi_{i_2} \cdots d\xi_{i_n} \tag{5a}$$

b. Discrete case (i.e. $\xi_{i_1}, \xi_{i_2} ... and\ \xi_{i_n}$ vary discretely over the probabilistic space):

$$\left\langle f(\boldsymbol{\xi}), g(\boldsymbol{\xi}) \right\rangle = \sum_{\xi_{i_1}} \sum_{\xi_{i_2}} \cdots \sum_{\xi_{i_n}} f(\boldsymbol{\xi}) g(\boldsymbol{\xi}) w(\xi_{i_2}) \cdots w(\xi_{i_n}) \tag{5b}$$

where $w(\xi_{i_1})$, $w(\xi_{i_2})....w(\xi_{i_n})$ represent the weighting functions.

Equation (4) can be used to get U_i, i = 0-∞. Multiplying Eq. (2) with Ψ_i, i = 0-∞ and simplifying the resulting equations according to Eq. (5a) or (5b) give

$$U_i = \frac{\langle u \Psi_i \rangle}{\left\langle \Psi_i^2 \right\rangle}, \quad \forall i = 0\text{-}\infty \tag{6}$$

In practice, not all U_i are computed. Eq. (2) can be truncated as follows

$$u = \sum_{i=0}^{M} U_i \Psi_i(\boldsymbol{\xi}) \tag{7}$$

where M is $\frac{(n+P)!}{n!P!} - 1$, n represents the total number of uncorrelated random variables, and P is the highest order of polynomial term used to equate Ψ_i.

Tables 1-2 (e.g. Xiu and Karniadakis, 2002) list available choices of orthogonal polynomials and corresponding statistical distributions and weighting functions to generate Ψ_i, i = 0-∞, $\xi_{i_1}, \xi_{i_2} ... \xi_{i_n}$, and $w(\xi_{i_1})$, $w(\xi_{i_2})....w(\xi_{i_n})$; respectively.

Distribution	Polynomial	w(ξ)	Interval
Gaussian	Hermite polynomial $H_n(x)$	$\exp(-\xi^2)$	(-∞,∞)
gamma	Laguerre polynomial $L_n(x)$	$\exp(-\xi)$	[0, ∞]
beta	Jacobi polynomial $G_n(p, q, x)$	$(1-\xi)^{p-q}\xi^{q-1}$	[a, b]
uniform	Legendre polynomial $P_n(x)$	1	[a, b]

Table 1. Polynomials, weighting functions, and statistical distributions for generating an GPC expansion (Continuous case) (e.g. Xiu and Karniadakis, 2002)

Distribution	Polynomial	w(ξ)	Interval
Poisson	Charlier polynomial $C(x, \lambda)$	$\exp(-\lambda)\lambda^{\xi}/\xi!$	{0, 1, 2…}
binomial	Krawtchouk polynomial $K_n(x, p, N)$	$N!p^{\xi}(1-p)^{N-\xi}/[\xi!(N-\xi)!]$	{0, 1…N}
negative binomial	Meixner polynomial $M_n(x, \beta, c)$	$(\beta)_{\xi}(1-c)^{\beta}c^{\xi}/\xi!$	{0, 1, 2…}
hypergeometric	Hahn polynomial $Q_n(x, \alpha, \beta, N)$	$(\alpha+\xi)!(\beta+N-\xi)!/[\xi!\alpha!(N-\xi)!\beta!]$	{0, 1…N}

Table 2. Polynomials, weighting functions, and statistical distributions for generating an GPC expansion (Discrete case) (e.g. Xiu and Karniadakis, 2002)

Furthermore, some references (e.g. Xiu and Karniadakis, 2002; Xiu and Karniadakis, 2003) had summarized useful properties about those orthgonal polynomials listed in Tables 1-2. Interested readers may refer to these papers and these properties are not repeatedly listed here.

2.2 Discretization error estimator

Discretizing u by Eq. (7) causes some discretization errors. These discretization errors can be quantified by some error estimators. For example, the exact value of standard deviation of u (or other statistical parameters) and the one provided by Eq. (7) can be used to equate an error estimator. However, the exact value of standard deviation of u may be unavailable or difficult to be obtained. At such a situation, a Monte Carlo simulation (MCS) is required. This MCS is performed by first generating some samples of u. The standard deviation of resulting samples of u is then computed. It had been concluded (e.g. Ghanem and Spanos, 1991) that the standard deviation of u provided by an MCS will approach its exact value, if a sufficiently large amount of samples have been generated to implement the MCS. Thus, statistical parameters computed by an MCS, which is completed using a large amount of samples of u, can be used to understand the accuracy of Eq. (7).

For simplicity, this study defines two types of error estimators to quantify the discretization errors. The values of these error estimators are kept within an acceptable range when constructing a GPC representation of a random field. The first type of error estimator is equated to quantify the error ε_1 caused by truncating Eq. (2) to derive Eq. (7). At a specific $\boldsymbol{\xi}$, ε_1 is defined by (Field and Grigoriu, 2004)

$$\varepsilon_1 = \frac{1}{u_{ex}} \sum_{i=M+1}^{\infty} U_i \Psi_i = \frac{1}{u_{ex}} \left(u_{ex} - \sum_{i=0}^{M} U_i \Psi_i \right) = 1 - \frac{1}{u_{ex}} \sum_{i=0}^{M} U_i \Psi_i \tag{8}$$

where the subscript ex denotes the exact value.

The second type of error estimator is equated to quantify the errors between the exact value of standard deviation of u and the one computed based on Eq. (7). Suppose σ_{ex} and σ_{GPC} denote the exact value of standard deviation of u and the one provided by Eq. (7); respectively. The latter σ_{GPC} is computed by (Ghanem and Spanos, 1991)

$$\sigma_{GPC} = \sqrt{\sum_{i=1}^{M} U_i^2 \left\langle \Psi_i^2 \right\rangle} \tag{9}$$

Based on Eq. (9), the error ε_2 between σ_{ex} and σ_{GPC} is defined by

$$\varepsilon_2 = \frac{1}{\sigma_{ex}} \left[\sigma_{ex} - \sqrt{\sum_{i=1}^{M} U_i^2 \left\langle \Psi_i^2 \right\rangle} \right] = 1 - \frac{1}{\sigma_{ex}} \sqrt{\sum_{i=1}^{M} U_i^2 \left\langle \Psi_i^2 \right\rangle} \tag{10}$$

If σ_{ex} is unavailable or diffcult to be obtained, σ_{MCS} is substituted for it where the subscript MCS denotes the Monte Carlo simulation.

Equations (8) and (10) will be used to define the objective functions in the next section.

3. Parameter identification procedure

As stated in Sec. 1, this study considers the derivation of an GPC representation of a random field as an MO problem and applies the SPEA2 (Zitzler et al., 2001) to solve this MO

problem. Mathematically, an MO problem is defined by (Tan, et al., 2005) finding a set of vector, **P** such that

$$\underset{\boldsymbol{\Theta}\in\boldsymbol{\Phi}}{\text{Min}}\,\mathbf{F}(\boldsymbol{\Theta}) \quad \boldsymbol{\Theta}\in\Re^{N} \tag{10}$$

where $\boldsymbol{\Theta} = \{\theta_1, \theta_2 ... \theta_N\}$ is an N-dimensional vector having N parameters, $\boldsymbol{\Phi}$ defines a feasible set of $\boldsymbol{\Theta}$, and $\mathbf{F} = \{f_1, f_2 ... f_m\}$ is an objective vector with m objective functions f_i (i = 1 to m) to be minimized.

For the succeeding research, suppose the mean value and σ of a random field to be discretized have been known or computed by an MCS. It is intended to identify n and P for constructing the GPC representaton of this random field; thus, $\boldsymbol{\Theta}$ is equal to {n, P}. Meanwhile, the goals are (a) limiting the computational efforts spent in applying the resulting GPC representation; (b) keeping the accuracy of resulting GPC representation satisfying all accuracy standards. The former goal can be attained by choosing n and P within an acceptable range; while, this sudy attains the latter goal by minimizing the next three objective functions f_i, i = 1-3:

$$\begin{aligned} f_1 &= \varepsilon_1(\xi_{i_1} = \xi_{i_2} = \cdots = \xi_{i_n} = \xi_a) - S_1 \\ f_2 &= \varepsilon_1(\xi_{i_1} = \xi_{i_2} = \cdots = \xi_{i_n} = \xi_b) - S_2 \\ f_3 &= \varepsilon_2 - S_3 \end{aligned} \tag{11}$$

where ξ_a and ξ_b denote two specific values of $\boldsymbol{\xi}$ and S_i, i = 1-3 are three constants.

Based on those concepts (Tan, et al., 2005), which are frequently mentioned in solving an MO problem, Sec. 3.1 lists the general steps to identify n and P. Secs. 3.2-3.3 explains the details of specific steps.

3.1 General structure

The identification of n and P is performed by next six steps:

a. Initially, generating 2N random numbers for creating a population $\boldsymbol{\Theta}_t$ (the subscript t denotes the generation number) containing N sets of candidate values of n and P. Since n and P should be integers, each random number is rounded to its nearest integer for producing a candidate n or P value. Besides, create an empty archive $\boldsymbol{\Xi}_t$ for storing the resulting Pareto optimal set. Limit the maximum size of $\boldsymbol{\Xi}_t$ to a number N_{max}.

b. Compute the fitness value of each individual of $\boldsymbol{\Theta}_t \cup \boldsymbol{\Xi}_t$. This fitness value is defined as the sum of the number of individuals dominating an individual and the density assessment at this individual in an objective space. The density assessment is used to speed up the convergence of Pareto optimal set until it contains only nondominated individuals. Sec. 3.2 further explains the computation of fitness values of an individual.

c. Generate a temporary Pareto optimal set as follows (Tan, et al., 2005 and Zitzler, et al., 2001): Nondominated individuals of $\boldsymbol{\Theta}_t \cup \boldsymbol{\Xi}_t$ are first copied to $\boldsymbol{\Xi}_{t+1}$. If there are less than N_{max} individuals in the resulting $\boldsymbol{\Xi}_{t+1}$, sort dominated individuals of $\boldsymbol{\Theta}_t \cup \boldsymbol{\Xi}_t$ in an ascending order by their fitness values. Fill $\mathbf{Q}_{t+1}$ with first (N_{max} - $|\boldsymbol{\Xi}_{t+1}|$) dominated individuals ($|\boldsymbol{\Xi}_{t+1}|$ is the size of $\boldsymbol{\Xi}_{t+1}$). If there are more than N_{max} individuals in the resulting $\boldsymbol{\Xi}_{t+1}$, sort $\boldsymbol{\Xi}_{t+1}$ in a descending order by the distance of each individual to its k-th nearest neighbor. Then, remove last ($|\boldsymbol{\Xi}_{t+1}|$ - N_{max}) individuals from the sorted $\boldsymbol{\Xi}_{t+1}$. A reference book (Tan, et al., 2005) gives the pseudo-codes to implement this step.

d. Test whether a stopping criterion is satisfied. For example, it can stop when a maximum generation number is reached. If the stopping criterion is satisfied, the final Pareto optimal set is Ξ_{t+1}. Then, go to Step (f). Otherwise, execute the tournament selection, crossover, and mutation genetic operators to fill Θ_{t+1} with P offspring of individuals of Ξ_{t+1}. The implementation of these genetic operators is explained in Sec. 3.3.
e. Increment the generation number t. Repeat Steps (a) to (d).
f. Select manually the final values of n and P from Ξ_{t+1}.

3.2 Fitness assignment

Continuing Step (b) of Sec. 3.1, suppose the fitness value of each individual of $\Theta_t \cup \Xi_t$ to be computed. The fitness value F_i of an i-th individual is obtained by (Zitzler, et al., 2001)

$$F_i = R_i + D_i \tag{12}$$

where R_i is the raw fitness value defined based on the total number of individuals dominating the i-th individual and D_i denotes the density assessment at the i-th individual in an objective space.

The total number of individuals dominating an i-th individual is represented by a strength value S_i expressed in the form as

$$S_i = \left| \{ j \mid j \in \Theta_t \cup \Xi_t \wedge i \succ j \} \right| \tag{13}$$

where j denotes the j-th individual, $|.|$ denotes the cardinality of a set, and $\succ$ denotes the Pareto dominance. Based on Eq. (13), R_i, is defined by

$$R_i = \sum_{j \in \Theta_t \cup \Xi_t \wedge j \succ i} S_j \tag{14}$$

However, Eq. (14) may fail when most individuals do not dominate each other. Therefore, Eq. (12) includes D_i and it is computed by the distance d_i^k to its k-th nearest neighbor. The method for calculating d_i^k is as follows: First, calculate the distances of an i-th individual to all other j-th individuals of $\Theta_t \cup \Xi_t$. Thus

$$d_{ij} = \sqrt{\sum_{L=1}^{3} (f_{L,i} - f_{L,j})^2} \tag{15}$$

where the subscripts i and j denote the i-th and j-th individual; respectively. The resulting d_{ij} are then stored in a list. Sort this list and d_i^k is the k-th element of sorted list. The resulting d_i^k is used to define D_i as follows:

$$D_i = \frac{1}{d_i^k + 1} \tag{16}$$

in which 1 in the denominator is to ensure Eq. (16) is less than 1.

3.3 Genetic operators

The tournament selection, crossover, and mutation operators are applied to produce the offspring of those random numbers generated for getting candidate n and P values. Those random numbers should be coded into chromosomes before applying those genetic operators (Tan, et al., 2005).

a. n the tournament selection operation, two chromosomes are randomly selected. Discard the chromosome dominated by the other one. If else, two chromosomes are all conserved. Consider a population Ξ as an example. The tournament selection operation will repeat $|\Xi|$ times ($|\Xi|$ denotes the size of Ξ).
b. In the crossover operation, the crossover probability p_c is first defined. Two chromosomes, which have survived from the tournament selection operation, are chosen as the parents for producing the offspring. The offspring are generated by combining parts of the chromosomes contributed by each parent. Continuing using Ξ in Step (a), the crossover operation will repeat until Ξ is recovered to its original size. Figure 1(a) further illustrates this crossover operation in which chromosomes are represented by binary strings.
c. In the mutation operation, the mutation probability p_m is first set. A chromosome, which has survived from the tournament selection operation, is randomly selected. The structure of it is randomly changed to produce a new chromosome. Continuing using Ξ in Steps (a)-(b), the crossover operation will repeat $\frac{|\Xi|}{2}$ times. Figure 1(b) illustrates this mutation operator in which the chromosome to be mutated is also represented by a binary string.

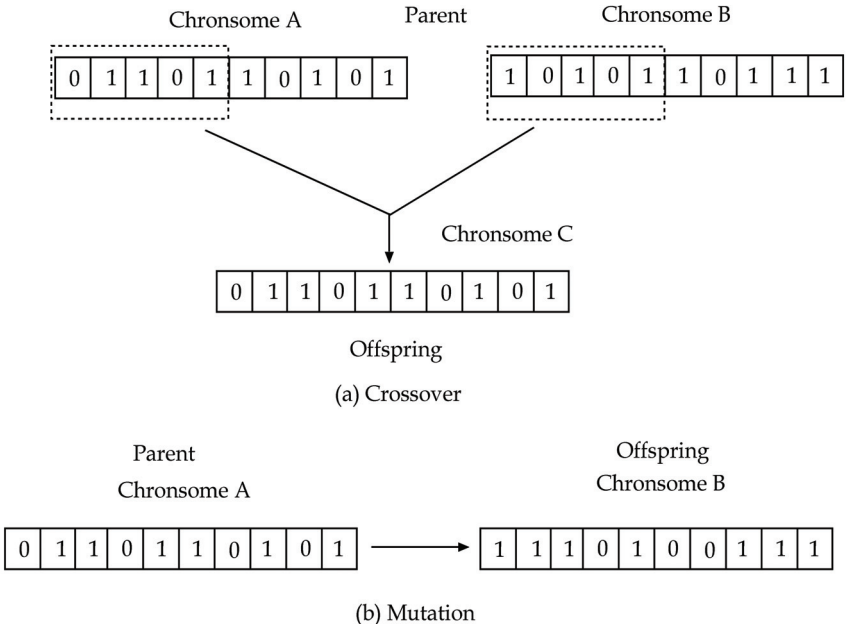


Fig. 1. Illustration of two genetic operators: (a) Crossover; (b) Mutation

4. Results

Two examples are generated to study the performance of resulting works in Sec. 3. The first example is discretizing a random field u varying with a lognormal distribution:

$$u = 1 + \log\left(0.12^2 \sum_{i=1}^{n} \xi_i\right) \tag{17}$$

where $-1 \le \xi_i$ (i = 1-2) ≤ 1 denotes n random numbers. By implementing an MCS using 10^5 samples of u, the mean μ_u and σ_u of u are calculated by 2 and 0.12; respectively. A GPC representation of u is constructed with satisfying the following accuracy standard:

$$S_j = 10\%; \quad j = 1\text{-}3 \tag{18}$$

Essential parameters for identifying n and P are set by

a. Find n and P within the range $1 \le n \le 3$ and $1 \le P \le 10$.
b. Set $\xi_a = 1$ and $\xi_b = -1.0$.
c. Set temporarily p_c and p_m are all equal to 1 and N_{max} is 100. Study subsequently the convergence of f_i, i = 1-3 with respect to different p_c and p_m values.
d. Set N = 100 in producing candidate n and P values. (That is, total 200 random numbers are generated to produce candidate n and P values). Besides, generate 100 generations of candidate n and P values.
e. Following temporarily Table 1 (e.g. Xiu and Karniadakis, 2002), apply the Hermite PC to construct an GPC representation of u. Introduce subsequently a different type of the GPC (e.g. the Legendre PC) to equate another GPC representation of u and compare the accuracy of two different GPC representations of u.

Figures 2 shows the convergence of f_i, i = 1-3. The diversity of 1st and 100th generations of random numbers for generating candidate n and P values is shown in Fig. 3. Note that less than 100 (= N) points are drawn in Fig. 2, since candidate n and P values are gotten by rounding some random numbers to their nearest integers.

Sorting the data of f_i, i = 1-3 depicted in Figs. 2-3 finds that the minization of f_3 and f_i, i = 1-2 cannot be simultaneously attained. For example, if n and P are chosen to minimize f_3 (= 0.00319), the corresponding f_i, i = 1-2 (f_1 = 0.0012; f_2 = 0.0082) are not equal to their minimum values ($f_{1,min}$ = 0.00079 and $f_{2,min}$ = 0.0001 (the subscript min denotes the minimum value)). In addition, examining Figs. 2-3 indicates that there may be two choices of final values of n and P. As listed in Table 3, if n and P are equal to 2 and 6; respectively, f_3 is minimized but f_i, i =

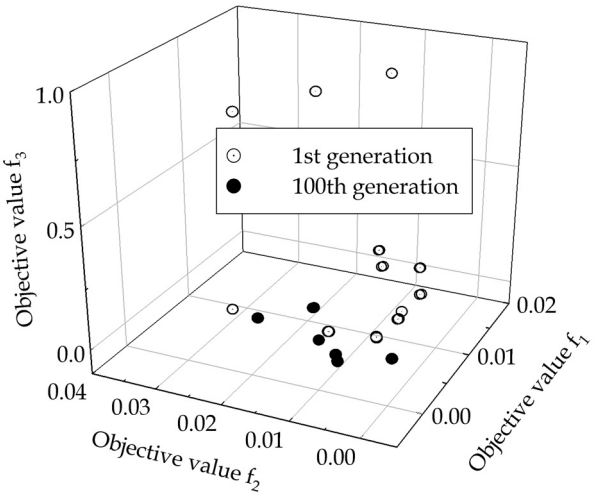


Fig. 2. Convergence of f_i, i = 1-3 (Using the Hermite PC, Lognormal distribution)

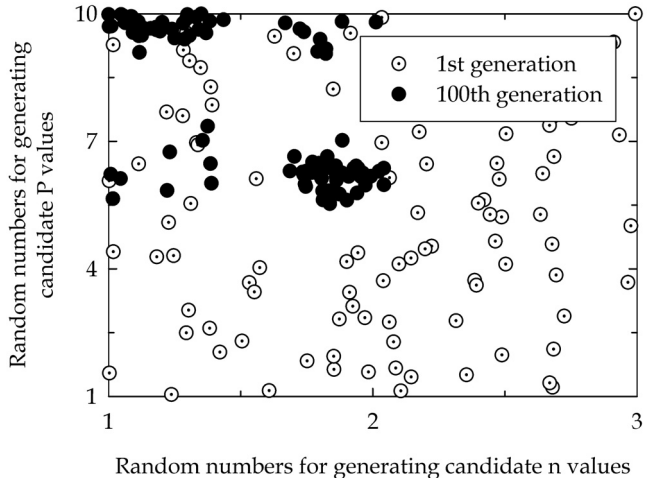


Fig. 3. Diversity of random numbers for generating candidate n and P values (Using the Hermite PC, Lognormal distribution)

1-2 are not minimized. Meanwhile, if n and P are equal to 1 and 10;respectively, f_i, i = 1-2 are minimized but f_3 is not minimized. If only one set of final values of n and P is forced to be left, this study prefers the former set. Getting a sufficiently accurate predicted standard deviation is more important.

n	P	M	f_1	f_2	f_3
2	6	28	0.0012	0.0082	0.00319
1	10	10	0.0427	0.0019	0.257

Table 3. Two different choices of final values of n and P (Using the Hermite PC, Lognormal distribution)

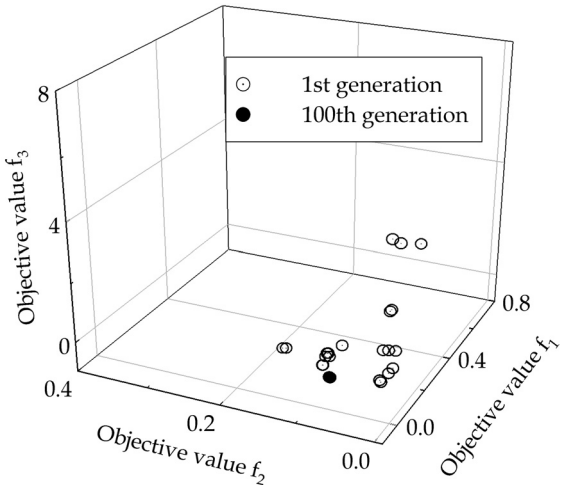


Fig. 4. Convergence of f_i, i = 1-3 (Using the Legendre PC, Lognormal distribution)

One may suspect that f_3 can be further minimized, if u is represented by another type of the GPC. As a test, Fig. 2 is modified by substituting the Legendre PC for the Hermite PC to re-discretize u. Fig. 4 depicts the convergence of resulting f_i, i = 1-3.
It seems to be difficult to compare f_3 values provided by the Legendre PC-based and Hermite PC-based representations of u by simply observing Figs. 2 and 4. Therefore, another set of n and P values, which minimize f_3, are chosen from the data for depicting Fig. 4. Table 4 lists the resulting n and P values.

n	P	M	f_1	f_2	f_3
3	2	10	0.0022	0.0778	0.08811

Table 4. Final values of n and P (Using the Legendre PC, Lognormal distribution)

Comparing Tables 3 and 4, this study suggests that a random field varying with a lognormal distribution is better represented by the Hermite PC. If the Legendre PC-based representation of u is applied, f_3 is not further minimized. In other words, the accuracy of predicted σ_u is not further improved, if the Legendre PC-based representation of u is used.
The second example is representing another random field v varying with a uniform distribution:

$$v = 1 + 0.12^2 \sum_{i=1}^{n} \xi_i \tag{19}$$

where $-1 \le \xi_i$ (i = 1-2) ≤ 1 still denote n random variables. Another MCS using 10^5 samples of v is performed. The mean value μ_v and standard deviation σ_v of v are 1 and 0.12; respectively.
Following Table 1 (e.g. Xiu and Karniadakis, 2002), the Legendre PC is applied to discretize v. With fixing other essential parameters for sketching Figs. 2-4, Fig. 5 shows the convergence of f_i, i = 1-3. The diversity of 1st and 100th generations of random numbers for generating candidate n and P values is shown in Fig. 6.

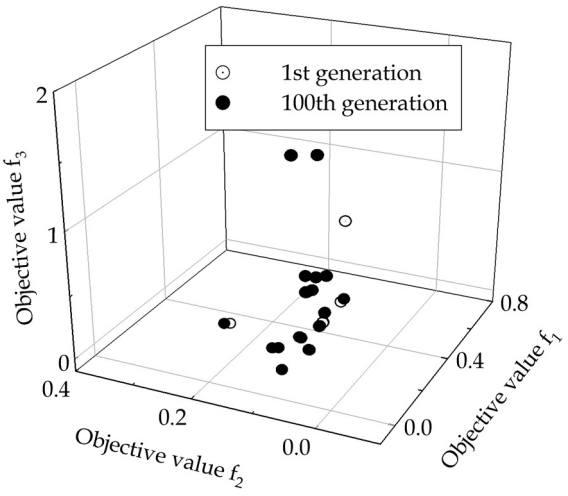


Fig. 5. Convergence of f_i, i = 1-3 (Using the Legendre PC, Uniform distribution)

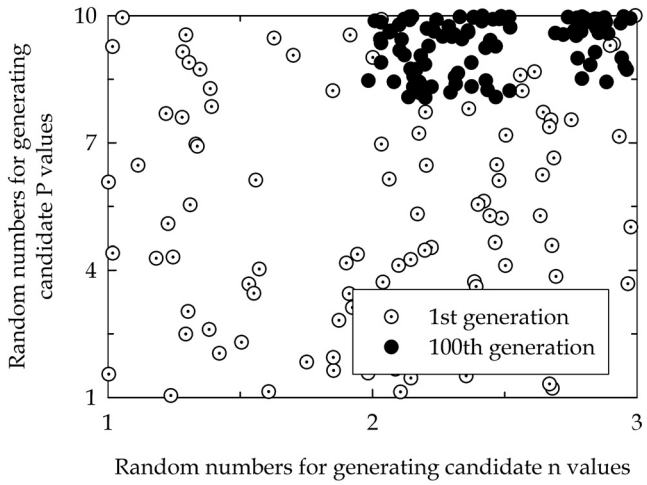


Fig. 6. Diversity of random numbers for generating candidate n and P values (Using the Legendre PC, Uniform distribution)

Sorting the data of f_i, i = 1-3 depicted in Figs. 5-6 still finds that the minization of f_3 and f_i, i = 1-2 cannot be simultaneously attained. If n and P are chosen to minimize f_3 (= 0.02635), the corresponding f_i, i = 1-2 (f_1 = 0.1363; f_2 = 0.1934) are not equal to their minimum values ($f_{1,min}$ = 0.0103 and $f_{2,min}$ = 0.0266). Also examining Figs. 5-6 finds two choices of final values of n and P. Table 5 lists these two sets of final values of n and P. If n and P are equal to 3 and 10; respectively, f_3 is minimized but f_i, i = 1-2 are not minimized. Meanwhile, if n and P are equal to 2 and 8; respectively, f_i, i = 1-2 are minimized but f_3 is not minimized. This study still prefers the former set, although more computational efforts are required in applying this set of n and P values.

n	P	M	f_1	f_2	f_3
3	10	286	0.1363	0.1934	0.0264
2	8	45	0.0103	0.0266	0.1422

Table 5. Two different choices of final values of n and P (Using the Legendre PC, Uniform distribution)

Furthermore, similarly manipulating Fig. 4, the Hermite PC is substituted for the Legendre PC to re-discretize v and the convergence of corresponding f_i, i = 1-3 is depicted in Fig. 7. Then, a set of n and P values, which minimize f_3, is chosen from the data for drawing Fig. 7. Table 6 lists the resulting n and P values.

If it is desired that the GPC representation of v should be as accurate as possible, Tables 5-6 confirms the reported conclusion (e.g. Xiu and Karniadakis, 2002) that a random field varying with a uniform distribution is better represented by the Legendre PC. Comparing Tables 5-6 indicates that the application of Hermite PC-based representation of v gives a less accurate predicted σ_v. However, if the computational efforts spent in applying a set of n and P values is the major concern, n = 2 and P = 10 may be used. The corresponding f_3 value is not too worse.

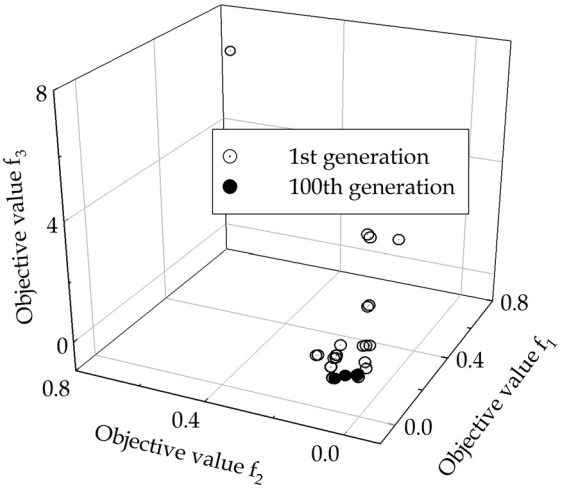


Fig. 7. Convergence of f_i, i = 1-3 (Using the Hermite PC, Uniform distribution)

n	P	M	f_1	f_2	f_3
2	10	66	0.053	0.027	0.066

Table 4. Final values of n and P (Using the Hermite PC, Uniform distribution)

Before closing this section, the effects of changing p_c and p_m on the determination of n and P values are studied. As an illustration, Figs. 3-4 are modified by changing $p_c = p_m = 1.0$ to $p_c = p_m = 0.9$. Figs. 8-9 depict the convergence of f_i, i = 1-3 and diversity of 1st and 100th generations of random numbers for producing candidate n and P values.

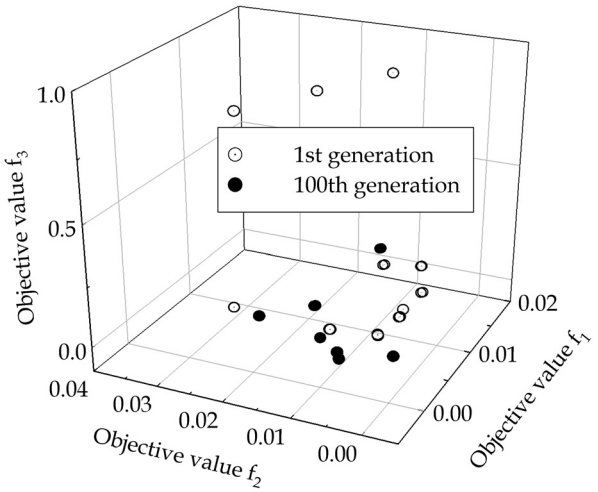


Fig. 8. Convergence of f_i, i = 1-3 with $p_c = p_m = 0.9$ (Using the Hermite PC, Lognormal distribution)

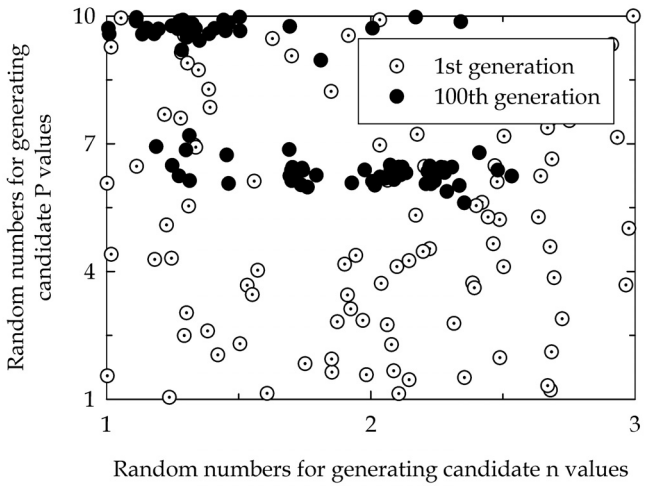


Fig. 9. Diversity of random numbers for generating candidate n and P values with $p_c = p_m = 0.9$ (Using the Hermite PC, Lognormal distribution)

Observing Figs. 8-9 finds that the decease of p_c and p_m affects insignificantly the convergence of f_i, i = 1-3 but diversifies the random numbers for generating candidate n and P values.

5. Discussion and conclusion

This study applies the SPEA2 (Zitzler, et al., 2001) to develop an auxiliary tool for identifying n and P values, which are two essential parameters for constructing a GPC representation of a random field. In Sec. 4, the proposed tool is tested to identify n and P for constructing GPC representations of two random fields varying with the lognormal and uniform distributions; respectively. The test results illustrate that an MOEA can be a good tool for constructing a sufficiently accurate GPC representation of a random field, irrespective of how many accuracy standards should be satisfied. Besides, the resulting GPC representation can be applied with keeping computational costs as few as possible.

In addition, Sec. 4 demonstrates the need of generating some pilot tests for studying the performance of GPCs in discretizing a random field before choosing one type of the GPC to discretize this random field. Such pilot tests can be quickly implemented by applying an MOEA. Tables 5-6 and Figs 5 and 7 demonstrate that the Hermite PC may be used to discretize a random field varying with a uniform distribution, if it can be sacrificed some accuracy of predicted statistical parameters (e.g. the standard deviation) of this random field.

Nevertheless, a disadvantage should be mentioned: Ψ_i, i = 0-M may be difficult to be equated in case of $n > 3$. Although much computational efforts will be spent, $n > 3$ may give more accurate GPC representation of a random field. Fortunately, consulting with some references (e.g. Field, 2004) finds that $1 \le n \le 3$ seems to be enough to generate sufficiently accurate GPC representations of random fields.

As a conclusion, an MOEA can be an efficient auxiliary tool for constructing an GPC representation of a random field satisfying multiple accuracy standards. Only Ψ_i, i = 0-M at $n > 3$ may be complementally derived in the future.

6. References

Ghanem, R. G. ; Spanos, P. D. (1991). *Stochastic Finite Elements: A Spectral Approach,* Springer-Verlag, 0387974563, New York

Xiu, D.; Karniadakis, G. E. (2003). Modelling uncertainty in flow simulations via generalized polynomial chaos. *Journal of computational physics,* Vol. 187, No. 1, 137-167, 0021-9991.

Allaix, D. L.; Carbone, V. L. (2009). Discretization of 2D random fields: A genetic algorithm approach. *Engineering Structures,* Vol. 31, Issue 5, May 2009, 1111-1119, 0141-0296.

Zitzler, E.; Laumanns, M.; Thiele, L. (2001). *SPEA2: improving the strength Pareto evolutionary algorithm.* Technical report 103. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Switzerland.

Xiu, D.; Karniadakis, G. E. (2002). The Wiener-Askey polynomial chaos for stochastic differential equations. *Journal of scientific computing,* Vol. 24, No. 2, 619-664, 0885-7474.

Field, R. V. Jr.; Grigoriu, M. (2004). On the accuracy of the polynomial chaos approximation. *Probabilistic Engineering Mechanics,* Vol. 19, Issue 1-2, January 2004, 65-80, 0266-8920.

Tan, K. C., Khor, E. F., Lee, T. H. (2005). *Multiobjective evolutionary algorithms and applications.,* Springer-Verlag, 9781852338367, London, U. K.

26

Evolutionary Algorithms in Modelling of Biosystems

Rosario Guzman-Cruz, Rodrigo Castañeda-Miranda,
Juan García-Escalante, Luis Solis-Sanchez, Daniel Alaniz-Lumbreras,
Joshua Mendoza-Jasso, Alfredo Lara-Herrera, Gerardo Ornelas-Vargas,
Efrén Gonzalez-Ramirez and Ricardo Montoya-Zamora
Universidad Autonoma de Zacatecas, Universidad Politécnica del Sur de Zacatecas
México

1. Introduction

Globally there is a land area of 13000 million ha, of which the area used for agricultural production is 1500 million ha (Naciones Unidas sobre el Desarrollo de los Recursos Hídricos en el mundo [NUDRHM], 2006), that is 12% of the total surface. Because the productivity of the agricultural sector is reduced, in general, the intensive production in controlled environments is an alternative to increase productivity in the aforementioned sector. For this reason, the use of greenhouses for agricultural production has increased in recent years. In the world, currently some 265,000 hectares are cultivated in greenhouses. Asia, with more than 138,000, represents at this moments the leading world power in intensive production under plastic cover. In the second place, with 95,000 ha, the Mediterranean basin is placed. Northern Europe (16,000 ha) and the American Continent (15,600 ha) share the rest. Holland remains the maximum exponent in agriculture under high-tech cover, especially so that which refers to crystal greenhouses.

The crop inside greenhouse allows to obtain productions of better quality and higher yields, at any time of the year, while it permits extending the cycle of farming, enabling production in the most difficult times of the year and getting better prices. Due to the benefits that provide crops inside greenhouse, crops should have a controlled microclimate using different systems to regulate environmental conditions, limiting the excesses and filling shortcomings in terms of crop needs, in order to take place in every moment the optimal conditions of different stages of plant development. In addition to the advantages that production has in controlled environments it is important to reduce production costs, such as heating costs, irrigation and fertilization. To achieve this it is necessary to have a system that controls the environment inside the greenhouse that allows improving production and crop quality, product quality, time of the production process and improving production costs. Moreover, the system should allow the farmer to have sufficient information about requirements and changes observed over time in crop development. These control systems are based on mathematical models that describe the behavior of variables, such as air temperature, the concentration of carbon dioxide (CO_2) and absolute humidity. Therefore, to improve the productivity of the crop under greenhouse it is necessary to study the process

of crop growth as well as in quantitative form moreover, the climatic environment in which the crop develops. The modeling allows having a quantitative form of the interaction of simultaneous processes (Lopez et al., 2006).
If the model is not too complex in terms of state variables it can be used to design control systems for example optimal control strategies (Seginer & Ioslovich, 1998a; Van Henten, 1994; Tap, 2000) for the best growth, production and crop quality.
Due to the advantages that production has in controlled environments, mathematical models have an essential role in the description of the main state variables of the system. Hence the importance of having a model that adequately describes the climatic conditions inside the greenhouse. That is, the problem consists of calibrating or fitting a model that describes the internal environment of the greenhouse, i.e., the problem consists in finding the set of parameters that make the difference between estimated values by the model and real values be minimal.
Then the problem of calibrating a model is reduced to a problem of search and optimization of parameters involved in the model which is formulated as a function of multivariable nonlinear optimization, but these types of problems can have multiple local optimal solutions. Some local optimization methods have been used to adjust the model parameters. However, it is necessary to use a method that allows escaping from local optima and finding the global optimum. So, for the optimization of the parameters it is possible to use a global search method such as evolutionary algorithms.
Evolutionary Algorithms (EA) techniques inspired by the theory of evolution, such as genetic algorithms (GA), Evolutionary Strategies (ES), Evolutionary Programming (EP) and Differential Evolution (DE) that are useful to solve the problem of optimization of the parameters of a mathematical model. Once the mathematical model is adjusted, the behavior of the main variables involved in the greenhouse environment can be known and, on the basis of this, take the appropriate decisions to operate the systems installed in it, such as the ventilation system, mesh shading, etc. to keep appropriate levels of temperature, humidity and CO_2 levels and in this way increase crop yields and improve final product quality.

2. Models for climate prediction

The crop production under greenhouse is influenced by the climate inside, so it has to keep state variables that characterize it within a certain range of values (CIDEIBER, 2005).
To study a model for the greenhouse production it should identify of the variables that interfere with the greenhouse climate dynamics such as state or control variables (indoor air temperature, soil temperature, internal CO_2 concentration, internal absolute humidity), external or disturbance variables (outside air temperature, external relative humidity, external CO_2 concentration, radiation, wind speed) and control variables: natural ventilation, heating). It is important to take into account that any part of the greenhouse can act as an individual body or volume. When the flow of physical quantities, such as, energy or mass is considered, the conservation law is valid for any volume.
For the prediction of environmental variables it is necessary to adjust the climate greenhouse model. Consequently, the methodology used for this adjustment is to a) defining the structure of a mathematical model that describes the climate inside a greenhouse, b) conduct a sensitivity analysis of parameters involved in the model to identify the parameters that affect more the state variables, c) making calibration of parameters identified using AE and finally e) doing the validation of the mentioned model, Figure 1.

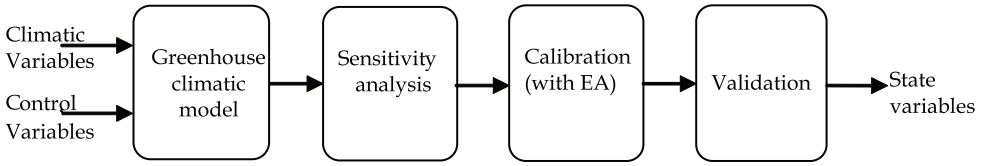


Fig. 1. Diagram of the development process for the greenhouse climate model in last years, empirical and mechanistic mathematical models have been developed to predict climate and crop growth under greenhouses.

In last years there have been developed empirical and mechanistic mathematical models to predict climate and crop growth under greenhouses.
The process of crop production in greenhouses is complex because it depends on crop growth and external climatic conditions and the design of the greenhouse. A tool to improve management of the greenhouses is mathematical models to predict the behavior of climate variables. There are complex simulation models with a relatively large number of state variables that describe a system; however, these cannot be implemented in control systems. For which it is required the use of simple models with the minimum possible number of variables for optimization and control systems. Thus, the model studied in this research is a model for optimization and control purposes.
Generally there are three types of mathematical models: white-box models, black box and gray box. White box models are deterministic and explanatory models of a system (Thornley & Johnson, 2000; France & Thornley, 2006). Typically, a set of ordinary differential equations is defined to describe the behavior of the system state variables, variables that represent the properties or attributes of the system that is under consideration. White box models are more appropriate to express hypotheses in mathematical form and thus provide a quantitative description and explanation of the most important processes occurring in a system. Black box models are direct descriptions of data and provide direct observable relationships between the variables in a system without any explanation of the underlying mechanisms. They are a powerful means to describe and summarize data. Grey box models are used when some physical understanding is available, but several parameters stay to determine the observed data.

2.1 White box models

The development of a white-box mathematical model requires sufficient understanding of the physical, chemical and biological processes that occur in a system and its use demands a proper validation. Explanatory models can be static or dynamic. The simulation of crop growth and development involves two processes construction of mathematical models and numerical solution of the set of equations that describe the behavior of the system, through the use of a digital computer.
Dynamic simulation models are based on the assumption that the state of a system can be quantified and that changes in the state can be described by mathematical equations, equations of rate of change or differential equations. A model of growth and development of crop includes several components: state variables, differential equations, parameters and inputs. Normally a state variable is a variable that appears in the accumulation term of a dynamic balance of mass or energy. A state variable is a variable that can be quantified (at least conceptually) and it allows knowing the behavior of the system at all future instant in

time. The differential equations represent the change velocity of the state variables (López Cruz, 2004). Crop growth or climate in a greenhouse is described by a dynamic model, represented by a nonlinear ordinary differential equation as follows:

$$\dot{x} = f(x,u,\theta,t), \qquad x(t_b) = \beta \tag{1}$$

Where $x \in \Re^n$ are the state variables, $u \in \Re^m$ are the control inputs, $\theta \in \Re^q$ are time-invariant parameters, β is the vector of initial conditions and t denotes the time (Van Henten & Van Straten, 1994).

From the 60's to date, models for crop growth have been developed. The most known results are the model of crop growth SUCROS (a Simple and Universal Crop Growth Simulator) and LINTUL (Light Interception and Utilization) which are generic models (Bouman et al, 1996) i.e., they can be adapted to any crop year (Hernández Hernández, 2009).

Moreover, in the last 30 years mechanistic mathematical models have been applied in modeling of greenhouse climate. Two of the first proposals were made in 1983 by Boy and Udink Ten Cate. Both models consider equations for the temperature of the greenhouse warming effect and opening windows.

2.2 Black box models

The problem of identifying a given system consists of given certain inputs, $u(t)$, and outputs, $y(t)$, of n dynamic system:

$$u^t(t) = \left[u(1), u(2), ..., u(t)\right] \tag{2}$$

$$y^t(t) = \left[y(1), y(2), ..., y(t)\right] \tag{3}$$

It is looked for the relationship between past observations $\left[u^{t-1}, y^{t-1}\right]$ and future outputs $y(t)$:

$$y(x) = g\left(u^{t-1}, y^{t-1}\right) + v(t) \tag{4}$$

The term $v(t)$ considers the fact that the output $y(x)$ will not be an exact function of past data, $v(t)$ is described as a random noise signal. However, a goal should be that $v(t)$ be as small as possible so that you can think that $g\left(u^{t-1}, y^{t-1}\right)$ is a good prediction of $y(x)$ given past data. Eq. (4) models general dynamic systems in discrete time. Static systems can be viewed as particular cases of dynamical systems.

Now, the problem consists of finding a function g in (4). So, it has to look for a family of functions. This family of functions can be parameterized by a vector of parameters θ with finite dimension:

$$g\left(u^{t-1}, y^{t-1}, \theta\right) \tag{5}$$

Parameterize the function g with a vector θ of finite-dimension is usually an approximation. To find a good parameterization, it is necessary to decide on a structure and having a set of data collected $\left[u^N, y^N\right]$, the quality of θ can be evaluated by adjusting the model and the registered data:

$$\sum_{t=1}^{N} \left\| y(t) - g\left(u^{t-1}, y^{t-1}, \theta\right) \right\|^2 \tag{6}$$

The standard and the real way to achieve or try to achieve minimum in θ may differ but many schemes of system identification follow this concept.
Now the family structure models (5) is too general and useful to write g as a concatenation of two mappings: one that takes the increasing number of past observations u^t, y^t and maps them in a vector $\varphi(t)$ of fixed dimension and one that maps this vector to space of outputs:

$$g\left(u^{t-1}, y^{t-1}, \theta\right) = g\left(\varphi(t), \theta\right) \tag{7}$$

where

$$\varphi(t) = \varphi\left(u^{t-1}, y^{t-1}\right) \tag{8}$$

This vector is called vector regression and its components are called regressors. The regressor vector can be parameterized as:

$$\varphi(t) = \varphi\left(u^{t-1}, y^{t-1}\right) \tag{9}$$

Which can be denoted as $\varphi(t, \eta)$. Sometimes $\eta = \theta$, that is the regression vector depends on all model parameters (Sjöberg, 1995).
Black box linear models for dynamic systems can be described with one $g(\cdot, \cdot)$ selected to be a linear mapping, $\theta^T \varphi(t)$. Regressor selection $\varphi(t)$ specifies whether the model is an ARX, ARMAX, a state space model, etc. Consider the same type of regressors but using a nonlinear mapping given black-box nonlinear models.
The selection of a nonlinear mapping (5) is separated into two partial problems for dynamic systems:

1. How to choose the regression vector $\varphi(t)$ of inputs and past outputs.
2. How to choose the nonlinear mapping $g(\varphi)$ from regressor space to output space

2.3 Gray box models

A gray box model corresponds to a combination of a black box model with a white box model. This model is also named semi-physique or based on knowledge, because in it all the knowledge of the process is introduced (or part of it) and additionally those unknown parameters are estimated through system measurements (Moreno and Acuña, 2005).

3. Evolutionary algorithms

As mentioned above, mathematical models have a set of parameters that need to be adjusted so that the data estimated by the model is more closely to the observed data. This problem of parameter estimation is also known as calibration parameters problem.
The task of calibrating a model is formulated as an optimization problem, so that different methods of solution have been proposed. Local or global methods of optimization can be used to solve the calibration problem. Some researchers have used local optimization

methods (Tap, 2000, Linker et al., 2004) or manual calibration (Van Henten, 1994) to adjust the parameters of a greenhouse climate model. A method for calibrating a mathematical model is using a nonlinear multivariable optimization function, but it is well known that these optimization problems may have optimal local solutions. Such problems are known as multimodal (Eiben & Smith, 2003). However, in recent years it has increased the use of global optimization methods to solve these kind of problems (Michalewicz, 1994) due to the advantages of getting global optimal possible solutions. For this reason, there is a need to develop algorithms based on this methodology to be applied to greenhouse climate. According to Eiben and Smith (2003), evolutionary algorithms (EA) could be an excellent alternative to provide a response to the challenge of achieving automated solution methods for problems more complicated and faster. EA are stochastic search methods that include genetic algorithms (GA), Evolutionary Strategies (ES), Evolutionary Programming (EP) and Differential Evolution (DE) (Michalewicz, 1994).

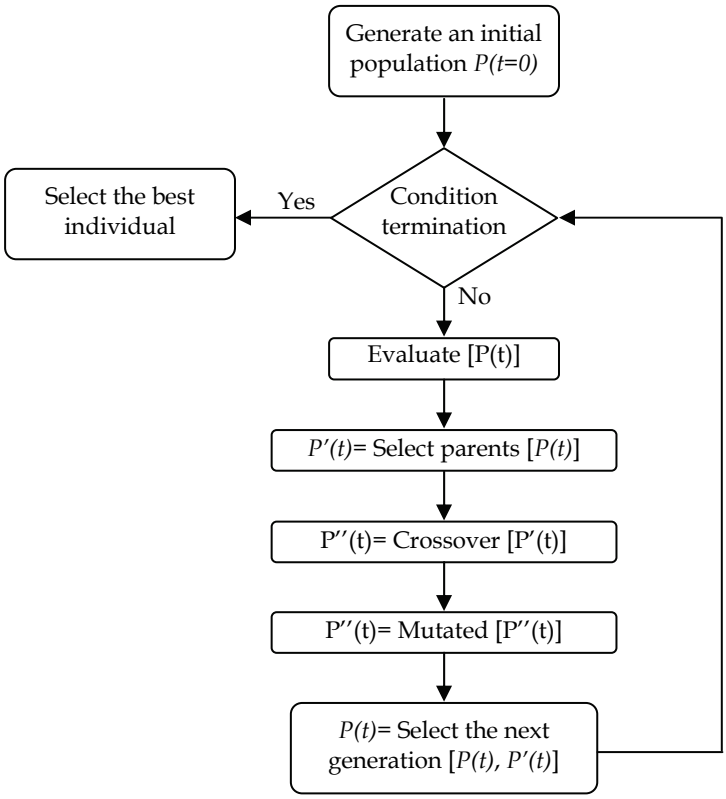


Fig. 2. Diagram of a general scheme of an evolutionary algorithm.

The structure of any AE is the same (Eiben & Smith, 2003), Fig. 3.3. The differences between evolutionary techniques consist of the type of selection operator, mutation and crossover applied to find the optimal value of the parameters to calibrate.

In general, it takes five basic components to implement an EA that solves a any problem (Michalewicz, 1992):

1. A representation of potential solutions to the problem.
2. One way to create an initial population of potential solutions (this is usually done randomly, but it can also be used deterministic methods).
3. An evaluation function that plays the role of the environment, qualifying the solutions produced in terms of its "fitness."
4. Genetic operators that alter the composition of the offspring (normally using the crossing and mutation).
5. Values for the various parameters used by the genetic algorithm (population size, cross and mutation probability, maximum number of generations, etc.).

3.1 Genetic algorithms

The traditional representation used in GA to encode a set of solutions that is the binary scheme, i.e. it is a string formed by zeros and ones (Eiben and Smith, 2003). However, to solve the problem of setting parameters of a greenhouse climate model to a series of real values is used to represent a candidate solution to a problem, i.e. a vector $(p_1,...,p_n)$ where $p_i \in R$ and n is the number of parameters that are needed to be calibrated.

3.2 Parental choice

N is the population size P. The parent selection is determined by tournament (Coello, 2007) which consists of:

1. Shuffle to the individuals in the population.
2. Choose k individuals in the population and comparing them with the basis of their adaptability (typically k = 2).
3. The fittest individual is the winner of the tournament. "
4. The population is shuffled k times to select N parents

3.3 Crossing

The cross is an operator that forms a new individual by combining parts of each parent. In this case, the type of employed crossing is crossbreeding in two points where both parents are employed to generate a new individual and crossover points are chosen randomly (Coello, 2007). Be $p,\ p' \in P$ and $i, j \in \{1,2,...,n\}$ random crossing points, so parents.

$$p = \left(p_1,...,p_{i-i},p_i,p_{i+1},...,p_{j-1},p_j,p_{j+1},...,p_n\right) \tag{10a}$$

and

$$p' = \left(p'_1,...,p'_{i-i},p'_i,p'_{i+1},...,p'_{j-1},p'_j,p'_{j+1},...,p'_n\right) \tag{10b}$$

Generate the children

$$h = \left(p_1,...,p_{i-i},p_i,p'_{i+1},...,p'_{j-1},p_j,p_{j+1},...,p_n\right) \tag{11a}$$

And

$$h' = \left(p'_1,...,p'_{i-i},p'_i,p_{i+1},...,p_{j-1},p'_j,p'_{j+1},...,p'_n\right) \tag{11b}$$

3.4 Mutation

The type of mutation is not applied uniformly (Michalewicz, 1996), where the position of the individual that will be altered is selected randomly. Given $p=(p_1,...,p_{,n})$, this is changed by the individual mutant $p=(p_1,...,p_i',...,p_s)$ where

$$p_i' = \begin{cases} p_i + \Delta(t, U - p_i), & \text{si } R = 0 \\ p_i - \Delta(t, p_i - L), & \text{si } R = 1 \end{cases} \tag{12}$$

$p_i \in [L,U]$, $R \in \{0,1\}$ are chosen randomly and

$$\Delta(t,y) = y * \left(1 - r^{\left(1-\frac{t}{T}\right)^b} \right) \tag{13}$$

Where $r \in [0,1]$ is a random, T is the maximum number of generations and b is the degree of non-uniformity of the mutation (Michalewicz use b = 5).

3.5 Survival selection

The survival selection is used in the replacement based on age, that is, on each generation parents are replaced by children who now represent the current population.

3.6 Evolutionary strategies

The type of representation used to represent a candidate solution for a problem is real.

3.7 Parental choice

Parents selection is random with uniform distribution of the population of N individuals.

3.8 Crossing

The type of crossing used is discrete where each element of the generated child is obtained randomly from the elements of parents with an equal probability for both parents. That is, given $p,\ p' \in P$ where $p=(p_1,...,p_n)$ and $p'=(p'_1,...,p'_n)$, a child $h=(h_1,...,h_n)$ is created with

$$h_i = \begin{cases} p_i & \text{si } R = 0 \\ p'_i & \text{si } R = 1 \end{cases} \tag{14}$$

where $i \in \{1,...,n\}$ and $R \in \{0,1\}$ are chosen randomly

3.9 Mutation

In this case, the mutation is not correlated with n step sizes is used which consist on a given $p=(p_1,...,p_n)$, p is extended in n step sizes thus resulting $p=(p_1,...,p_n,\sigma_1,...,\sigma_n)$. Then, the mutated individual is $p'=(p'_1,...,p'_n)$ where

$$\begin{aligned} p_i' &= p_i + \sigma_i' \cdot N_i(0,1), \\ \sigma_i' &= \sigma_i \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)} \end{aligned} \tag{15}$$

with $\tau' = \frac{1}{\sqrt{2n}}$ and $\tau = \frac{1}{\sqrt{2\sqrt{n}}}$

3.10 Survival selection

Let N be the number of elements of the population and M the number of children generated. The selection of survival used is deterministic of type (N, M) in which after creating M children and calculating its adaptability, the N best are chosen to move to the next generation (Eiben & Smith, 2003).

3.11 Evolutionary programming

The type of representation used to represent a candidate solution to a problem is real.

3.12 Parental choice

Deterministic selection is used in which each parent generates exactly one child via mutation.

3.13 Crossing

Evolutionary programming is a technique in which there is no crossing between individuals of the population.

3.14 Mutation

A Gaussian perturbation is applied. That is, given $p = (p_1,...,p_n,\sigma_1,...,\sigma_n)$ is transformed into $p' = (p'_1,...,p'_n,\sigma'_1,...,\sigma'_n)$ where

$$\begin{aligned} p_i' &= p_i + \sigma_i' \cdot N_i(0,1), \\ \sigma_i' &= \sigma_i \cdot (1 + \alpha \cdot N(0,1)) \end{aligned} \tag{16}$$

With $\alpha \approx 0.2$.

3.15 Survival selection

The survival selection is deterministic of type (N + N) in which competition between each pair is through tournament after creating N children of the N individuals in the population. Therefore, let P 'represents the total population that includes parents and children, then each individual $p \in P'$ is compared with other q individuals selected randomly. In each comparison, if p is better than its opponent a gain is assigned to p. The N individuals with the greatest values of gain are selected to be parents of the next generation. Typically, q = 10 is recommended (Eiben & Smith, 2003).

3.16 Differential evolution

Representation of a candidate solution to a problem in ED is of real type. Thus that to solve the problem of setting parameters of a greenhouse climate model a series of real values is used to represent a candidate solution for a problem, i.e. a vector $p_j = (p_{j1},...,p_{jn})$ where j = 1, ..., N, $p_i \in R$ N is the number of elements of the population and n is the number of parameters that need to be calibrated (Price and Storn, 2005).

The initial population is chosen randomly if nothing is known about the problem. As a rule, it is assumed a uniform distribution for random decision making. In ED different strategies can be adopted (Price and Storn, 2005). The main idea behind the ED is a new scheme to generate vectors. The ED generates these new vectors when adding the weight difference between two members of the population vectors to a third vector member. If the fitness of the resulting vector is less than the population member chosen then the new vector replaces the vector with which it was compared. This vector to be compared can be (though not necessarily is) part of the generation process. In addition, the best individual is evaluated in each generation G for keeping track of progress during which it is done minimization.
There are different variations in ED, in this case, it is applied the classic strategy ED/random/1/bin in the calibration problem.

3.17 Selection

The selection of vectors to disturb is random.

3.18 Mutation

The type of mutation is randomly applied which consists of generating N vectors of the form:

$$v = p_1 + F \cdot (p_2 - p_3) \tag{17}$$

where $p_1, p_2, p_3 \in \mathrm{P}$ are different, $F \in [0,2]$ is a real constant factor that controls the amplification of the differential variation between the vectors p_2 y p_3 .

3.19 Crossing

The crossing applied is binomial type which consists of combining the previously mutated vector $v = (v_1, v_2, ..., v_n)$ with a target vector $p = (p_1, p_2, ..., p_n) \in \mathrm{P}$ to generate a test vector $p' = (p'_1, p'_2, ..., p'_n)$ where

$$p'_j = \begin{cases} v_j & \text{if } r_j \le CR \\ p_j & \text{other case} \end{cases} \tag{18}$$

with $r_j \in [0,1]$ random and $CR \in [0,1]$ is the constant of crossing, a parameter that increases the diversity of individuals in the population.

3.20 Survival selection

For the selection of individuals that move to the next generation a simple selection one to one is used where the trial vector competes with the target vector. For the case of calibration in which minimization of J (p), the vector with the lowest value in the objective function goes to the next generation.

4. Case study and results

Table 1 shows a summary of the techniques used in each evolutionary algorithm. . To each algorithm, the parent selection, the crossover and the mutation were chosen based on the advantages that these have over the others (Michalewicz, 1996; Eiben and Smith, 2003; Coello, 2007).

	Genetic Algorithms	Evolutionary Strategies	Evolutionary Programming	Differential Evolution
Representation	Real-valued	Real-valued	Real-valued	Real-valued
Parent selection	Deterministic: by mean tournament	Uniform random	Deterministic (each parent creates one offspring)	Random
Recombination	2-point crossover	Discrete	None	Binomial
Mutation	Non uniform	Uncorrelated with n step sizes	Gaussian perturbation	Random
Survival Selection	Generational	Determination: (μ, λ)	Determinations: $(\mu + \mu)$	Elitist

Table 1. Comparative table of the main techniques of the evolutionary algorithms

4.1 Study cases

Some general statistics are used to analyze the obtained results, such as the correlation coefficient, r, which establishes the association degree between the measured and simulated variable. It is defined by means

$$r = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2 \sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}} \tag{19}$$

Where x_i y y_i are measured data and estimated data, respectively, on the time i. making sure that $-1 \leq r \leq 1$.

The standard percentage error of the prediction (%SEP) which establishes the dispersion degree between the measured and simulated variable.

$$\%\ ESP = \frac{100}{\bar{x}}\sqrt{\frac{\sum_{i=1}^{n}\left(x_i - y_i\right)^2}{n}} \tag{20}$$

The efficiency coefficient (E) and the average relative variance (ARV). These estimators are used to determine the way in which the model can explain the total variance of the data (Ríos-Moreno et al., 2006).

$$E = \frac{S_{obs} - S}{S_{obs}} \tag{21}$$

and

$$\Delta\left(t, y\right) = y * \left(1 - r^{\left(1 - \frac{t}{T}\right)^b}\right) \tag{22}$$

where

$$S_{obs} = \sum_{i=1}^{n} \left(x_i - \bar{x} \right)^2 \tag{23}$$

and

$$S = \sum_{i=1}^{n} \left(x_i - y_i \right)^2 \tag{24}$$

For a perfect relation, r and E should be near 1 and the values of %SEP and ARV near 0.

4.2 White box model

The model proposed by Tap (2000) was considered, one of the most complete and available in the literature, which is associated a differential equation to each one of the state variables that are: air temperature (T_i), soil temperature (T_s) CO_2 concentration (C_i) and absolute humidity inside (V_i).

The equation of the greenhouse air temperature indicates that the variation of temperature inside the greenhouse is proportional to the heat exchange ventilation, heat input due to the heating system, to the exchange through the roof and walls, to heat exchange with the deep soil, to heat input by radiation, to the evaporative heat loss due to transpiration as well as the interchange due to condensation on the roof of the greenhouse. So, the rate of change of air temperature (T_i) with regards to time is described by:

$$C_g \frac{dT_i}{dt} = k_v \left(T_o - T_i \right) + H + k_r \left(T_o - T_i \right) + k_s \left(T_s - T_i \right) + \eta G - \lambda E + \frac{\lambda}{\varepsilon + 1} M_c \tag{25}$$

Where C_g is the greenhouse heat capacity expressed in J $^{o}C^{-1}$ m^{-2}, k_v is the coefficient of heat transfer vent W $^{o}C^{-1}$ m^{-2}, T_o is the outside temperature in ^{o}C, H is the heat input for heat, k_r is the coefficient of heat of ventilation from the cover in W $^{o}C^{-1}$ m^{-2}, k_s is the coefficient of heat transfer from the ground in W $^{o}C^{-1}$ m^{-2}, η is the solar efficiency factor, G is the radiation in W m^{-2}, E is the crop transpiration, λ is the energy of vaporization of water in J g^{-1}, ε heat resistance from the cover between inside and outside and M_c is the condensation of water on the cover.

Then a more detailed description of each one of the parameters that interfere in equation (25), which represents the rate of change of air temperature inside the greenhouse. So, it defines the energy of vaporization of water such as:

$$\lambda = l_1 + l_2 T_i \tag{26}$$

where l_1 and l_2 are the coefficients of energy of vaporization.

The heat transfer coefficient ventilation k_v is a function of ventilation rate Φ_v

$$k_v = M_{air} c_p \Phi_v \tag{27}$$

where M_{air} is the air density and c_p the specific heat of air.

The relationship for the natural ventilation flow through windows Φ_v, does not differ between the opening of barlovento windows (r_{ww}) and sotavento (r_{wl}), when in fact these

have a very different influence on ventilation (from Jong, 1990). Thus, the way to generalize this relationship is:

$$\Phi_v = \left(\frac{\sigma r_{wl}}{1 + \chi r_{wl}} + \zeta + \xi r_{ww} \right) w + \psi \tag{28}$$

Where σ, χ, ς, ξ and ψ are parameters that determine the ventilation rate and w represents wind speed. This relationship was developed by Van Henten (1994), and it is based on work by Jong (1990).

Another important factor that interferes with both the air temperature change and in the absolute humidity is the crop transpiration E, which is calculated by an adopted version of the Penman-Monteith transpiration. The Penman-Monteith model was chosen because it does not need the leaf temperature as input. In this sense, it is saved a state variable, while the model is still reasonably accurate (Jolliet & Bailey, 1992).

$$E = \frac{sn\eta G + \rho_a c_p D_g g_b}{\lambda \left(s + \gamma \left(1 + \frac{g_b}{g} \right) \right)} \tag{29}$$

The Penman-Monteith equation assumes that the leaf area index (LAI) is one that is the crop in the greenhouse is conceived as a big leaf. Basically transpiration is generated by a contribution of the net short-wave radiation absorbed nηG and a contribution due to the vapor pressure deficit D_g, γ is the apparent psychometric constant and g is the conductance of the leaf. In Eq. (30), the slope of the curve of saturated steam pressure (s) can be approximated by the polynomial

$$s = s_1 T_i^2 + s_2 T_i + s_3 \tag{30}$$

Where s_1, s_2 and s_3 are polynomial parameters, $\rho_a c_p$ represents the heat capacity of air volume, gb is the conductance of the leaf. All these quantities are assumed constant. The vapor pressure deficit of air D_g, is calculated as the difference between saturated vapor pressure p_g^* a T_i (Hanan, 1998)

$$p_g^* = a_1 e^{\frac{a_2 T_i}{a_3 + T_i}} \tag{31}$$

And air vapor pressure to the prevailing water content p_g

$$p_g = \Lambda \left(T_i + T_0 \right) V_i \tag{32}$$

Where Λ is the constant of pressure that can be obtained from the ideal gas laws, T_0 is used to convert T_i ^oC to Kelvin and V_i the concentration of water vapor in the air. So

$$D_g = p_g^* - p_g \tag{33}$$

The conductance of the leaf is related to short-wave radiation and the concentration of CO_2 through the regression equation

$$g = g_1\left(1 - g_2 e^{-g_3 G}\right) e^{-g_4 C_i} \tag{34}$$

Where from g_1 to g_4 are regression coefficients.
A detailed modelling of the transport of water vapor to the cover would require a description of the cover temperature, humidity inside the greenhouse, the possible re-evaporation of condensation, and the present quantity of condensation on the cover. To calculate the condensation are needed humidity inside the greenhouse (V_i) and the cover temperature (T_c). To keep the model as simple as possible, the temperature of the cover will be calculated based on the relationship obtained by Bakker (1995), which describes the temperature of the cover as an algebraic average between the outdoor temperature and indoor temperature

$$T_c = \frac{\varepsilon}{\varepsilon + 1} T_o + \frac{1}{\varepsilon} T_i \tag{35}$$

In this sense, the influence of the heat capacity of the cover and short-wave radiation from the sky are ignored. So, the approximation (3.8) is allowed, as Bakker (1995) showed that even instantly condensation can be incorrect, an average of condensation on the basis of a day is correct. Condensation on the greenhouse cover takes place when the roof temperature (T_c) is below the dew point (dew point) from the air of the greenhouse. Introducing the index of humidity as the mass of water vapor per humid air mass unit, condensation takes place when the percentage of humidity under pressure of saturated steam on the cover (W_c^*) is lower than the percentage of air humidity (W_g). The superscript * indicates that the amount considered is saturated vapor pressure. T_0 complete W_c^*, first the pressure of saturated steam at T_c is calculated according to Eq. (35) (replacing T_i by T_c). The vapor pressure in the air of the greenhouse (p_g) is calculated according to Eq. (32). When it is known the vapour pressure, humidity ratio W can be calculated as follows

$$W = \frac{\omega p}{p_{atm} - p} \tag{36}$$

Where ω is the parameter of the ratio of humidity and p_{atm} is the atmospheric air pressure. W_g or W_c^* can be calculated substituting p_g or p_c for p, respectively.
Used W_c^* and W_g, the rate of condensation (M_c) is calculated as follows

$$M_c = \begin{cases} m_1 \left| T_i - T_c \right|^{m_2} \left(W_g - W_c^* \right), & if \quad W_g > W_c^* \\ 0, & if \quad W_g \le W_c^* \end{cases} \tag{37}$$

Where $m_1 \left| T_i - T_c \right| m_2$ is the mass transfer coefficient, m_1 and m_2 are the parameters of mass transfer coefficients. In the process of condensation, water and energy are transported simultaneously. At the moment that water condenses, the energy of condensation is released into the surrounding environment. This is why condensation is part of the temperature equation (25) and the humidity equation (40).
In the case of the equation for soil temperature it is indicated that it is proportional to the heat exchange between the surface layer and the ambient temperature and the heat exchange between the surface layer and the deep soil. Thus, the rate of change of soil temperature (T_s) respect to time is:

$$C_s \frac{dT_s}{dt} = -k_s \left(T_s - T_i\right) + k_d \left(T_p - T_s\right) \tag{38}$$

Where C_s is the heat capacity of soil in $J \cdot {}^oC^{-1} \cdot m^{-2}$, k_d coefficient of heat transfer from layer to layer of soil in $W \cdot {}^oC^{-1} \cdot m^{-2}$, T_p is the temperature of the deep layer soil in °C.
For the CO_2, concentration, the equation establishes that the change in CO_2 concentration is proportional to CO_2 exchange with the outside, to the injection of CO_2 to increase of CO_2 by respiration as well as the reduction of CO_2 by photosynthesis. Thus, the rate of change of CO_2 concentration (C_i) with respect to time is described by:

$$\frac{V_g}{A_g} \frac{dC_i}{dt} = \Phi_v \left(C_o - C_i\right) + \varphi_{inj} + R - \mu P \tag{39}$$

V_g/A_g is the average height of the greenhouse in m, Φ_v is the ventilation flow in $m \cdot s^{-1}$, C_o is the external CO_2 concentration in $g \cdot m^{-3}$, φ_{inj} is the CO_2 injection flow $g \cdot s^{-1} \cdot m^{-2}$, R is the respiration of the crop in $g \cdot s^{-1} \cdot m^{-2}$, P is the photosynthesis of the crop in $g \cdot s^{-1} \cdot m^{-2}$ and μ is the molar fraction of CO_2 and CH_2O.
For humidity, the equation indicates that the change of humidity inside the greenhouse is proportional to the increase of humidity by transpiration, to the exchange of humidity by respiration as well as the loss of humidity from condensation on the cover of the greenhouse. Thus, the rate of change of humidity in the greenhouse (V_i) respect to time is:

$$\frac{dV_i}{dt} = \frac{A_g}{V_g} \left(E - \Phi_v \left(V_i - V_o\right) - M_c\right) \tag{40}$$

Where V_o is the outdoor humidity, in $kg \cdot m^{-3}$.
As mentioned before, measurements were made from climate variables involved in the model that describes the greenhouse environment, among them it is found the outdoor relative humidity. However, the model has as one of its inputs to external absolute humidity, so it is estimated as absolute humidity (AH) from air temperature (T) and relative humidity (RH) by means of the following expression (Hanan, 1998):

$$HA = \frac{18 e_1}{29\left(1 - e_1\right)} \tag{41}$$

where

$$e_1 = \frac{HR}{100} \frac{e_2 / 18}{1/29 + e_2/18} \tag{42}$$

$$e_2 = 3.77 \times 10^{-3} 2.965 \times 10^{-4} T + 5.2 \times 10^{-6} T^2 + 3.7 \times 10^{-7} T^3 \tag{43}$$

Moreover doing the estimation of the four state variables which has the mathematical model, it is determined from the internal relative humidity because it is most used the unit in practice, calculated by the following equation (Hanan, 1998):

$$HA = 100 \frac{p_g}{p_g^*} \tag{44}$$

where p_g is the vapor pressure of air in the greenhouse and p_g^* is the saturated vapor pressure of air in the greenhouse.
In order to calibrate a mathematical model properly, it is important to carry out a sensitivity analysis (Van Henten, 2003), which evaluates the relative importance of input variables and model parameters on the evolution over time of the model's state variables (Saltelli et al., 2000). Then model calibration is performed. Finally, a validation process is completed, by comparing simulation results using parameter values obtained from calibration, with measurements which were not used during model calibration.
To achieve a good fit of a greenhouse climatic model, it is necessary to find suitable values for the parameters in the model. In order to find the parameters that most affect the state variables, it is necessary to do a sensitivity analysis of the model.
The time evolution of the sensitivity model (S(t)) of all states with respect to all parameters is defined as

$$S(t) = \frac{\partial x(t)}{\partial p} \tag{45}$$

where $x(t) \in R^n$ is the evolution time of the state vector, $p \in R^m$ is the parameter vector and t denotes time (Tap, 2000). Thus, S(t) is a $n \times m$ dimensional matrix, where every element represents the evolution time of the derivative of one state by one parameter. Eq. (46) is an approximation of the time evolution of

$$\frac{dS}{dt} \approx \frac{\partial f}{\partial x} S + \frac{\partial f}{\partial p}, \qquad S(t_0) = 0 \tag{46}$$

Where the right-hand side of the output equation is defined by f:

$$\frac{dx(t)}{dt} = f(x, u, p) \tag{47}$$

Where u is the input vector (controls) which does not depend on p. To be able to make a good comparison between the different sensitivities the relative sensitivity S_{rel} is calculated, as given by the following equation:

$$S_{rel} = S\frac{p}{y} = \frac{\partial y}{\partial p}\frac{p}{y} \tag{48}$$

A drawback is that this can yield numerical problems when x is very small or equal to zero. Parameter ranking is done on the basis of

$$I = \int_{t_0}^{t_f} |S_{rel}| \, dt \tag{49}$$

where t_0 is the initial time and t_f the final time.
The following step is the process of calibration of the model which consists of altering parameters to obtain a better fit between the simulated and measured data. An appropriate method to make the calibration is to use a non-linear multivariable optimization function (Tap, 2000), to minimize the sum of square errors (J):

$$J(p) = \sum_{h=1}^{L}\sum_{i=1}^{M}\sum_{j=1}^{N} w_h\left(\overline{y_h}(t_i, p) - y_{hj}(t_i)\right)^2; \qquad p^* = \arg\min J(p) \tag{50}$$

Where w_h is the relative weight of each output , $y_h(t_i, p)$ is the simulated output, y_h in time t_i, $y_{hj}(t_i)$ is the j-th repetition of the measurement y_h in time t_i, L is the number of outputs, M is the number of real measurements (time), N number of repetitions in each real measurement (time), p is the parameter set of calibration and p* are the parameters that reduce J(p) to a minimum. The weights w_h determine the relative importance of the different outputs in Eq. (50). These were calculated by normalization of the output vector to avoid problems with the units of the state variables.

4.3 Black box model

In Other control the air temperature inside of a closed environment it Can Be Modeled by auto regressive models with external inputs (ARX) and by auto regressive moving average models with external inputs (ARMAX) considering the inputs and outputs Are Measured by sensors. For a system with one input and one output (SISO) the model is Given by Eq. (1) (Ljung, 1999, Aguado & Martinez, 2003; Ljung, 2005)

$$\begin{aligned} y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) = \\ b_1 u(t-n_k) + \ldots + b_{n_b} u(t-n_k-n_b+1) + e(t) + c_1 e(t-1) + \ldots + c_{n_c} e(t-n_c) \end{aligned} \tag{51}$$

Where y(t) is the output of the ARX and ARMAX models for t = 1, t-1,… t-na; u(t) is the input for t = t-nk, t-nk -1,…, t-nk-nb +1; na is the number of samples passed in the time of the output; nk is the delay time of the input u(t), e(t) is the white noise associated with the input and t is discrete time.

To evaluate the temperature inside of a closed environmental using ARX and ARMAX models, more input variables are required, so the models have multiple inputs and one output (MISO). The structures ARX and ARMAX for MISO systems are defined by Eq. (2) and (3), respectively

$$A(q)y(t) = B(q)u(t-n_k) + e(t) \tag{52}$$

Where A(q), B(q) are matrices and C(q) is a vector, all defined by Eq. (53) –(54):

$$\text{A(q):}\ 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \tag{53}$$

$$\text{B(q):}\ 1 + b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b} \tag{54}$$

And the operator q^{-1} is the backward shift operator

$$q^{-1}u(t) = u(t-1) \tag{55}$$

For a system in which the number of inputs is given by n_y and the number of outputs by nu, A(q) and B(q) are n_y by n_y and n_u by n_u matrices, respectively, whose elements are polynomials in the shift operator q-m (with m any natural number). The entries $a_{ij}(q)$ and $b_{ij}(q)$ of the matrices A(q) and B(q), respectively, can then be written as

$$a_{ij}(q) = \delta_{ij} + a_{1_{ij}} z^{-1} + \cdots + a_{n_{a_{ij}}} z^{-n_{a_{ij}}} \tag{56}$$

and

$$b_{ij}(q) = b_{1_{ij}} z^{-n_{k_{ij}}} + \cdots + b_{n_{b_{ij}}} z^{-n_{k_{ij}} - n_{b_{ij}} + 1} \tag{57}$$

Where δ_{ij} represents the Kronecker symbol.
From the above it is clear that the ARX structure for a given system can be defined by means of the number of poles n_a, the number of zeros n_{b-1} and the number of time delays n_k. The matrices A(q) and B(q) are determined by means of off-line parameter identification methods (Uchida-Frausto et al., 2003).
To achieve a good fit of a greenhouse climatic model, it is necessary to estimate the suitable values for the coefficients implicated in the autoregressive model. That is, for each structure of ARX and ARMAX models we need to obtain the coefficients $a_1,...,a_{n_a}$ and $b_1,...,b_{n_b}$ and the order of the model given by the values the parameters n_a, n_b and nk, based on the information provided by the inputs and outputs in order to get the best fit between the measured values and the estimated values by the model.
To determine the coefficients of ARX and ARMAX models that better fit the simulated to the measured data. A method to make it is to minimize the sum of square errors (J):

$$\begin{aligned} J(p) &= \sum_{i=1}^{N} \left(\bar{y}(t_i, p) - y(t_i) \right)^2 \\ p^* &= \arg\min J(p) \end{aligned} \tag{58}$$

Where $\bar{y}(t_i, p)$ is the simulated output, y in time t_i, $y(t_i)$ is the measurement y in time t_i, N is the number real measurement (time), p is the parameters set (coefficients of the model) and p* are the parameters that reduce J(p) to a minimum.
In the current work, the minimization of Eq. (58) is a non-linear multivariable optimization problem that can be solved by using evolutionary algorithms, such as: GAs and EP since they are global optimization methods. The structure of any EA is the same (Eiben & Smith, 2003), as is shown in Fig. 1. Differences among evolutionary techniques consist of the kind of selection, mutation and crossover operators applied to find the optimum value of the parameters for the optimization function. In this case, the kind of selection, the crossover and the mutation used for each EA is presented in Table 1.

4.4 Gray box model

In a gray box model it is made the integration of a white box model and a black box model. The workout of the gray box model is done with the indirect strategy used in research as Acuña et al. (1999), Dimitri (1992), Thomson and Kramer (1994) y Thornley & Johnson (2000). In figure 3 is presented the indirect training diagram.
The objective of a gray box model training consist of determining the ARX model parameters which approximate the existing and unknown relation between and a priori of some white box model parameters and the relevant resultant variables. Since the ARX

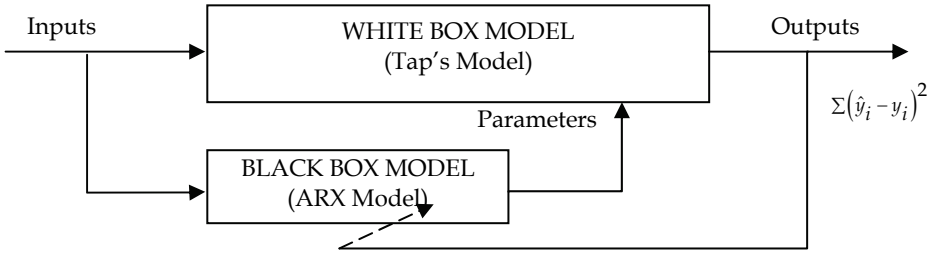


Fig. 3. Indirect training diagram in a gray box model.

model parameters are unknown, the output of the white box model (measurable outputs) is minimized. The objective function is:

$$J(p) = \frac{1}{2}\sum_{i=1}^{N}\left(y(t_i) - y_{cg}(t_i, p)\right)^2$$
$$p^* = \arg\min J(p) \tag{59}$$

Where $y(t_i, p)$ is the simulated output by the gray box model, $y(t_i)$ is the measurement y in time t_i, N is the number real measurement (time), p is the set of parameters (coefficients of the model) and p* are the parameters that reduce J(p) to a minimum.

The objective function gives a consistent real value in the sum of the square error between obtained values by the model and the expected outputs, with which the optimization method determines the ARX model parameters.

In this case, the White box model used is Tap's (2000). However, for the natural ventilation flow through windows Φ_v, Eq. (28), and the saturated steam p_g^*, Eq. (31), and Ti (Hanan, 1998) would be estimated by a black box model, this is, with an ARX model. Due to Eq. (28) and (31) depend on parameters σ, χ, ς, ξ, ψ, a_1, a_2 and a_3 that according to the sensitivity analysis these turn out to be the most sensitive.

5. Results

5.1 White box model

Initial values of model parameters were taken from Tap (2000). Table 2 shows the statistics corresponding to the results of the simulation before model calibration. Figure 4 shows the results obtained before the calibration for the simulation of a week's data.

A local sensitivity analysis was carried out using measured climate data in order to select the most sensitive model parameters to be estimated during the calibration process. As result of this analysis, the parameters related to the opening of the windows and to the evapo-transpiration of the plant were very sensitive. That is, the inside air temperature and the absolute humidity were determined to be most affected by the saturation vapour pressure parameters (a1, a2 and a3) and the ventilation rate parameters (σ, χ, ς, ξ and ψ). The number of parameters could be reduced by considering a smaller sensitivity index. However, in this case, 8 parameters were considered because there was a considerable difference between the effect of these 8 and that of the rest.

Calibration of parameters was made by means of the four evolutionary techniques (global search method): GAs, ES, EP and ED.

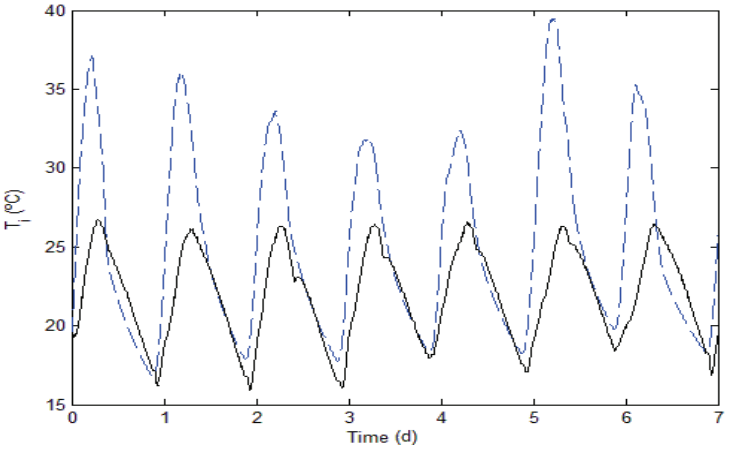


Fig. 4. Indirect Measured (solid line) and simulated (dotted line) air temperature before calibration.

	Method	r	E	%SEP	AVR
Before calibration	Original values	0.6643	-2.3902	24.5262	3.3902
After calibration	GAs	0.6422	-0.1936	14.5527	1.1936
	EP	0.6785	-0.5266	16.4581	1.5266
	ES	0.6398	-1.345	20.3982	2.345
	DE	0.6874	-0.2708	15.0159	1.2708

Table 3. Statistical results of greenhouse air temperature before calibration and after calibration of a white box model using evolutionary algorithms.

The statistics of the results of the simulation using the parameters given by GAs, ES, EP and DE are shown in Table 3. The obtained results of the simulations for a week with the GAs,

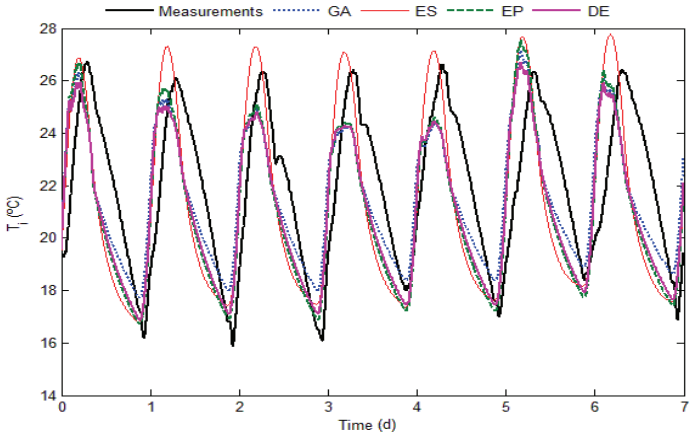


Fig. 5. Measured (solid line) and simulated air temperature after calibration with GAs, ES, EP and DE.

ES, EP and DE parameters are shown in the Figure 5. Results for Evolutionary Algorithms are obtained from 10 runs and the selection was made considering the parameters that minimized the error better between measured and estimated data.
Results show a minor error in the model's predictions for the air temperature within the greenhouse after calibration. Although there is not an increase in the correlation (r), for T_i, for each calibration method results between results obtained before and after calibration with GA and EP, unlike the other two methods, Table 3. Furthermore, it can be observed that the efficiency coefficient (E) is negative in all cases, suggesting that average use values of observed data is better than the estimates obtained. However, percentage standard error of the prediction (% SEP) changed from 24 to 14 (Table 3) for the temperature T_i , when GA are used. Finally, average relative variance (ARV) decreased from 3.39 to 1.19 (Table 3) for the temperature T_i when GA is used. In general terms there is an improvement when GA is used.

5.2 Black box model

An ARX model with structure na = 1 (one output variable), for nb=2 (four input variables) and for nk = 1 was evaluated.

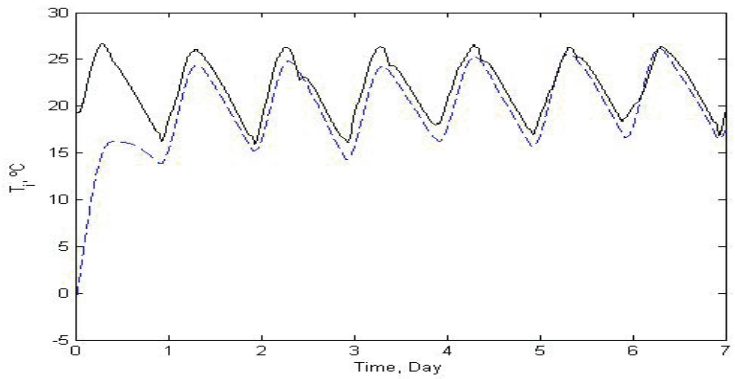


Fig. 6. Measured (solid line) and simulated (dotted line) air temperature by means of an ARX model

Table 4 shows the statistical results obtained when 25% of the data is evaluated to estimate the ARX model. Fig. 6 shows the behavior of the ARX model with structure na=1, nb=[2 2 2 2]. Looking at Table 4 it can be observed that a better fit occurs when the identification parameter is performed with GAs. Correlation r takes the value 0.86 and the efficiency (E) obtains value of 0.0.44, percentage standard error of the prediction (%SEP) is 9.94 and the average relative variance (ARV) is 0.55.

Method	r	E	%SEP	AVR
ARX	0.6620	-0.8402	18.0704	1.8402
GAs	0.8691	0.4429	9.9427	0.5571
EP	-0.0058	-0.5386	16.5233	1.5383
ES	-0.0029	-1.3235	20.3051	2.3235
DE	0.7735	-0.2726	11.3607	0.7274

Table 4. Statistical results of greenhouse air temperature before calibration and after calibration of a black box model using evolutionary algorithms.

According to the results, when the parameter identification of an ARMAX structure is performed by means of GAs there is a better fit of the simulated data to the measured data when t 25% of the data are used to estimate and 75% of the data is used to validate the model.

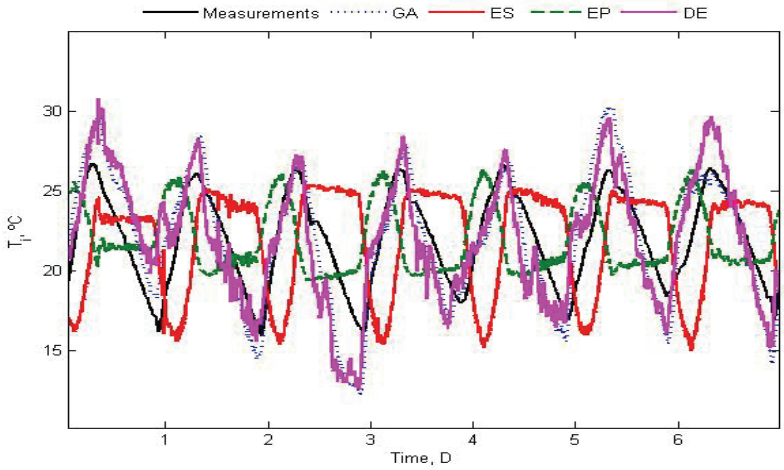


Fig. 7. Measured (solid line) and simulated air temperature using GAs, ES, EP and DE

5.2 Gray box model

In this case the white box model applied was that of Tap (2000), where natural ventilation flow through windows Φ_v and the saturated vapour p_g^* are estimates through an ARX structure. The results of this simulation are shown in Fig. 8 where 50 % of the data was used for estimating the parameters.

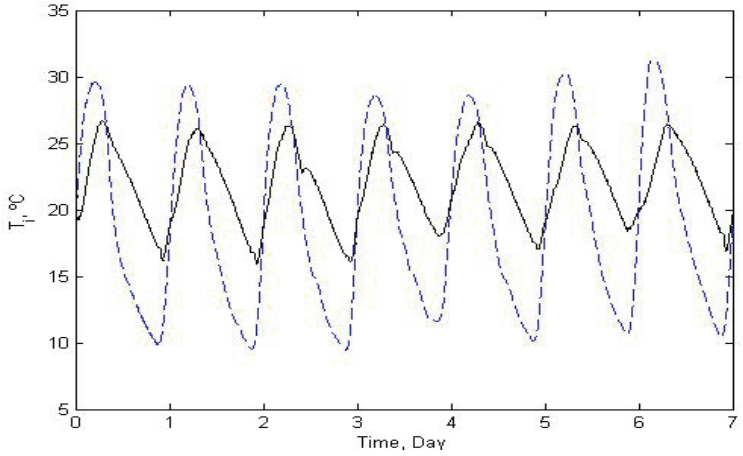


Fig. 8. Measured (solid line) and simulated (dotted line) air temperature by means of a gray box model

The statistical results of the parameter identification through GA, ES, EP, and DE are shown in Table 5. According to these results there is a better fit of the estimated data through the gray box model when parameter identification is done using DE. Fig. 9 shows the results of each of the simulations

Method	r	E	%SEP	AVR
Gray box model	0.7196	-3.1438	27.0180	4.1438
GAs	0.7186	-32.1576	76.4268	33.1576
EP	0.6787	0.3450	10.7419	0.6550
ES	0.671	0.2778	11.3200	0.7222
DE	0.9015	0.7064	7.2173	0.2936

Table 5. Statistical results of greenhouse air temperature before calibration and after calibration of a gray box model using evolutionary algorithms

The obtained results of the simulations for a week with the GAs, ES, EP and DE parameters are shown in the Fig. 5.

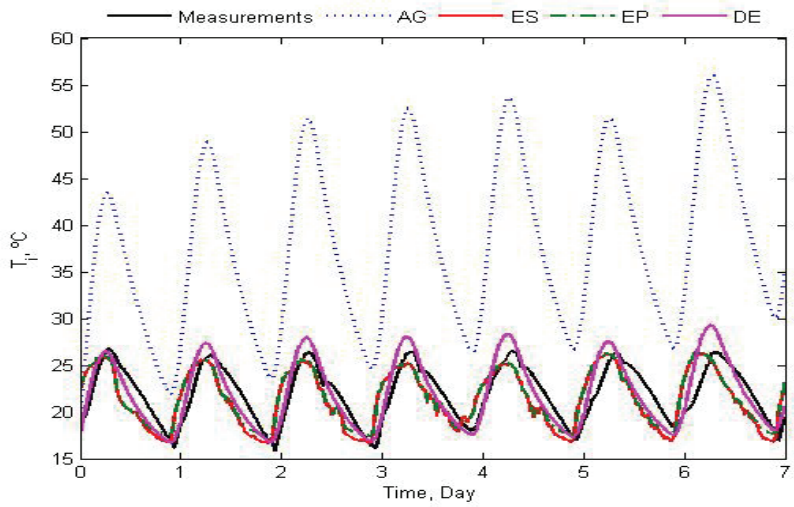


Fig. 9. Measured (solid line) and simulated air temperature using GAs, ES, EP and DE

6. Conclusions

Climate conditions were measured in a greenhouse located in the central region of Mexico. To perform the calibration, 4 global evolutionary algorithms (GAs, ES, EP and DE) were applied and the estimations obtained by the model using the parameter values given with the different methods were compared in order to ascertain which method was more effective.

The climate model gave best predictions for the air temperature within the greenhouse when using the parameter values obtained by means of the GA when a white box model was used.

In the same way, results obtained with GAs show that this method is more effective than the others methods to find parameters for auto regressive models to predict air temperature inside of a greenhouse with structure na = 1, nb = [2 2 2 2], and nk = [1 1 1 1] and the data group 25%:75%. The advantage in this case consists in the use of small sample data (25%) can give a better estimation that the traditional method (square least) used to the parameter identification of an ARX model.

For gray box model, the prediction of air temperature inside of a greenhouse is better when the parameter identification is done by means of the DE method.

To this point, the model can be used to design and development of algorithms of control. Likewise, the model can be integrated with a physiological model to get a production process model of a greenhouse.

7. Acknowledgment

This research was partially supported by CONACyT. Many thanks to Cindy Esparza of the Translation Edition Office of UPSZ for the language revision.

8. References

Acuña, G.; Cubillos, F.; Thibault, J. & Latrille, E. (1999). *Comparison of methods for training grey-box neuronal networks models*. Computers and Chemicals Eng., Supplement, 561-564

Aguado, B. A. & Martínez, I. M. (2003). *Automática y Robótica: Identificación y Control Adaptativo*, Prentice Hall Ed.

Bakker, J. C. (1991). *Analysis of humidity effects on growth and production of glasshouse fruit vegetables*, PhD thesis, Wageningen Agricultural University, The Netherlands

Bot, G. P. A. (1983). *Greenhouse climate: form physical processes to a dynamic model*, PhD thesis, Wageningen Agricultural University, The Netherlands

Bouman, B. A. M.; Van Keulen, H.; Van Laar, H. H. & Rabbinge, R. (1996). *The 'School of De Wit' crop growth simulation models: A pedigree and historical overview*, Agric. Sys., Vol. 52, Nos.2/3, pp. 171-198

CIDEIBER. (October 2005). México, actividades del sector primario, sector agrícola vegetal, Available from Internet: http:// www.cideiber.com/infopaises/Mexico/ Mexico-04-01.html

Coello, C. A. (2007). *Introducción a la computación evolutiva*, Notas de curso, Departamento de computación, CINVESTAV-IPN, México

De Jong, T. (1990). *Natural ventilation of large multi-span greenhouses*, PhD thesis, Wageningen Agricultural University, The Netherlands

Dimitri, C. (1992). *A Hybrid Neural Network-First Principles Approach to Process Modeling*, Universidad de Pennsylvania, AIChE Journal, Vol. 38

Eiben, A. E. & Smith, J. E. (2003). *Introduction to Evolutionary Computing*, Springer, Berlin,

France, J. & Thornley, J. (2006). *Mathematical Models in Agriculture*, CABI

Hanan, J. (1998). GREENHOUSES: Advanced Technology for Protected Horticulture, volumen 1, CRC, first edition

Hernández-Hernández, F. (2009). Simulación del crecimiento y desarrollo del chile pimiento (Capsicum Annuum L.) para mejorar su producción en condiciones controladas, Tesis Maestria, Universidad Autónoma de Querétaro

INU. (2006). *El agua, una responsabilidad compartida*. 2º Informe de las Naciones Unidas sobre el desarrollo de los recursos hídricos en el mundo

Jolliet, O. & Bailey, B. J. (1992). The effect of climate on tomato transpiration in greenhouses: measurements and models comparison, Agricultural and Forest Meteorology, 58:43–62

Linker, R.; Seginer, I. & Buwalda, F. (2004). *Description and calibration of a dynamic model for lettuce grown in a nitrate-limitingenvironment*, Mathematical and Computer Modelling, 40(9–10), 1009–1024

Ljung, L. (1999). System Identification, Theory for the user, Prentice Hall Ed

Ljung, L. (2005). System Identification Toolbox for Use with MATLAB, The Matworks Inc

López-Cruz, I.; Ramirez-Arias, A. & Rojano-Aguilar, A. (2004). Análisis de sensibilidad de un modelo dinámico de crecimiento para lechugas (Lactuca sativa) cultivadas en invernadero, Agrociencia 38 (6): 613-624

López, I. L.; Rojano, A.; Ramírez, A. & Bonilla, M. (2006). *Modelos matemáticos para el crecimiento y desarrollo de cultivos: concepto y metodología*, Notas, Universidad Autónoma de Chapingo

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, second edition

Michalewicz, Z. (1994). *Evolutionary computation techniques for nonlinear programming problems*, International Transactions in Operational Research, 1(2), 223–240

Michalewicz, Z. (1996). Genetic Algorithms + Data Structures = Evolution Programs, Springer, USA

Moreno Quintana, V. & Acuña Leiva, G. (2005). Estimacion de parámetros utilizando modelos de caja gris con redes neuronales de base radial, 13° encuentro chileno de computación

Price, K. V. & Storn, R. M. (2005). Differential Evolution: A practical approach to global optimization, Springer, Germany

Ríos-Moreno, G. J.; Trejo-Perea, M.; Castañeda-Miranda, R.; Hernández-Guzmán, V. M. & Herrera-Ruiz, G. (2006). *Modelling temperature in intelligent buildings by means of autoregressive models*, Automation in Construction, 16(5), 713–722

Saltelli, A.; Chang, K. & Scott, M. (2000). *Sensitivity Analysis*, J. Wiley & Sons, Chichester, England, 475 p

Seginer, I. & Ioslovich, I. (1998). A single state variable model to mimic tomgro for control applications, Acta Horticulturae 456: 93-100

Sjöber, J.; Zhang, Q.; Ljung, L.; Benveniste, A.; Deylon, B.; Glorennec, P.Y.; Hjalmarsson, H. & Juditsky, A. (1995). *Nonlinear black-box modeling in system Identification: a unified overwied*, Automatica, 31:1691-1724

Tap, F. (2000). *Economics-based optimal control of greenhouse tomato crop production*, Phd Thesis, Wageningen Agricultural University, Wageningen The Netherlands, 127 P

Thibault, I.; Acuña, G.; Perez, R.; Jorquera, H.; Molin, E. & Agosin, E. (1999). *A hybrid representation approach for modeling complex dynamic bioprocesses*, Bioprocesses Engineering 22

Thomson, M. & Kramer, M. (1994). Modeling Chemical Processes Using Prior Knowledge and Neural Networks, MIT, Process System Eng., Vol. 40

Thornley, J. H. M. & Johnson, I. R. (2000). *Plant and crop modelling, a mathematical aapproach to plant and crop physiology*, The Blackburn Press, New Jersey, USA., 669 P

Uchida Frausto, H.; Pieters, J. G. & Deltour, J. M. (2003). *Modelling greenhouse temperature by means of auto regressive models*, Biosystems Engineering, 84, 147-157

Udink-Ten-Cate, A. J. (1983). *Modeling and (adaptive) control of greenhouse climates*, PhD thesis, Wageningen Agricultural University, The Netherlands

Van-Henten, E. J. (1994). *Greenhouse climate management: an optimal control approach*, PhD Thesis, Wageningen Agricultural University, The Netherlands

Van Henten, E. J. & Van Straten. (1994). *Sensitivity analysis of a dynamic growth model of lettuce*, J. Agric. Engng. Res., 59, 19-31

Van-Henten, E. J. (2003). Sensitivity Analysis of an Optimal Control Problem in Greenhouse Climate Management, Biosystems Engineering, 85:335-364

Stages of Gene Regulatory Network Inference: the Evolutionary Algorithm Role

Alina Sîrbu, Heather J. Ruskin and Martin Crane
Centre for Scientific Computing and Complex Systems Modelling
School of Computing, Dublin City University, Dublin 9
Ireland

1. Introduction

Uncovering interactions between genes and their products has been a major aim of Systems Biology over recent years, (Przytycka et al. (2010) and references therein). The objective is to gain a better understanding of the functioning of different organisms, together with discovery of disease markers and new treatments, (Bar-Joseph, 2004; Tan et al., 2008). Gene regulatory network (GRN) analysis has been facilitated by the advent of technologies for measuring gene expression: these include mature technologies such as *qrtPCR*, (Logan et al., 2007), suitable for a limited number of genes, and *microarrays*, (e.g. Baldi & Hatfield (2002)), which allow for high-throughput measurement of thousands of genes at the same time. More recently, *RNA-Seq* (Hurd & Nelson, 2009) measurements have become available, due to advances in high throughput *sequencing* technology, but these data are still scarce, due to high experimental costs. Characterised as they are by high dimensionality and noise levels, analysis of these data is far from trivial. The class of computational methods known as Evolutionary Algorithms, (EAs), has demonstrated relevance for different investigative targets, (Pal et al., 2006; Sîrbu et al., 2010a). This chapter, in consequence, presents an overview of approaches and issues in GRN modelling and inference, and discusses the role of EAs in this regard.

Three different analysis stages can be identified for GRN inference: (i) expression pattern analysis, (ii) mathematical modelling from expression data and (iii) integrative modelling. At each of these, and most particularly at the last stage, EAs have an important role to play, due to the strength and flexibility of these search methods.

Expression pattern analysis is largely concerned with the application of classification and clustering methods to gene expression data. Clustering of genes, as a first step towards GRN modelling, (Lee & Yang, 2008; Thieffry, 1999), together with classification, which aims at assigning samples to different classes (usually for gene expression data, to distinguish between tissue types, e.g. control/treatment or healthy/infected, for diagnostic purposes), give valuable insight on gene involvement in different processes. EAs are typically employed at this stage, with some success, for feature selection and clustering.

At the second stage, a GRN model is created to explain the data (see e.g. He et al. (2009) for a review), which can be used for *in silico* simulation and process analysis under various criteria. Such a model is built by reverse engineering from available time course expression data, with inferential algorithms used to fit model parameters to the data, using evolutionary optimisation. Different EA approaches are presented here, for inferences on *discrete qualitative*

to *continuous quantitative* models. Consequently, the discussion includes *classical* to *hybrid* EAs, and identification of strengths and weaknesses.
A general limitation in GRN modelling is that, although qualitative models can be built for entire GRNs, quantitative analysis is still restricted to sub-networks, due to limited data available and the large number of parameters to be optimised. Quantitative models allow for a better representation of interaction links, and for continuous simulation of dynamical behaviour, but the limitations in size and accuracy have impeded their use in real-world scenarios. Consequently, a third stage in network inference, integrative analysis, (Hecker et al., 2009), aims at reconciling different sources for the large amount of biological data available, in order to improve reliability of the inferential process, and realism of the models. This is not without risk, as multi-source data can contain heterogeneous noise, which has to be dealt with. Further, large scale integrative analysis requires a large amount of computational resources, and algorithms have to be optimised and parallelised to address this. Additional data types, which can contribute to this synthesis, include DNA-protein interactions, knock-out/knockdown experiments, binding site affinities, as well as known transcription factors (TFs) and RNA interference measures.
To date, integration efforts are sparse. Nevertheless, examples of approaches based on EAs are presented here, although these typically combine only *one* additional data type with expression measurements. Ideally, all related data should contribute to the inferential process. With this aim, a novel algorithm, based on evolutionary computation, that aims at large scale data integration for quantitative modelling, is also outlined, and the advantages and disadvantages of EAs for data unification discussed.
The rest of this chapter gives background on GRNs, general modelling methodology and mathematical models in Section 2, then follows the development of existing evolutionary algorithm approaches through the three stages of inference in Sections 3, 4 and 5. Section 5 also describes a novel approach for data integration, which builds a framework for inclusion of multiple data types in the inferential process, followed by a concluding discussion on EA role in Section 6.

2. Background

2.1 Gene regulatory networks (GRNs)

DNA encodes the information the cell needs to create proteins that are vital for its mechanisms (e.g. Brown (2002)). Each cell of an organism contains the same information, i.e. the DNA sequence, but in different tissues, cells will behave differently. This indicates that other mechanisms must exist to control protein levels depending on the environment; one such example is the gene regulatory network.
The Central Dogma of Molecular Biology describes gene expression[1] as $DNA \rightarrow RNA \rightarrow$ protein, but, in fact, the process is more complex as it consists of several stages and can be influenced by several factors at each stage, (Brown, 2002). The *initiation of transcription* is one of these stages, influenced by proteins called transcription factors (TFs). These TFs bind to the region upstream of the gene that needs to be expressed and regulate its transcription in a positive or negative way, (up- or down-regulation). Such interactions create a regulatory network between TFs and genes, i.e. a GRN. As TFs are in turn encoded by other genes, we can consider these interactions as being between pairs of genes instead of gene-protein pairs. GRNs are used to control protein levels during biological processes, and perturbations in the network lead to unwanted behaviour, i.e. disease.

[1] Formation of a protein from the corresponding gene.

GRNs feature a set of characteristics that distinguish them from random networks (Marbach et al., 2009). They are scale-free, modular and contain *network motifs*, i.e. patterns of interactions that appear with high frequency and control oscillatory behaviour of the network. Also, the in-degree of the nodes, (i.e. the number of regulators for each gene), is bounded by a small number compared to the total number of genes in the network. These properties are very important, as they can be used to enhance the inferential process and also to generate plausible *synthetic networks* which are used to validate inferential algorithms.
Several technologies that measure gene expression have been developed, typically concerned with gene activity at the mRNA[2] level. Among these, high-throughput technologies, (microarrays, Baldi & Hatfield (2002), and RNA-Seq, Hurd & Nelson (2009)), allow for measurement of mRNA concentrations for a large number of genes at the same time. These measurements can be viewed as snapshots of the expression levels of genes under certain conditions and, with a large-enough set of snapshots, it is theoretically possible to uncover the underlying GRN (Liang et al., 1998).

2.2 From gene expression data to GRNs

Gene expression data consists of the expression levels of many genes under multiple conditions, (Stekel, 2003). Hence, for each gene, a vector of values shows the *gene expression pattern* for a number of different experiments, (Figure 1a). At the same time, the data can be viewed as a set of vectors describing the behaviour of the organism under certain conditions, (experiments), i.e. the *experimental patterns*, which represent expression values for many genes in a single experiment, (Figure 1b). By analysing both pattern types, (separately or together), useful knowledge related to the connections between genes or the similarity between conditions can be found. The ultimate aim is to build a reliable model for the underlying GRN, which can be further used for *in-silico* simulation and analysis of the system. This goal has triggered significant research efforts, (e.g. He et al. (2009) and references therein), which can be classified, based on the type of analysis performed, specifically: (i) expression pattern analysis, (ii) modelling from time series data and (iii) integrative modelling. Evolutionary algorithms have had an important role in the different stages of analysis, due to their flexibility and search power.
The *first stage* in studying gene expression data for GRN discovery applies pattern analysis algorithms on both experimental and gene patterns seen in the data. Such algorithms include clustering, classification and feature selection techniques.
Clustering is considered here as unsupervised [3] learning where a set of data entries has to be grouped into clusters, based on their attribute values, (Manning & Schütze, 1999). The clusters and cluster assignment for the training data set are typically not known beforehand, but are deduced based on dissimilarity or distance measures. Such measures may be statistical constructs, such as correlation, as well as standard spatial distance measures: Euclidean, Manhattan and others. Bi-clustering is also a variant that has been widely applied to gene expression data, (Kerr et al., 2008). This aims at grouping both genes and experiments at the same time, indicating not only clusters of co-expressed genes, but also in which experiments these appear.

[2] Messenger RNA (mRNA) is the RNA that results from transcription of DNA during the expression of a gene. mRNAs are translated into proteins based on their nucleotide sequence. Other different types of RNA exist, but relate to other functions in the organism.
[3] Recently, supervised clustering methods have emerged from the need for more control over the meaning of the resulting clusters or the features that are considered by the unsupervised clustering technique. However, this section concentrates on unsupervised clustering.

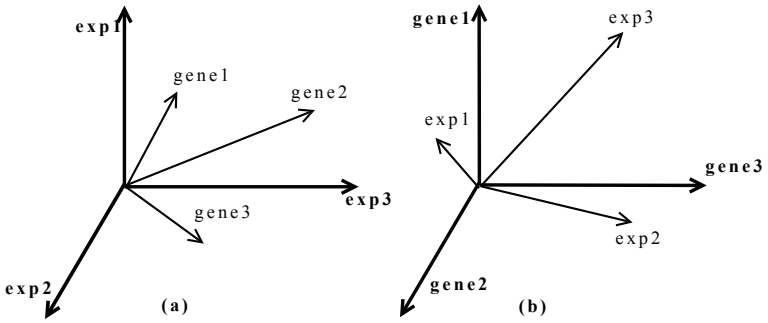


Fig. 1. Gene expression data interpretations: (a) a set of vectors representing expression values for one gene under different experiments and (b) a set of vectors representing expression values for multiple genes under a single experiment

In the case of gene expression data, clustering gives insight on the relationships between genes and, in consequence, is considered the *first step towards gene network inference* (Lee & Yang, 2008; Thieffry, 1999). Genes that belong to the same cluster are assumed to be co-regulated, (i.e. regulated by the same protein complex), or co-regulating, (i.e. regulating each other). Once the clusters are generated, the objective is to look for binding site motifs in the precursors[4] of the genes in each cluster. In this way it is possible to find unknown binding sites or to generate hypotheses on which proteins regulate the co-regulated genes in the cluster, (based on previous knowledge on regulatory motifs). These hypotheses can be further validated by laboratory experiment.

A gene expression dataset can contain thousands of genes so that the elements to be clustered/classified are points in a high-dimensional space, hence analysis is computationally intensive. Also, as data are intrinsically noisy, the high number of dimensions can bias the algorithm convergence. It is possible that some features of the gene expression data are redundant, (Liu et al., 2002), hence the need to develop feature selection techniques, in order to make analysis more efficient. Subsection 3.2 describes some of the feature selection methods, which have been applied in the context of clustering or classification of microarray data. Although classification of different experiments, (for diagnostic purposes, such as distinguishing between infected and healthy tissue), may not have an immediate use in GRN inference, feature selection techniques do give an indication as to which genes are most important in the processes under analysis, so we have included them at the first inferential stage.

A *second stage* in GRN inference is mathematical modelling using time series gene expression data. In these data, gene expression levels are measured over time, with each experiment in the data describing a different time point. These series patterns can be modelled using mathematical tools, of which a large number have been applied to GRNs, (see e.g. He et al. (2009); Lee & Tzou (2009) and references therein). Generally, the process of modelling GRNs consists of a few main steps: choosing an appropriate model, inferring parameters from data, validating the model and conducting simulations of the GRN to predict its behaviour under different conditions. Due to the large number of genes in such datasets, clustering methods (stage one) have been applied by some authors for dimensionality reduction (either by considering cluster centroids as being one gene in the network, Wahde & Hertz (2000), or by analysing subsets of genes corresponding to selected clusters, Lee & Yang (2008)).

[4] The region in the DNA sequence located before the gene.

In order to model a GRN, genes are taken to be variables that change their (expression) values over time. Depending on variable type, methods can be classified as discrete or continuous, deterministic or stochastic, or as *hybrid* (using more than one type of variable). Two broad approaches are described in the literature (Lee & Tzou (2009)): coarse-grained and fine-grained models, with the former containing less detail on the interactions between genes. Usually, coarse-grained models use discrete variables, while fine grained models use continuous ones. A GRN can be very large and can contain complicated interactions, so a fine-grained model will have an enormous number of parameters to deal with. Analysis of this kind of model is very complex, so viewing the network 'top-down', in order to be able to analyse it globally, is the aim of coarse-grained models, (e.g. Linden & Bhaya (2007); Maki et al. (2001); Repsilber et al. (2002)). Other authors, (e.g. Kikuchi et al. (2003); Morishita et al. (2003); Noman & Iba (2006); Tominaga et al. (1999); Wahde & Hertz (2000); Xu et al. (2007)), have chosen to focus on detailed models, but for analysis of sub-networks only of the entire GRN. A useful approach, clearly, is to combine the two levels of detail, moving between the coarse and fine-grained model to highlight key biological knowledge (Maki et al., 2001).
To reverse engineer GRNs, EAs require a specified model type and data set. This enables parameter evolution to be monitored and performance in terms of fitting input data to be evaluated. A population of such parameters representing different models, (also known as population of *candidate solutions* or *individuals*), evolves towards a better set, by applying genetic operators (e.g. crossover and mutation). The fitness function is typically defined as the difference between the observed data and the output of the model, (squared, or averaged over the data points), as described in Equation 1.

$$\text{fitness} = \sum_{i=1}^{n} \sum_{t=1}^{T} (x_i(t) - y_i(t))^2 \tag{1}$$

where $x_i(t)$ is the expression value of gene i at time t, observed in real experiments, and $y_i(t)$ is the expression value of gene i at time t generated by the model. Since every model has its distinctive features, steps in the algorithm differ from one approach to another, but the main skeleton is usually preserved. In this chapter we describe several such methods, following the development from classical to advanced hybrid methods.
The ideal model for a GRN would be fine-grained, accounting for all the features of the real GRN, applied to the entire genome in a cell. Achieving such a model is a non-trivial task, as most methods to date are either too coarse or can not model large systems. Also, existing gene expression time-series data are insufficient to infer the large number of parameters for such a detailed model, resulting in an *under-determined* problem. However, a very large pool of biological knowledge, from different types of experiments, does exist in the literature. The issue then is whether it is possible to combine all existing knowledge in an attempt to improve system inference. Approaches that use gene promoter data, results from protein-protein interaction experiments, knock-out microarray experiments and other related information have started to appear, (Hecker et al. (2009) and references therein), opening the road for a *third stage* of GRN inference, based on *heterogeneous data integration*. Some efforts have used evolutionary computation but are at an early stage only, so that they benefit only partially, at best, from the flexibility offered by EAs for large scale data integration. We aim to address this question also, (Section 5).
In the rest of this section, an outline of mathematical models, used in conjunction with evolutionary algorithms for GRN modelling, is provided.

2.2.1 GRN modelling approaches

Boolean networks

Boolean networks are coarse-grained models for GRNs that use Boolean values for gene expression: the gene is on/off with values 1/0 respectively (Liang et al. (1998)). Regulation is expressed in terms of Boolean functions attached to each gene :

$$Y_i = F_i(X_{i_1}, .., X_{i_k})$$

where $X_{i_1}, .., X_{i_k}$ are the binary expression levels of regulators of gene i and Y_i is the predicted expression value for gene i. This model is very well suited to modelling large networks, as it does not require a large number of parameters. Due to their relative simplicity and limited detail, Boolean networks have been employed in the analysis of steady states and general behaviour of GRNs. However, they have a few disadvantages as they can not simulate continuous behaviour and complex nonlinear interactions, characteristic of GRNs. Additionally, discretisation of expression values, which are continuous, may lead to information loss, which can result in fewer interactions identified.
A generalisation of the Boolean network is the multistate discrete network (Repsilber et al., 2002). In this model, gene expression levels can take more than two discrete values (in the set $S = \{0, .., n\}$) and the transition functions are general functions $F_i : \{0, .., n\}^k \rightarrow \{0, .., n\}$, mapping between current expression values for all genes and that of gene i at the next time point.

Rule sets

Another model of regulation uses different types of rules to explain the observed patterns in the data. This approach has the advantage of being more intuitive, as relationships between genes are expressed using natural language. One such model uses fuzzy rules, (Linden & Bhaya, 2007), which are based on the notion of fuzzy sets. These sets have imprecise boundaries, defined by a membership function: applied to any element in the universe, they return a number in the interval [0,1], representing the degree to which that element is a member of the current set. A fuzzy rule is a conditional of the form *if x is in A then y is in B*, which specifies a relation between fuzzy sets A and B. Every fuzzy rule also has a membership function that specifies the degree of truth of the implication.

Ordinary differential equations

The models described in previous paragraphs are coarse-grained models. These use discrete states for gene expression values, with the influences of a given set of genes on other genes described qualitatively, rather than quantitatively. However, gene interactions are very complex and, in order to model these, a fine-grained continuous model is needed, which considers interactions quantitatively. One such model is a system of differential equations.
Ordinary differential equation systems express the change in the expression level of each gene in time as a function of the expression levels of other genes, but make no other assumption about the mathematical form:

$$\frac{dx_i}{dt} = F_i(x_1, .., x_n) \tag{2}$$

where x_i represents the expression level of gene i. The inferential algorithm, therefore, is not restricted to a prescribed set of functions and can model complex behaviour. At the same time, few constraints mean that the search space is very large and more sophisticated methods are typically required to refine the analysis.

Linear differential equations

The simplest model example, and one that has received a lot of attention (e.g. (Akutsu et al., 2000; Ando & Iba, 2003; Deng et al., 2005)), is the linear system of differential equations. This simplifies the way genes interact, by using linear dependencies but at the same time retains the continuous aspects inherent in differential equations. This simplification results in loss of modelling power, compared to a nonlinear model choice, (as gene interactions are known to be more complex), but gains from the perspective of simpler inference.
This model describes changes in gene expression values as:

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} w_{ij} x_j \tag{3}$$

where x_i and x_j represent expression values of genes i and j and w_{ij} the regulation *strength* of gene j on gene i. A negative value for w_{ij} corresponds to repression of gene i by gene j, a positive value corresponds to activation and a null value to no effect of gene j on gene i. Different versions of the model exist, which add other terms to the equation, accounting for external stimuli, degradation rates or noise. The system can be described by the matrix $\mathbf{W} = (w_{ij})$, also known as the *interaction or regulation matrix*. Inferring a model means finding the values in $\mathbf{W}$.

S-Systems

Although linear systems improve the level of detail achieved by the modelling approach, these still exclude some information. Regulatory networks are intrinsically *nonlinear* systems and approaches that correctly model gene interactions are needed. S-Systems are a special type of differential equation systems, based on power-law formalism, and are capable of capturing complex dynamics. The disadvantages are an increase in the number of parameters and reduction in the available choices of reverse engineering techniques, as linear regression methods are not applicable any more. The equations in S-Systems are of the form:

$$\frac{dx_i}{dt} = \alpha_i \prod_{j=1}^{n} x_j^{g_{ij}} - \beta_i \prod_{j=1}^{n} x_j^{h_{ij}} \tag{4}$$

The two terms correspond, respectively, to synthesis and degradation influence from other genes in the network; specifically, α_i and β_i, are *rate constants* and represent basal synthesis and degradation rate, while g_{ij} and h_{ij}, (*kinetic orders*), indicate the influence of gene j on the synthesis and degradation of the product of gene i.

Linear time-variant model

The linear time-variant model expresses the regulatory effect on a gene using a linear expression:

$$r_i(t) = \sum_{j=1}^{n} w_{ij}(t) x_j \tag{5}$$

and computes the expression value of that gene at time $t+1$ by applying a sigmoid function [5] to this effect:

$$x_i(t+1) = \frac{1}{1 + \exp(-r_i(t))} \tag{6}$$

[5] Mathematical function that has an S-shaped graphical representation. One example of a sigmoid function is the logistic function that restricts the output to the interval$(0,1)$: $f(x) = \frac{1}{1+e^{-x}}$

The values $w_{ij}(t)$ represent the interactions between genes, and are expressed using a Fourier series, as described in Equation 7.

$$w_{ij}(t) = \alpha_{ij} sin(\omega_i t + \phi_{ij}) + \beta_{ij} \tag{7}$$

Thus, interactions are modelled by a linear term, β_{ij} and a non-linear sinusoidal term.

Partial differential equations

The differential equations models, presented so far, do not take into account the spatial distribution of cells and gene products. However, in certain situations, such as cell differentiation during development, the spatial information is very important, so more complex differential equation-based models are needed (partial differential equation systems, introduced by Baldi & Hatfield (2002)). These express concentration changes in both space and time, by reaction-diffusion equations. Here, the one-dimensional version of these equations is described, but these can be extended to 2 or 3-D situations. Considering a linear sequence of L compartments or cells, the concentration of product i in cell l depends on the regulatory effects in cell l but also on the diffusion process between this cell and its neighbours. Diffusion is considered to be proportional to the concentration difference between the two cells. So, the differential equation that describes this process is:

$$\frac{dX_i^{(l)}}{dt} = F_i(X_1^{(l)}, .., X_n^{(l)}) + D_i(X_i^{(l-1)} - 2X_i^{(l)} + X_i^{(l+1)}) \tag{8}$$

where F_i are the regulation functions and D_i are diffusion functions. This is for the case when space is discrete (well delimited cells and compartments). In the continuous case, the concentrations of the products are functions of both time and space, so the system can be modelled with equations of the form

$$\frac{\partial X_i}{\partial t} = F_i(X_1, .., X_n) + D_i(\frac{\partial^2 X_i}{\partial s^2}) \tag{9}$$

where s is the space variable.

Artificial neural network models

ANNs are very suited for modelling complex behaviour as, any function can be simulated, by adjusting the weights. A type of ANN that has been repeatedly used to model GRNs is the recurrent neural network (RNN) (Lee & Yang, 2008; Vohradsky, 2001; Wahde & Hertz, 2000), which model dependencies between genes as:

$$\frac{dx_i}{dt} = m_i S(\sum_{j=1}^{n} w_{ij} x_j + b_i) - r_i x_i \tag{10}$$

where r_i is the degradation rate of gene product i, b_i accounts for external input, m_i is the maximum expression rate and x_j are expression levels while S is a sigmoid function. This model is similar to that of linear systems of differential equations; however the introduction of the sigmoid function allows for modelling non-linear behaviour. A variant of this model considers discrete time points, and computes the expression value of gene i at time point $t+1$ using the values of the regulators at time t:

$$x_i(t) = S(\sum_{j=1}^{n} w_{ij} x_j + b_i - r_i x_i) \tag{11}$$

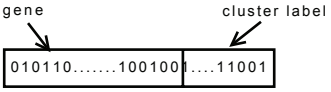


Fig. 2. Chromosome representation in GenClust

This reduces the computational cost for simulating the data, as no differential equations are involved, which is an important advantage in the context of evolutionary optimisation, (which requires simulation for every fitness evaluation).

3. Stage 1 : Pattern recognition methods

3.1 Clustering

Evolutionary algorithms have been applied in clustering gene expression data, either individually, (Di Gesu et al., 2005), or by hybridisation with classical clustering methods (Lu et al., 2004a;b). Additionally, evolutionary bi-clustering methods have been developed and applied to this type of data (Chakraborty & Maka, 2005; Mitra & Banka, 2006).

GenClust (Di Gesu et al., 2005) is a novel method, using a genetic algorithm-like approach. It differs from other genetic algorithms in that the population does not represent a set of possible solutions, but only one. Each individual in the population encodes one sample and a label representing its cluster (Figure 2). By analyzing these labels the components of each cluster can be computed. The approach incorporates elements of EC, such as genetic operators, that are applied at each generation. To generate a secondary population, classical one-point crossover and bit-flip mutation are applied to each individual with given probability. After applying the operators, redundancy and inconsistency have to be removed from the population, so that it still represents a partition of the data set. Fitness evaluation is based on the sum of intra-cluster variances. The aim of the algorithm is to minimise this measure and, consequently, to obtain tight clusters. The method has been validated on five datasets (Rat Nervous System, Reduced Yeast Cell Cycle, Yeast Cell Cycle, Peripheral Blood Monocytes and Reduced Peripheral Blood Monocytes) and compared with other clustering methods like K-Means. The algorithm has been shown to converge rapidly to a local minimum; however, the resulting clusters were comparable those obtained by other techniques.

A similar objective was pursued by Lu et al. (2004a;b), where a hybrid genetic K-Means algorithm (GKA) with two different versions (FGKA - fast GKA and IGKA - iterative GKA) was introduced. Hybridisation with K-Means consists of a custom genetic operator, based on this classical clustering method, which changes cluster allocation to the closest centroid in random individuals. This operator is applied to each individual with given probability. Neither FGKA nor IGKA uses a crossover operator, and mutation is performed based on dynamically computed probabilities depending on current cluster assignment. The difference between the two algorithms is that the latter updates cluster centroids and within cluster variance each time a mutation is performed on an individual, while the former computes these for each generation. This makes IGKA faster when mutation probabilities are low, while FGKA is faster when these are large. In consequence, a hybridisation of the two (HGKA-hybrid GKA) is also proposed (Lu et al., 2004b). The algorithms were applied to microarray yeast and serum data and IGKA was shown to obtain better clusters of genes from the same functional categories.

In Chakraborty & Maka (2005) a genetic bi-clustering algorithm based on K-Means and greedy local search seeding is presented. The algorithm was applied to yeast and human lymphoma data and was shown to provide better bi-clusters when validated against previous biological

knowledge, compared to Cheng & Church (2000) (which adopts a greedy search approach). A similar algorithm, that of Mitra & Banka (2006), employs multi-objective optimisation for bi-clustering. The algorithm is initialised using a greedy algorithm based on random initial solutions. Two objective functions are used, one maximising the number of genes and conditions in the bi-cluster, and another maximising homogeneity. Method evaluation was performed on the same yeast and human lymphoma datasets, and results indicate better performance compared to the single objective variant and to simulated annealing for bi-clustering Bryan (2005).

3.2 Feature selection techniques

Given the high dimensionality of gene expression data, clustering and classification are computationally expensive. However, a large fraction of the genes in these datasets are not differentially expressed between different experiments, (i.e. are redundant), so these could be eliminated from the analysis to reduce dimensionality. To achieve this, feature selection techniques have been formerly applied to gene expression data, mainly for classification purposes. Such methods select features (genes) that are important in the process under analysis, as they display a change in expression from one experiment to another. This filtering not only reduces the computational cost, but also improves pattern recognition for participating elements.

Feature selection methods can be classified into two categories: wrapper and filter methods. *Filter* methods compute for each feature a measure of relevance for the current classification task. The features are sorted by their relevance and the top n are further used for pattern recognition. *Wrapper* methods, on the other hand, use the classifier itself to find the importance of a set of genes. They select a feature subset and train a chosen method on that set. The performance of the trained classifier can be seen as a measure of the relevance of the genes in the subset. The wrapper method iterates this operation for different subsets and chooses the best one, and the difficulty is how to choose feature subsets that maximise the accuracy of the classifier, while minimising the number of selected genes and iterations. The search space for this problem is huge: if the number of initial genes is n, 2^n possible subsets exist. In this context, evolutionary techniques are known to cope well, as they benefit from mechanisms obtaining good solutions by searching a small portion only of the entire space, (Baeck et al., 2000). Consequently, there are several approaches that use EAs as wrapper methods for feature selection, for example Li (2001); Li et al. (2004); Ooi & Tan (2003); Shah & Kusiak (2004); Souza & Carvalho (2005). One of these has also been applied to proteomics data, (Li, 2001; Li et al., 2004).

Most evolutionary approaches for wrapper methods are very similar. A population of gene subsets is maintained and allowed to evolve using different genetic operators. The fitness of each candidate solution is a measure based on the training error of the classifier when using that specific set of features. After applying genetic operators, the fittest individuals remain in the next generation. While principles are the same, existing methods differ in terms of classifier used or EA components, (e.g. size of the chromosomes, fitness function, etc). Additionally, some methods ,(Li, 2001; Liu et al., 2009; Shah & Kusiak, 2004), aggregate features obtained in multiple runs in order to improve performance. Table 1 summarises existing EA wrapper methods.

Recently, a new method for feature selection using genetic algorithms has been developed (Zhu et al., 2007). This is a hybrid of the wrapper and filter methods: two operators that add or remove features from a set, in a filter-like manner, are applied to the feature set encoded by the best individual of each generation. In this way, the individuals of the genetic algorithm, (candidate feature sets), are fine tuned to improve the overall fitness and reduce the number of

Method	EA	Classi-fier	Multi-class	Feature set size	Combining results	Fitness	Datasets
Li (2001); Li et al. (2004)	GA	K-NN	No	Fixed	Filter by appearance count	Classifier accuracy	Leukemia (microarray), Ovarian cancer (SELDI-TOF)
Shah & Kusiak (2004)	GA	Decision tree	No	Fixed	Reunion or intersection	Classifier accuracy	Emulated
Ooi & Tan (2003)	GA	Bayesian	Yes	Variable	None	Classifier error rate in cross-validation and independent test	9 cancer types, 14 cancer types
Souza & Carvalho (2005)	GA	SVM	Yes	Variable	None	Classifier error rate and feature set size	Leukemia, Blue-cell tumour
Liu et al. (2009)	GA	ICA-SVM, P-ICR	No	Variable	Intersection	LOOCV + feature set size	Colon cancer, High-grade glioma

Table 1. Features of EA wrapper methods. Abbreviations: ICA - Independent Component Analysis, (Liu et al., 2009), GA- genetic algorithm, SVM - support vector machine, K-NN - K - nearest neighbours, LOOCV - leave one out cross validation

generations. The algorithm uses an SVM as a classifier. The two newly introduced operators are based on the *Markov Blanket* concept. The Markov Blanket of a feature F is the set M of features that satisfies :

$$P(\mathcal{F} - M - F|F, M) = P(\mathcal{F} - M - F|M), \tag{12}$$

where $\mathcal{F}$ is the set of all possible features. So, the probability of the values for all features except F and M are independent of F, given M. Intuitively, in our case, if the Markov Blanket of feature F is in a subset of features, then it can bring no more information to that subset so it can be removed. The algorithm was applied by the authors on 11 different datasets, including ones for lung or breast cancer, and compared to other filter and wrapper feature selection methods. It was shown to perform better than other methods for most datasets. In Zhu & Ong (2007), also, another hybrid filter-wrapper genetic algorithm-based feature selection algorithm is presented, which implements the new genetic operators using a ranking method, i.e. Robnik-Łikonja & Kononenko (2003), instead of the Markov Blanket. This is shown to have similar results in terms of accuracy. However, the Zhu et al. (2007) algorithm finds smaller gene sets, so it has an important advantage in terms of practical use: the smaller the number of features, the less expensive the diagnostic procedure. Further, the Markov Blanket embedded genetic algorithm has also been applied very recently to multi-class problems, (Zhu, Jia & Ji, 2010; Zhu, Ong & Zurada, 2010), using multi-objective optimisation, (where each objective corresponds to the accuracy of a bi-classification task in a one-versus-all manner). Here, the notions of *full class relevant* and *partial class relevant* features are introduced. These, respectively, are features that have a role in differentiating all classes (i.e. display different expression levels in all classes) and features that differentiate only part of the classes (i.e. may

have similar values is some classes). The algorithm identifies both types of features and is shown to perform better than Zhu et al. (2007) on synthetic and microarray gene expression data.

4. Stage 2: Modelling GRNs from time series data

An overview of existing EAs for GRN model inference from time series data is presented in this section. The discussion considers methods applied to both discrete and continuous models. Due to added complexity of the latter models, many EA approaches have been developed, from classical to advanced algorithms, and these are outlined, indicating their gradual development and their role in GRN inference.

4.1 Discrete models

Although applied more extensively for continuous models, evolutionary algorithms have been used for qualitative model analysis also. Linden & Bhaya (2007) introduced a method of inferring fuzzy rules from microarray data, using genetic programming. The algorithm uses the reverse Polish notation[6] for rules, which can be easily represented as trees, with three Boolean operators for the conditions: NOT, AND and OR. A population of this type was evolved using classical genetic operators on trees and the best individuals were selected to progress to the next generation. Fitness was defined as the percentage error observed between real data and the data generated by the rules. The algorithm was applied to finding rules in microarray data from experiments on the response to cold of the plant *Arabidopsis Thaliana*, as well as on the rat nervous system. A clustering algorithm was applied initially to reduce dimensionality, with resulting clusters considered to form one node in the network. Results were validated based on previous knowledge of the datasets, while new hypotheses for subsequent laboratory experimentation were proposed.

In Repsilber et al. (2002) a genetic algorithm was used to fit a multistate discrete network, (Section 2.2.1), to simulated gene expression data. The aim was to rank previously known hypotheses about the structure of the network, by allowing model parameters to evolve. The approach also introduced time delays ($\delta = \{\delta_1, .., \delta_N\}$) to model the *time gap* between the transcription of one gene and the regulation effect of the resulting protein. This time gap is the time needed for the mRNA to be translated into a protein, so a node changes its state only after initiation of transcription *plus a time delay*. The algorithm thus searches for the most probable model structure for the data available.

Another method of inferring discrete GRN models, based on genetic programming, was developed by Eriksson & Olsson (2004). Here, genes take Boolean values and the regulatory network structure for each target gene is encoded as a tree and evolved to obtain better structure. For each such tree in the population, a Boolean function is determined from data, (by computing the truth table using expression levels in the data), and fitness is assigned based on ambiguities that arise. This results in choosing those structures that have fewer ambiguities, so indicate more plausible interactions. The method was tested on synthetic networks of different size, (10 to 160 genes), and shown, for networks smaller than 40 genes to successfully locate structures with over 75% of the optimal fitness, (with a value of 51% for the 160-gene network). However, to date, this method has not been validated with real data.

A similar evolutionary algorithm optimising the *wiring of a Boolean network* is described in Esmaeili & Jacob (2009). This method starts with randomly generated wirings with a limited number of regulators for each gene and evolves these structures using differential evolution.

[6] In the reverse Polish notation, the symbol order in an expression is changed: the operators are in front of the operands. For instance, $a \times (b + c) - d$ becomes $- \times a + bcd$.

Method	EA	Mo-del	Fitness	Local search	Sta-ges	Datasets (Size)
Ando et al. (2002); Iba (2008); Sakamoto & Iba (2001)	GP	ODE	Error + degree penalty	LMS	-	Synthetic (5)
Fomekong-Nanfack et al. (2007)	ES	PDE	Error	-	-	Fly (6)
Ando & Iba (2003)	GA	LDE	Error	-	√	E. coli (9), Yeast (8)
Deng et al. (2005)	GA	LDE	1 + Error	-	√	Rat 20)
Tominaga et al. (1999)	GA	SS	Error	-	-	Synthetic (2)
Iba & Mimura (2002)	GA	SS	Error	-	√	Synthetic (10)
Kikuchi et al. (2003)	GA	SS	Error + parameter penalty	Simplex Crossover	√	Synthetic (5)
Kimura et al. (2003)	GA	SS	Error + parameter penalty	QP	√	Synthetic (30)
Noman & Iba (2005)	DE	SS	Error + parameter penalty	-	√	Synthetic (5)
Noman & Iba (2006; 2007)	DE	SS	Error + parameter penalty	HC	√	Synthetic (20), Yeast (14-qualitative)

Table 2. Evolutionary algorithms for continuous model inference (1). *Error* is a measure of the difference between observed and simulated data, and different versions of this (RSS, MSE) have been used; however, their use is equivalent, as the number of genes and time points, (i.e. degrees of freedom), is the same for all individuals to be evaluated in a given optimisation run. Methods employing any type of iterated optimisation (Section 4.2.2), nested optimisation (Section 4.2.1) or *divide et impera* (Section 4.2.1) contain √ in column *Stages*. Abbreviations: ODE - ordinary differential equations, EA - evolutionary algorithm, GP - genetic programming, LMS - least means squares, PDE - partial differential equations, LDE - linear differential equations, SS - S-System, ES - evolutionary strategy, DE - differential evolution , HC - hill climbing, QP- quadratic programming.

Each structure is evaluated using a multi-objective approach, which aims at optimising sensitivity, attractor cycle length and number of attractors. The method is shown to yield more stable structures for a synthetic network of size 8. However as before, the method was not applied to real gene expression data, so further analysis is required.

4.2 Continuous models

Several algorithms for inference of continuous GRN models from gene expression data have been developed in recent years, and Tables 2 and 3 give an overview of methods. These include application of classical evolutionary techniques and development of novel algorithms, especially tailored for gene expression data.

One of the first approaches to GRN reverse engineering, based on evolutionary computation, is that of Tominaga et al. (1999). A classic, double-encoded genetic algorithm is used to

Method	EA	Mo-del	Fitness	Local search	Sta-ges	Datasets (Size)
Xu et al. (2007)	DE PSO	RNN	Error	-	-	Synthetic (8), E. Coli(8)
Koduru et al. (2007; 2004; 2005; 2008)	GA PSO	LDE, SS, RNN	Multi Objective - Error per gene	Simplex	-	Rice (2), A. Thaliana(3)
Imade et al. (2004; 2003); Morishita et al. (2003); Ono et al. (2004)	GA	SS	Error	GA	√	Synthetic (5)
Spieth, Streichert, Supper, Speer & Zell (2005); Spieth et al. (2004)	GA	SS	Error	ES	√	Synthetic (20)
Keedwell & Narayanan (2005)	GA	ANN	BP Error	BP	√	Synthetic Boolean (10), Rat (112), Yeast (2468)
Daisuke & Horton (2006)	GA	SS	Error	Simplex Crossover, Scale free	√	Synthetic (5), Mouse (7)
Spieth, Streichert, Speer & Zell (2005b)	GA, ES	SS	Multi Objective - Error, Connectivity	-	-	Synthetic (5, 10)
Kabir et al. (2010)	SA-DE	LTV	Error	-	-	Synthetic (5), E. Coli (6)

Table 3. Evolutionary algorithms for continuous model inference (2). Abbreviations: LDE - linear differential equations, SS - S-System, PSO - particle swarm optimisation, RNN - recurrent neural network, ES - evolutionary strategy, ANN - artificial neural network, BP - back-propagation, SA-DE - self adaptive differential evolution, LTV - linear time-variant.

infer S-System models from time series data. However, this method was only applied to synthetic data for a very small network (2 genes). Another more recent application of a classic evolutionary algorithm is that of Fomekong-Nanfack et al. (2007). This employs an evolutionary strategy to optimise model parameters for a 6-gene developmental network for *Drosophila Melanogaster*, based on partial differential equations, (reaction-diffusion model, Section 2.2.1). Although suitable for a larger network than Tominaga et al. (1999), the size of the inferred GRN is still very small compared to the total number of genes typically involved in such a system, showing that classical evolutionary algorithms are not powerful enough for larger networks. This was also indicated in Sîrbu et al. (2010a), where an empirical comparison was performed between different evolutionary algorithms, including Tominaga et al. (1999), and which showed that only hybrid evolutionary algorithms scaled to larger systems.

The limitation in size and model quality for quantitative analysis derives from the nature of the data to be studied. Typically, gene expression time series are too short, due to experimental limitations, while the number of parameters needed to describe quantitative models is very large. This creates an *under-determination* problem, so that multiple models can have very similar simulated behaviour, with corresponding poor reliability for inferential algorithms. Additionally these data are intrinsically noisy, hence application of algorithms to real data is not straightforward, (Sîrbu et al., 2010a). This is emphasised in Tables 2 and

3, where many algorithms are seen not to have been applied to real data. A consequence of noise and under-determination is the *ruggedness* of the fitness landscape[7] for this problem, (Rodrigo et al., 2010). However, evolutionary algorithms are known to perform well on underdetermined problems and noisy fitness functions (Mitchell, 1999), so have clear benefit over other inferential methods. Evolutionary computation approaches that address these issues can guide the optimisation towards more plausible solutions and are discussed next.

4.2.1 Addressing the under-determination problem

Divide et impera

Given that model parameters are independent for each gene, (relying only on the expression level of the genes at previous time points), one method of addressing under-determination is to use a *divide et impera* approach. This consists of separate optimisation of parameters for each individual gene, using observed rather than simulated expression levels for the other genes. This method has been implemented in several evolutionary algorithms, (Ando & Iba, 2003; Iba & Mimura, 2002; Keedwell & Narayanan, 2005; Liu et al., 2008; Noman & Iba, 2006; 2007), and has the advantage of reducing the solution space by decreasing the number of parameters to be inferred at any one time. In a recent comparison study, Sîrbu et al. (2010a), algorithms using this approach scaled better than those which sought to optimise parameters for the entire network simultaneously. However, a disadvantage of this method is that, typically, expression levels in the data are noisy, and these are used in single gene simulations, resulting in model parameters slightly different from those that would be obtained by simulating all genes. In consequence, models obtained by combining single gene solutions may not fit the data very well. This can be avoided by a second optimisation stage: starting from single gene models, evolutionary optimisation is employed to fine-tune the parameters for the entire network, (e.g. Ando & Iba (2003); Noman & Iba (2005)). Further, a model that handles noise better, such as an artificial neural network, (which employs a sigmoid function to compute expression levels), may also decrease this effect.

Obtaining skeletal/ scale free structures

To further distinguish between many possible models, known characteristics of the network structure, such as low connectivity or scale-free nature, have been considered also. Many methods apply such knowledge to the optimisation process at different levels of the algorithm. The simplest idea sets parameters to zero once these fall below a fixed threshold (Kikuchi et al., 2003; Tominaga et al., 1999). However, more advanced approaches have also been developed. For instance, Kikuchi et al. (2003); Kimura et al. (2003); Noman & Iba (2005; 2006) use an additional term that penalises solutions with large parameter values. A refinement of this penalty-based idea can be seen in methods, which start by penalising all connections, and then use a connectivity threshold to reduce possibilities. This results in more advanced fitness functions, and is possible because evolutionary optimisation, unlike numerical methods, has the advantage of not restricting fitness function type. A similar method, (Spieth, Streichert, Speer & Zell, 2005b), uses the connectivity as a second objective in multi-objective optimisation. Analogously, Ando et al. (2002); Iba (2008); Sakamoto & Iba (2001) have used genetic programming to evolve sparse ordinary differential equations, by penalising functions of large degree. Deng et al. (2005) also employed a limit on the connectivity of a linear differential equation model, evolving the connectivity parameter during optimisation, rather

[7] In evolutionary algorithms, each candidate solution can be considered a point in a multidimensional space (i.e. the solution space), with a corresponding fitness value. The fitness values for all points in the solution space generate the so called fitness landscape.

than fixing it initially, in order to find the optimal connectivity for the network, i.e. the number of regulators that achieves best data fit.
Another mechanism, used to obtain solutions with more plausible structures, is local search. For instance, Noman & Iba (2006) use hill climbing to set as many parameters to zero in two candidate solutions for each generation. Also, in Daisuke & Horton (2006), models are checked for scale-free structure, and modified if they do not comply, by adding or removing random connections, (setting the corresponding parameters to 0). All these methods result in sparser networks, as unnecessary parameters are set to zero.

Nested optimisation

Reduction in the number of parameters to be optimised has been also performed using a nested optimisation approach, (Keedwell & Narayanan, 2005; Morishita et al., 2003; Spieth, Streichert, Supper, Speer & Zell, 2005; Spieth et al., 2004). These methods divide the search into two stages: structure and parameter search. During structure search, network topology is evolved using a genetic algorithm. Candidate structures are built so that the number of regulators is bounded for each gene, and these are evaluated by a second algorithmic stage, which optimises parameters for the existing connections. This reduces the number of real-valued parameters to be inferred at the second stage. The parameter search is performed using an evolution strategy, (Spieth, Streichert, Supper, Speer & Zell, 2005; Spieth et al., 2004), a genetic algorithm, (Morishita et al., 2003), or back-propagation, (Keedwell & Narayanan (2005), with an artificial neural network as the model). Again, this is facilitated by the flexibility of fitness evaluation, which is characteristic of evolutionary algorithms. Nested optimisation increases parallelisation potential, (a parallelised version of Morishita et al. (2003) was later developed by Imade et al. (2003)). Separation of structure and parameter search is important as this allows the topology to have a larger influence in the optimisation process, rather than optimising real-valued parameters directly. This is particularly relevant in the current context as dynamical behaviour in biological networks relies mostly on topology, (Alvarez-Buylla et al., 2007), with parameter perturbations of lesser importance.

Parallelisation

Quantitative models require optimisation of a very large number of parameters, and fitness evaluation is costly in simulation terms. These costs increase when additional time series datasets are used, so parallelisation of methods is mandatory. Evolutionary algorithms have the advantage of being intrinsically parallel, facilitating efficient multi-threading of the optimisation process. Several examples of parallel implementations exist in evolutionary methods for GRN modelling, (Daisuke & Horton, 2006; Fomekong-Nanfack et al., 2007; Imade et al., 2004; 2003; Spieth, Streichert, Speer & Zell, 2005a). These correspond to both grid and cluster systems, while parallel frameworks for analysis have been implemented and are publicly available (Spieth et al., 2006; Swain et al., 2005).

4.2.2 Handling local minima

Combining multiple methods

Due to the ruggedness of the fitness landscape, an evolutionary algorithm can be trapped in local minima, and fail to find an optimal solution. One approach that seeks to avoid this is to combine different evolutionary methods. This is facilitated by their simplicity and flexibility. For instance, Xu et al. (2007) alternates differential evolution and *particle swarm optimisation*, (parameterisation of a neural network model), to obtain better overall models than from the two optimisation strategies separately. A different approach, (Daisuke & Horton, 2006; Kikuchi et al., 2003), uses *Simplex crossover*, which efficiently balances the

exploration and exploitation of the search space (Kikuchi et al., 2003), and in consequence speeds up convergence, (as the local minima problem is diminished).

Iterated optimisation

A second technique with the same aim is iterated optimisation, (possible due to the stochastic nature of evolutionary computation). Multiple runs of the same algorithm typically lead to different solutions, i.e. different local minima, which can be combined to obtain a better model: Kikuchi et al. (2003) describe a second optimisation run, initialised with these local solutions. An alternative is to analyse local solutions by methods other than evolutionary algorithms. For instance, Daisuke & Horton (2006); Deng et al. (2005) employ a *voting procedure* for connections found in multiple runs, while Noman & Iba (2005) use voting to find null parameters in the model. Similarly, Noman & Iba (2007) apply Z-score analysis to local solutions to find plausible qualitative connections for yeast cell cycle data (quantitative analysis being hampered by data limitations of length and noise).

4.2.3 Handling noise

Noise is a serious problem in gene expression measurements and, unfortunately, most of the algorithms developed for model inference from these data do not specifically take it into account. This makes many methods unfit for real data, even when validated in principle for synthetic systems. A recent comparison of evolutionary methods for quantitative model inference has enabled evaluation of method performance on noisy data, (Sîrbu et al., 2010a). While most methods give good behaviour up to 5% added noise, only two maintained this with up to 10%. One, (Keedwell & Narayanan, 2005), uses an artificial neural network to model gene regulation, while the second employs a local search procedure, based on Quadratic Programming, that handles noisy measurements, (Kimura et al., 2003). The superior performance of these two methods is a strong indication that noise needs to be explicitly addressed in the model or evolutionary process, in order to obtain algorithms that can be applied to real-world data. (Refer again to Tables 2 and 3)

5. Stage 3: Data integration for GRN modelling

Evolutionary approaches for data integration are few, so far, even though the need for inclusion of other types of data in the inferential process, (due to the under-determination problem), has been widely acknowledged, (Hecker et al. (2009) and references therein). One of the first methods attempting to incorporate previous knowledge is that of Shin & Iba (2003), where the AIC-based[8] fitness function, (similar to that of Noman & Iba (2006)), is modified to account for known interactions between genes in an S-System model. Thus models containing known interactions have better fitness and this leads the search towards regions in space that are more likely to contain the correct structure. The Shin & Iba (2003) algorithm was applied repeatedly and results were analysed using Z-scores to identify significant relationships. On synthetic data, the approach was shown to have increased sensitivity in finding correct interactions, compared to the standard method, which made no use of previous knowledge. Further, the former worked well even when previous knowledge was partially incorrect, (not unusual in a real experiment). When applied to the real microarray data of *E. Coli*, the method was also shown to identify previously known interactions.

A second data type integrated into the evolutionary optimisation process relates to knock-out experiments. Ono et al. (2004) attempted inclusion of time series knock-out data, and

[8] Akaike's Information Criterion (AIC, Noman & Iba (2006)) is an information criterion used for model selection, which is based on the error between observed and simulated data.

demonstrated that this improved the structure search. Again, the method was only applied to synthetic systems. However, Ferrazzi et al. (2007) integrated steady state knock-out measurements to infer parameters for a linear model of regulation in the cell cycle of *Saccharomyces Cerevisiae*. The additional data are used to initialise a genetic algorithm with biologically plausible interactions, by analysing differentially expressed genes in the knock-out experiments, and keeping these known interactions fixed in the structure. This enhanced approach was compared to a simple genetic algorithm, and was shown to be more robust. Thus, feeding the optimisation with interactions from knock-out data guided the algorithm towards similar solutions in the search space during different runs, (implying that these were closer to the real network of interactions).

5.1 Large scale data integration

To date, only one additional piece of knowledge data has been added to the evolutionary optimisation process. However, combining information sources could bring significant improvement. Consequently, we are developing a novel integrative evolutionary algorithm that aims at combining multiple data sources for GRN modelling. The algorithm uses an ANN as a model and is based on the idea introduced by Keedwell & Narayanan (2005) of dividing the structure and parameter optimisation process. A genetic algorithm is used for structure search, and back-propagation for parameter search and structure evaluation. As this is a *divide et impera* approach, a second optimisation stage has been added to the original algorithm, which performs fine tuning of models, obtained by combining single gene results. This second stage is executed after iterating the first stage several times for each gene with different connectivity limits (i.e. different number of regulators), so each gene connectivity is optimised to obtain structures that are better able to simulate the data. Additionally, the algorithm allows for reducing the list of possible regulators to a user defined list, (from previously known transcription factors, if available). This further decreases the search space and allows introduction of meta-information into the algorithm.

A first step for heterogeneous data integration is combining gene expression data from different sources. Although time series from one laboratory are typically short, multiple data sets from different sources, which nevertheless measure the same process, are available. A study of cross-platform microarray data integration has been performed, (Sîrbu et al., 2010c), and has demonstrated that models obtained from combined data are both more robust to noise and parameter perturbation, and display less noise over-fitting. Additionally, normalisation methods for multiple microarray platforms have been analysed in the context of GRN modelling (Sîrbu et al., 2010b).

It appears that time series data integration does reduce the under-determination problem, but it makes the inferential process much more computationally intensive, (especially for evolutionary optimisation, as multiple series have to be simulated for each fitness evaluation). To address this, a fine-grained parallelisation of the algorithm has been performed. This uses a different processor to evaluate each candidate solution in the population. Additionally, during the first stage, individual gene runs are also divided between multiple processors.

A second step for data integration, as seen earlier, is that of using steady state knock-out experiments. These are used here not only for initialisation, as in Ferrazzi et al. (2007), but also for model evaluation. Unlike Ferrazzi et al. (2007), the structure extracted from these data is not fixed, so the algorithm can change this during optimisation, in order to select the interactions that give better data fit. For evaluation, the models are simulated for a user-defined number of time points, to reach the steady state, using null expression values for the knocked out genes and starting with expression values of 0.5, (middle of interval $(0, 1)$, which is the range of gene expression values), for the rest. The error between the resulting

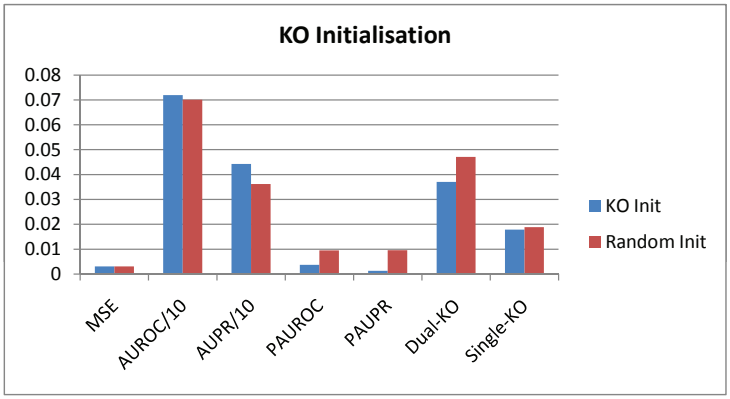


Fig. 3. Initialisation with knock-out (KO) experiments. Graph shows average values over three runs for each experiment: KO Init (initialisation with knock-out experiments) and Random Init (random initialisation). Qualitative results are displayed using AUROC (area under ROC curve), AUPR (area under precision-recall curve), PAUROC (p-value of AUROC) and PAUPR (p-value of AUPR). Quantitative results show MSE values between data and simulation for knock-out (Dual KO and Single KO) and wild-type (MSE) experiments. Results obtained using KO Init are better both quantitatively and qualitatively.

state and the data is used as an additional term in refining the fitness function. This evaluation is only performed during the second stage of the algorithm, when the model for the entire network is optimised, as it requires simulated values for all the genes in the network.

We have validated this approach with simulated gene expression data from the DREAM 4 challenge, (Marbach et al., 2009). Although these are synthetic data, they are generated using models inspired by real GRNs,(as provided by the authors), so have plausible structures. Also, the data contain added noise, in an effort to mimic real gene expression data. We have compared models, which use both knock-out initialisation and fitness evaluation, to those from the simple version of the algorithm, and results are shown in Figures 3 (for initialisation) and 4 (for evaluation). These show that adding initialisation with knock-out experiments leads to models containing more correct interactions, (higher AUPR, AUROC and lower p-values), which can simulate known dynamical behaviour, (giving lower MSE values in both wild type and knock-out simulations). If knock-out experiments are used for evaluation, simulation of wild-type experiments is disimproved (higher MSE), but this may be due to residual noise, given the improvement in dual and single knock-out simulations and in qualitative results, (more correct interactions). Considering the promising results for synthetic systems, validation of the method against real data is clearly required.

Given that structure is important in simulating GRN behaviour, with currently used fitness functions, (based on error between observed and simulated data), unable to reward those models that can reproduce these, albeit with shifts in expression values, an additional evaluation term has been introduced, (Sîrbu et al., 2010d). This measures the Pearson correlation between observation and simulation. Additionally, inclusion of this term in back-propagation, allows us to find parameters that minimise the error and maximise the correlation. In comparison to the initial algorithm, (incorporating error-based fitness and back-propagation), the modified approach is found to yield better structures (with higher

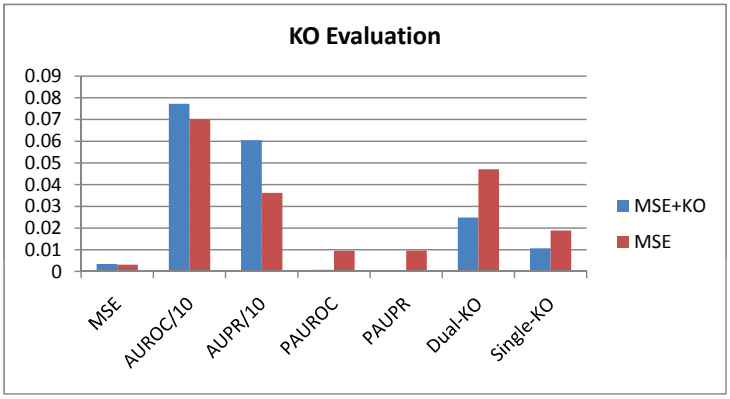


Fig. 4. Evaluation with knock-out experiments. Average performance over three runs for each experiment: MSE+KO (with knock-out evaluation) and MSE (without knock-out evaluation). Qualitative and quantitative results are analysed using the same criteria as in Figure 3. These indicate better behaviour for models obtained from MSE+KO.

AUPR and AUROC) for synthetic data (from DREAM4) and more plausible interactions in real microarray data (Yeast cell cycle). Further details are given in Sîrbu et al. (2010d). Extension of the integrative framework is anticipated for future work, which should involve inclusion of promoter sequence and binding affinity data (ChIP-chip), as well as incorporation of RNA-seq and RNA-interference measurements in model evaluation and inference process.

6. Conclusions

This chapter has presented the role of evolutionary algorithms at different stages of gene regulatory network inference. These include (i) expression pattern analysis, (ii) model inference from time series data and (iii) data integration for model inference. For (i), methods for clustering and feature selection for gene expression data have been described. For (ii), method development from classical to more advanced hybrid algorithms has been presented. This has been motivated by issues in network modelling, such as under-determination and noisy data. These issues have been addressed to some extent by taking advantage of the flexibility and power of evolutionary approaches. For instance, the flexibility of the fitness function has been used to reward models with sparse or scale free structures. Hybridisation with local search and other optimisation algorithms has also benefited from the simple basis of the evolutionary algorithmic scheme, in order to avoid local minima traps and to handle noise. Additionally, the parallelisation potential of these methods, combined with their stochastic underpinning, has led to iterated algorithm versions, (designed to handle local solutions), and nested optimisation, (used to limit the number of real-valued parameters to be addressed). All these improvements have permitted a scale-up of quantitative modelling, from 2 to 30 genes. However, this is still very modest compared to real GRN requirements.
Many of the methods presented have, to date, been applied only to synthetic data, while most applications to real data can yield only qualitative results, as quantitative models obtained remain unreliable. In order to further improve inference, different data sources can be combined, and this has been presented as the third stage of GRN inference. Advances in high-throughput technologies other than microarrays and global research efforts have created

very large biological data sets containing protein-protein interactions, protein measurements, knock-out experiments, protein binding sites and gene sequence information. Although current such data are insufficient to determine the underlying GRN, combining them could prove to be very powerful and EAs are flexible enough to enable their integration. Existing methods, nevertheless, under-exploit EC potential, to some extent, by integrating only one additional data type. In consequence, we have outlined here the basis for a novel framework, which is being developed and which aims at large-scale data integration for GRN quantitative modelling. This uses fitness evaluation, initialisation and parallelisation to include heterogeneous data and knock-out experiments in the optimisation process. The framework will be extended in the future to employ promoter sequences and protein binding affinities, (such as those extracted from ChIP-Chip data), for model evaluation, and will integrate RNA-seq data and RNA-interference measures in the inferential process.

7. Acknowledgements

We wish to acknowledge the SFI/HEA Irish Centre for High-End Computing (ICHEC) for the provision of computational facilities and assistance. This work has been conducted with support from the Irish Research Council for Science Engineering and Technology EMBARK Initiative.

8. References

Akutsu, T., Miyano, S. & Kuhara, S. (2000). Inferring qualitative relations in genetic networks and metabolic pathways, *Bioinformatics* **16**(8): 727–734.
URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/16/8/727*

Alvarez-Buylla, E. R., Benitez, M., Davila, E. B., Chaos, A., Espinosa-Soto, C. & Padilla-Longoria, P. (2007). Gene regulatory network models for plant development, *Current Opinion in Plant Biology* 10(1): 83 – 91.

Ando, S. & Iba, H. (2003). Estimation of gene regulatory network by genetic algorithm and pairwise correlation analysis, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on* 1: 207–214.

Ando, S., Sakamoto, E. & Iba, H. (2002). Evolutionary modeling and inference of gene network, *Inf. Sci. Inf. Comput. Sci.* 145(3-4): 237–259.

Baeck, T., Fogel, D. B. & Michalewicz, Z. (2000). *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physics Publishing Bristol and Philadelphia.

Baldi, P. & Hatfield, W. (2002). *DNA Microarray and Gene Expression. From experiments to data analysis and modeling*, Cambridge University Press.

Bar-Joseph, Z. (2004). Analyzing time series gene expression data, *Bioinformatics* 20(16): 2493–2503.

Brown, T. (2002). *Genomes*, BIOS Scientific publishers Ltd.

Bryan, K. (2005). Biclustering of expression data using simulated annealing, *CBMS '05: Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, IEEE Computer Society, Washington, DC, USA, pp. 383–388.

Chakraborty, A. & Maka, H. (2005). Biclustering of gene expression data using genetic algorithm, *Computational Intelligence in Bioinformatics and Computational Biology, 2005. CIBCB '05. Proceedings of the 2005 IEEE Symposium on*, pp. 1 – 8.

Cheng, Y. & Church, G. M. (2000). Biclustering of expression data, *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, pp. 93–103.

Daisuke, T. & Horton, P. (2006). Inference of scale-free networks from gene expression time series, *Journal of Bioinformatics and Computational Biology* 4: 503–514.

Deng, X., Geng, H. & Ali, H. (2005). Examine: a computational approach to reconstructing gene regulatory networks, *Biosystems* 81: 125–136.

Di Gesu, V., Giancarlo, R., Lo Bosco, G., Raimondi, A. & Scaturro, D. (2005). Genclust: A genetic algorithm for clustering gene expression data, *BMC Bioinformatics* 6(1): 289.
URL: *http://www.biomedcentral.com/1471-2105/6/289*

Eriksson, R. & Olsson, B. (2004). Adapting genetic regulatory models by genetic programming, *Biosystems* 76(1-3): 217 – 227. Papers presented at the Fifth International Workshop on Information Processing in Cells and Tissues.

Esmaeili, A. & Jacob, C. (2009). A multi-objective differential evolutionary approach toward more stable gene regulatory networks, *Biosystems* 98(3): 127 – 136.

Ferrazzi, F., Magni, P., Sacchi, L., Nuzzo, A., Petrovic, U. & Bellazzi, R. (2007). Inferring gene regulatory networks by integrating static and dynamic data, *International Journal of Medical Informatics* 76(Supplement 3): S462 – S475.

Fomekong-Nanfack, Y., Kaandorp, J. A. & Blom, J. (2007). Efficient parameter estimation for spatio-temporal models of pattern formation: case study of Drosophila melanogaster, *Bioinformatics* 23(24): 3356–3363.
URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/23/24/3356*

He, F., Balling, R. & Zeng, A.-P. (2009). Reverse engineering and verification of gene networks: Principles, assumptions, and limitations of present methods and future perspectives, *Journal of Biotechnology* 144(3): 190 – 203. Systems Biology for Biotechnological Innovation.

Hecker, M., Lambeck, S., Toepfer, S., van Someren, E. & Guthke, R. (2009). Gene regulatory network inference: Data integration in dynamic models–a review, *Biosystems* 96(1): 86 – 103.

Hurd, P. J. & Nelson, C. J. (2009). Advantages of next-generation sequencing versus the microarray in epigenetic research, *Briefings in Functional Genomics & Proteomics* 8(3): 174–183.
URL: *http://bfg.oxfordjournals.org/content/8/3/174.abstract*

Iba, H. (2008). Inference of differential equation models by genetic programming, *Information Sciences* 178(23): 4453–4468.

Iba, H. & Mimura, A. (2002). Inference of a gene regulatory network by means of interactive evolutionary computing, *Information Sciences* 145(3-4): 225–236.

Imade, H., Mizuguchi, N., Ono, I., Ono, N. & Okamoto, M. (2004). "gridifying" an evolutionary algorithm for inference of genetic networks using the improved goga framework and its performance evaluation on obi grid, *LSGRID*, pp. 171–186.

Imade, H., Morishita, R., Ono, I., Ono, N. & Okamoto, M. (2003). A grid-oriented genetic algorithm for estimating genetic networks by s-systems, *SICE 2003 Annual Conference* 3: 2750–2755 Vol.3.

Kabir, M., Noman, N. & Iba, H. (2010). Reverse engineering gene regulatory network from microarray data using linear time-variant model, *BMC Bioinformatics* 11(Suppl 1): S56.
URL: *http://www.biomedcentral.com/1471-2105/11/S1/S56*

Keedwell, E. & Narayanan, A. (2005). Discovering gene networks with a neural-genetic hybrid, *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 2(3): 231–242.

Kerr, G., Ruskin, H. J., Crane, M. & Doolan, P. (2008). Techniques for clustering gene expression data, *Computers in Biology and Medicine* 38(3): 283–293.

Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K. & Tomita, M. (2003). Dynamic modeling of genetic networks using genetic algorithm and S-system, *Bioinformatics* 19(5): 643–650. URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/19/5/643*

Kimura, S., Hatakeyama, M. & Konagaya, A. (2003). Inference of S-system models of genetic networks using a genetic local search, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on* 1: 631–638 Vol.1.

Koduru, P., Das, S. & Welch, S. M. (2007). Multi-objective hybrid pso using μ-fuzzy dominance, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 853–860.

Koduru, P., Das, S., Welch, S. & Roe, J. L. (2004). Fuzzy dominance based multi-objective GA-Simplex hybrid algorithms applied to gene network models, *Genetic and Evolutionary Computation - GECCO 2004*, pp. 356–367.

Koduru, P., Das, S., Welch, S., Roe, J. L. & Lopez-Dee, Z. P. (2005). A co-evolutionary hybrid algorithm for multi-objective optimization of gene regulatory network models, *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 393–399.

Koduru, P., Dong, Z., Das, S., Welch, S., Roe, J. L. & Charbit, E. (2008). A multiobjective evolutionary-simplex hybrid approach for the optimization of differential equation models of gene networks, *IEEE Trans. Evolutionary Computation* 12(5): 572–590.

Lee, W.-P. & Tzou, W.-S. (2009). Computational methods for discovering gene networks from expression data, *Briefings in Bioinformatics* 10(4): 408–423. URL: *http://bib.oxfordjournals.org/cgi/content/abstract/10/4/408*

Lee, W.-P. & Yang, K.-C. (2008). A clustering-based approach for inferring recurrent neural networks as gene regulatory networks, *Neurocomputation* 71(4-6): 600–610.

Li, L. (2001). Gene assessment and sample classification for gene expression data using a genetic algorithm/k-nearest neighbor method, *Combinatorial chemistry and high throughput screening* 4: 727.

Li, L., Umbach, D. M., Terry, P. & Taylor, J. A. (2004). Application of the ga/knn method to seldi proteomics data, *Bioinformatics* 20(10): 1638–1640.

Liang, S., Fuhrman, S. & Somogyi, R. (1998). Reveal: a general reverse engineering algorithm for inference of genetic network architectures, *Pacific Symposium on Biocomputing* pp. 18–29.

Linden, R. & Bhaya, A. (2007). Evolving fuzzy rules to model gene expression, *Biosystems* 88(1-2): 76 – 91.

Liu, H., Li, J. & Wong, L. (2002). A Comparative Study on Feature Selection and Classification Methods Using Gene Expression Profiles and Proteomic Patterns, *Genome Informatics* 13.

Liu, K.-H., Zhang, J., Li, B. & Du, J.-X. (2009). A ga-based approach to ica feature selection: An efficient method to classify microarray datasets, *ISNN 2009: Proceedings of the 6th International Symposium on Neural Networks*, Springer-Verlag, Berlin, Heidelberg, pp. 432–441.

Liu, X., Fu, J., Gu, D., Liu, W., Liu, T., Peng, Y., Wang, J. & Wang, G. (2008). Genome-wide analysis of gene expression profiles during the kernel development of maize (zea mays l.), *Genomics* 91(4): 378 – 387.

Logan, J., Edwards, K. & Saunders, N. (eds) (2007). *Real-Time PCR: Current Technology and Applications*, Caister Academic Press.

Lu, Y., Lu, S., Fotouhi, F., Deng, Y. & Brown, S. (2004a). Fgka: A fast genetic k-means algorithm: March 2004., *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC)*.

Lu, Y., Lu, S., Fotouhi, F., Deng, Y. & Brown, S. (2004b). Incremental genetic k-means algorithm and its application in gene expression data analysis, *BMC Bioinformatics* 5(1): 172.
URL: *http://www.biomedcentral.com/1471-2105/5/172*

Maki, Y., Tominaga, D., Okamoto, M., Watanabe, S. & Eguchi, Y. (2001). Development of a system for the inference of large scale genetic networks., *Pacific Symposium on Biocomputing* pp. 446–458.
URL: *http://view.ncbi.nlm.nih.gov/pubmed/11262963*

Manning, C. D. & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*, MIT Press, chapter 14.

Marbach, D., Schaffter, T., Mattiussi, C. & Floreano, D. (2009). Generating realistic in silico gene networks for performance assessment of reverse engineering methods., *Journal of Computational Biology* 16(2): 229–239.
URL: *http://dx.doi.org/10.1089/cmb.2008.09TT*

Mitchell, M. (1999). *An Introduction to Genetic Algorithms*, The MIT Press.

Mitra, S. & Banka, H. (2006). Multi-objective evolutionary biclustering of gene expression data, *Pattern Recognition* 39(12): 2464 – 2477.

Morishita, R., Imade, H., Ono, I., Ono, N. & Okamoto, M. (2003). Finding multiple solutions based on an evolutionary algorithm for inference of genetic networks by S-system, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on* 1: 615–622 Vol.1.

Noman, N. & Iba, H. (2005). Inference of gene regulatory networks using s-system and differential evolution, *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 439–446.

Noman, N. & Iba, H. (2006). Inference of genetic networks using S-system: information criteria for model selection, *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 263–270.

Noman, N. & Iba, H. (2007). Inferring gene regulatory networks using differential evolution with local search heuristics, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4(4): 634–647.

Ono, I., Seike, Y., Morishita, R., Ono, N., Nakatsui, M. & Okamoto, M. (2004). An evolutionary algorithm taking account of mutual interactions among substances for inference of genetic networks, *Evolutionary Computation, 2004. CEC2004. Congress on* 2: 2060–2067 Vol.2.

Ooi, C. H. & Tan, P. (2003). Genetic algorithms applied to multi-class prediction for the analysis of gene expression data, *Bioinformatics* 19(1): 37–44.

Pal, S., Bandyopadhyay, S. & Ray, S. (2006). Evolutionary computation in bioinformatics: a review, *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 36(5): 601–615.

Przytycka, T. M., Singh, M. & Slonim, D. K. (2010). Toward the dynamic interactome: it's about time, *Briefings in Bioinformatics* 11(1): 15–29.
URL: *http://bib.oxfordjournals.org/cgi/content/abstract/11/1/15*

Repsilber, D., Liljenström, H. & Andersson, S. G. E. (2002). Reverse engineering of regulatory networks: simulation studies on a genetic algorithm approach for ranking hypotheses, *Biosystems* 66(1-2): 31 – 41.

Robnik-Łikonja, M. & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff, *Machine Learning* 53: 23– 69.

Rodrigo, G., Carrera, J. & Elena, S. F. (2010). Network design meets in silico evolutionary biology, *Biochimie* 92(7): 746 – 752.

Sakamoto, E. & Iba, H. (2001). Inferring a system of differential equations for a gene regulatory network by using genetic programming, *Proceedings of Congress on Evolutionary Computation*, pp. 720–726.

Shah, S. C. & Kusiak, A. (2004). Data mining and genetic algorithm based gene/SNP selection, *Artificial Intelligence in Medicine* 31(3): 183 – 196.

Shin, A. & Iba, H. (2003). Construction of genetic network using evolutionary algorithm and combined fitness function., *Genome Informatics* 14: 94–103.
URL: *http://view.ncbi.nlm.nih.gov/pubmed/15706524*

Sîrbu, A., Ruskin, H. J. & Crane, M. (2010a). Comparison of evolutionary algorithms in gene regulatory network model inference, *BMC Bioinformatics* 11(59).

Sîrbu, A., Ruskin, H. J. & Crane, M. (2010b). Cross-platform microarray data normalisation for regulatory network inference, *PLoS ONE* 5(11):e13822.

Sîrbu, A., Ruskin, H. J. & Crane, M. (2010c). Integrating heterogeneous gene expression data for gene regulatory network modelling, *Proceedings of the European Conference on Complex Systems*, Lisbon, Portugal. 13-17 Sept.

Sîrbu, A., Ruskin, H. J. & Crane, M. (2010d). Regulatory network modelling: Correlation for structure and parameter optimisation, *Proceedings of the IASTED Technology Conferences (ARP,RA,NANA,ComBio 2010)*, Cambridge, Massachusetts. 1-3 Nov.
URL: *http://www.actapress.com/Abstract.aspx?paperId=41573*

Souza, B. F. & Carvalho, A. P. (2005). Gene selection based on multi-class support vector machines and genetic algorithms., *Genetics and Molecular Research* 4(3): 599–607.

Spieth, C., Streichert, F., Speer, N. & Zell, A. (2005a). Clustering-based approach to identify solutions for the inference of regulatory networks, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, Vol. 1, pp. 660–667.

Spieth, C., Streichert, F., Speer, N. & Zell, A. (2005b). Multiobjective model optimization for inferring gene regulatory networks, *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization, volume 3410 of Lecture Notes in Computer Science*, pp. 607–620.

Spieth, C., Streichert, F., Supper, J., Speer, N. & Zell, A. (2005). Feedback memetic algorithms for modeling gene regulatory networks, *Computational Intelligence in Bioinformatics and Computational Biology, 2005. CIBCB '05. Proceedings of the 2005 IEEE Symposium on* pp. 1–7.

Spieth, C., Streichert, F. & Zell, N. S. A. (2004). Optimizing topology and parameters of gene regulatory network models from time-series experiments, *Genetic and Evolutionary Computation - GECCO 2004*, pp. 461–470.

Spieth, C., Supper, J., Streichert, F., Speer, N. & Zell, A. (2006). JCell–a Java-based framework for inferring regulatory networks from time series data, *Bioinformatics* 22(16): 2051–2052.
URL: *http://bioinformatics.oxfordjournals.org/cgi/content/abstract/22/16/2051*

Stekel, D. (2003). *Microarray Bioinformatics*, Cambridge University Press.

Swain, M., Hunniford, T., Dubitzky, W., Mandel, J. & Palfreyman, N. (2005). Reverse-engineering gene-regulatory networks using evolutionary algorithms and grid computing, *J Clin Monit Comput* 19(4-5): 329–37.

Tan, K., Tegner, J. & Ravasi, T. (2008). Integrated approaches to uncovering transcription regulatory networks in mammalian cells, *Genomics* 91(3): 219 – 231.

Thieffry, D. (1999). From global expression data to gene networks, *BioEssays* 21: 895–899.

Tominaga, D., Okamoto, M., Maki, Y., Watanabe, S. & Eguchi, Y. (1999). Nonlinear numerical optimization technique based on a genetic algorithm for inverse problems: Towards

the inference of genetic networks, *GCB99 German Conference on Bioinformatics*, pp. 101–111.

Vohradsky, J. (2001). Neural network model of gene expression, *The FASEB Journal* 15: 846–854.

Wahde, M. & Hertz, J. (2000). Coarse-grained reverse engineering of genetic regulatory networks, *Biosystems* 55(1-3): 129–136.
URL: *http://dx.doi.org/10.1016/S0303-2647(99)00090-8*

Xu, R., Venayagamoorthy, G. K. & Donald C. Wunsch, I. (2007). Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization, *Neural Networks* 20(8): 917–927.

Zhu, Z., Jia, S. & Ji, Z. (2010). Towards a memetic feature selection paradigm, *Computational Intelligence Magazine* 5(2): 41–53.

Zhu, Z. & Ong, Y.-S. (2007). Memetic algorithms for feature selection on microarray data, *in* D. Liu, S. Fei, Z.-G. Hou, H. Zhang & C. Sun (eds), *Advances in Neural Networks Ű ISNN 2007*, Vol. 4491 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 1327–1335.

Zhu, Z., Ong, Y.-S. & Dash, M. (2007). Markov blanket-embedded genetic algorithm for gene selection, *Pattern Recognition* 40(11): 3236–3248.

Zhu, Z., Ong, Y.-S. & Zurada, J. (2010). Identification of full and partial class relevant genes, *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 7(2): 263 –277.

28

Evolutionary Algorithms in Crystal Structure Analysis

Attilio Immirzi[1], Consiglia Tedesco[2] and Loredana Erra[3]
[1,2]*Dept. of Chemistry, University of Salerno*
[3]*European Synchrotron Radiation Facility*
[1,2]*Italy*
[3]*France*

1. Introduction

Evolutionary algorithms (EA, see for a general introduction Holland, 1975; Goldberg, 1989; Davis, 1991; Back, 1996) are "probabilistic" optimization procedures. Differently from the "deterministic" ones (yielding an unique and reproducibile solution for assigned input data), EA are somewhat aleatory and not reproducible. Such methods are part of the heuristics (Polya, 1971). To be pedantic the term algorithm should indicate some *sequence of mathematical operations producing a foreseeable result if applied to definite input data*. There is a resounding oxymoron in the terms *evolutionary algorithm*. This contradictory language is however diffuse; as this one is not an erudite essay of epistemology, we prefer to be tolerant and fly over on semantic questions. Really evolutionary methods have aleatory routes and also aleatory conclusions in many cases (e.g. stock exchange estimate, weather forecast, etc.); in some cases instead (it happens in crystallography), unique and certain results can be reached in spite of the multiplicity of routes travelled.

Multi-parameter optimization problems are encountered in many fields: industrial process planning, financial investments, environment control, hurricane evolution, and many others (Weise, 2009). We shall however not consider the general aspects of EA but rather to treat about a specific argument: the study of chemical structures based on X-ray diffraction. In this field, as shown later, there is a lot of *a priori* and useful information so that the aleatority of the procedure can be considerably attenuated. Evolutionary methods in crystallography are better formulated as *Constraint Satisfaction Problems* (CSP, see Ionita et al., 2010).

Evolutionary methods have a future in crystallography, particularly in solving hard problems. The structural elucidation from X-ray powder diffraction data is among these (section 4.1). However, procedures are not consolidated yet, there are numerous alternatives to be considered for which the common sense and the experience, rather than the theory, could have a role.

This article will not treat the EA in the crystallographic context in its generality; rather it will summarize some ideas and strategies which yielded a specific procedure: it is a possible route, not the only possible one. These procedures are now included in a general-purpose computer program with numerous options for finding and refining crystal structures termed TRY (Immirzi, 2007b). In its actual structure the EA procedure needs that many parameters

are assigned by the user. Assigning them in a more automatic fashion could no doubt improve the program versatility.
The effectiveness of EA in solving crystal structures by diffraction methods is improved if one makes systematically use of the so called *internal coordinates* (i.c.) for describing structures, instead of the more common *cartesian coordinates*. The i.c. were first considered, in diffraction analysis, by Arnott & Wonacott (1966). A computer program performing least-square refinement of i.c. (LALS, Linked Atom Least Square) was first issued (Smith & Arnott, 1978) and later updated repeatedly (the last edition, WINLALS , was published by Okada et al., 2003). LALS has been used mainly in studying polymers. Also Tadokoro (1975) discussed the use of i.c. in polymer case.
Both Arnott (and disciples) and Tadokoro did however neglect a very important complication: the mutual independency (non-redundancy) of i.c., being of great importance in all crystallogaphic applications, and in particular in EA, as will be shown later.
Now a procedure for carrying-out a molecular building *always using non-redundant i.c.* has been devised (Immirzi 2007a,b); it is very simple, practical, and perfectly analytical, runs thoroughly, and eliminates the necessity of using exotic instruments in matrix manipulations (pseudoinversion, diagonalization, etc.; note that the mathematic instruments employed are those of Newton, Gauss, and Lagrange).

2. Crystalline matter and X-ray diffraction

Characterization of chemicals, both natural and synthetic, is done always looking at the matter at the atomic scale. In 19th century, atoms were speculative entities and scientists had not instruments for ascertaining their shape or position. Nevertheless, in the middle of century, some of them (Kekulé, Le Bel, van't Hoff and others), with evident attitude versus the heuristics (without awareness of course as heuristics did not exist yet) were persuaded about molecules (proposed many years before by Avogadro finding much skepticism); they begun, with much imagination and using logical arguments, to give them definite shapes, for instance the tetrahedral shape to methane and the hexagonal shape to benzene. Many of these intuitions were demostrated later be perfectly exact. Today to ascertain the geometrical structure of molecules is not more a work of imagination: it is an exact science. Structure is a "property" of any pure substance, defined and reproducible, like colour, density, melting temperature, etc.[1]
The discovery of X-ray diffraction by crystals by Laue in 1912 made possible to ascertain the structure. In the crystalline state, necessarily solid, atoms are ordered: there is a small portion of space, a parallelepiped, generally oblique, termed the *unit cell*, which repeats itself by translation in three directions and generates the whole solid. If the disposition of atoms in the cell is determined, the whole crystal is determined. Frequently there is also symmetry: only a fraction of the unit cell is independent; the remainder if given by appropriate orthogonal transforms. Edges and interaxial angles of the unit cell are collectively termed *lattice constants*.
Just order makes possible X-ray diffraction of crystal, whose study permits to guess the matter at the atomic scale; resolution is of the order of 0.02 Å or better; consider that the separation of bonded atoms is 1-2 Å. The size of molecules studied by X-ray diffraction was initially modest; now studying 50-100 atom molecules is a common matter.

[1] There are indeed exceptions: some substances exhibit more than one structure (*polymorphism*). Crystalline minerals are frequently polymorphic.

3. Molecules and molecular crystals

Molecules are finite assemblies of atoms joined by strong forces (chemical bonds). In molecular crystals molecules are orderly assembled through well weaker non-bonding forces (van der Waals forces, dipole-dipole forces, hydrogen bonds, etc.).[2]

Thanks to the repetition, crystals diffract X-ray radiation: when a beam of monochromatic radiation strikes a crystalline sample many beams emerge and their intensities can be one-by-one measured. The intensity of the diffracted beams is related to the molecular structure (see below).[3]

In absence of symmetry, if V is the volume of the unit cell, λ the radiation wavelength, and $2\vartheta_m$ the highest angular deviation, there are altogether $3.35(V/\lambda^3)\sin^3\vartheta_m$ different diffracted beams (reflections). The above relationship indicates that the number of reflections raises as $\sin\vartheta_m/\lambda$ increases. The latter is an important parameter: the higher is $\sin\vartheta_m/\lambda$ the higher is the chance of success in a structural analysis, and the higher is the accuracy of the result. Consider however that the intensity of the reflections decreases as ϑ angle increases and that $\sin\vartheta_m/\lambda$ may be limited by natural circumstances: there are materials which are intrinsecally weakly diffracting; their structural study cannot be done accurately.

4. Structure and diffraction

It is possible to obtain crystal structures from X-ray diffraction data thanks to the mathematical relationship between atomic positions and the intensities of the reflections. The latter, as numerous as the volume of the unit cell is large (see above), are singled out by three integer numbers: the Bragg indices (h,k,ℓ). Each is deviated from the incident direction by a characteristic angle 2ϑ, an analytical function of the Bragg indices and of the lattice constants. One must intercept these beams and measure intensity.

There are nowadays very sophisticated instruments (*diffractometers*), controlled by computers, allowing, with a minimal human intervention and in a short time, a complete characterization of a crystal: measurement of the lattice constants, diagnosis of symmetry, localization and measurement of all reflections with their Bragg indices h,k,ℓ. If the unit cell contains N atoms, and x_j,y_j,z_j are the atomic coordinates (referred to the unit cell), the complex quantities

$$F(h,k,\ell)=\sum_{j=1}^{N} f_j \exp[2\pi i(h\frac{x_j}{a}+k\frac{y_j}{b}+\ell\frac{z_j}{c})] \tag{1}$$

are termed *structure factors*. The f_j (atomic factors) are known real quantities which depend on the chemical nature of the j-th atom and on $\sin\vartheta/\lambda$. The larger is the atomic number the higher are f_j. In every cases f_j decreases as $\sin\vartheta/\lambda$ increases.

The above relationship applies at 0 K; at higher temperature F's are somewhat reduced because of the *thermal vibration*. There are of course as many $F(h,k,\ell)$ as reflections.

Now the squared moduli of $F(h,k,\ell)$ should be orderly proportional to the measured diffraction intensities $I(h,k,\ell)$. This occurs, of course, when the x_j,y_j,z_j coordinates are the true ones. To get the unknown crystal structure one must find all the x_j,y_j,z_j rendering the

[2] Not all crystalline materials are molecular . There also ionic crystals and *extended covalent structures*.

[3] The deviation of diffracted beams from the incident direction obeys to the Bragg's law. Geometry is quite similar to the case of a reflecting mirror: diffracted beams can be considered as "reflected" by the sample. For this reason they are frequently termed *reflections*

$|F(h,k,\ell)|^2$ as close as possible to the $I(h,k,\ell)$. Introducing the residual

$$\chi^2 = \sum_{hk\ell} [I(h,k,\ell) - |F(h,k,\ell)|^2]^2 \tag{2}$$

one can assume that the solution is the one rendering χ^2 a minimum. For this reason to find a structure is to solve a *global optimization problem*. The above χ^2, divided by the sum of $F^2(h,k,\ell)$ is the common R_2 index, widely used as *fitness function*.

4.1 Single-crystals and polycrystalline samples

X-ray diffraction studies use either *single-crystal samples* or *polycrystalline samples* (powder, fiber). In the second case things are more difficult because all diffracted beams having a given 2ϑ are overlapped. While the diffraction pattern of a single crystal is a three-dimensional function $I(h,k,\ell)$ rich of information, the powder diffraction is an uni-dimensional function $I(2\vartheta)$ with poor information.

Structural studies based on powder diffraction data meet the difficulty of measuring the diffracted intensities one-by-one, singled out by the Bragg indices, in presence of overlap. The difficulty can be overcome employing deconvolution techniques (see e.g. Harris, 1998) or, alternatively, renouncing to the separation of reflections and considering the quasi-continous function $I(2\vartheta)$ (Rietveld, 1967; Rietveld, 1969; Young, 1995) exploiting the so called *full-pattern powder profile analysys*.

In this case, the measured diffraction intensities $I(2\vartheta_i)$ are compared with the computed ones given by

$$I_{calcd,i} = \sum_k F_k^2 \, \Omega(2\vartheta_i - 2\vartheta_k) \tag{3}$$

where the sum is extended to *all the reflections whose Bragg* $2\vartheta_k$ *fall near to the current* $2\vartheta_i$ *point*. In equation (3) $\Omega(2\vartheta_i - 2\vartheta_k)$ is the *peak function* having a maximum for $2\vartheta_i = 2\vartheta_k$. In full-pattern profile analysis the residual χ^2 considered is given by the expression

$$\chi^2 = \sum_{2\vartheta_i} [I(2\vartheta_i) - I_{calcd,i}]^2$$

and the most used fitness index is $R_{wp} = \chi^2 / \sum I^2(2\vartheta)$

Just because EA techniques need a reduced number of data they are attractive in studies based on powder diffraction, particularly when the full-pattern profile analysis is performed. For recent studies see Harris (1998), Feng (2006), Hanson (2007), Oganov (2006).

When a structure has been solved (at a coarse level) a second problem arises: to refine atomic positions. This is a *local optimization problem*. Once again the goal is to find the minimum of χ^2 varying systematically atomic positions *in the vicinity of the initial coarse values* and adding some parameters for expressing the thermal vibration.

While the local optimization is done using deterministic algorithms (typically a least-square refinement) and proceeds, in most cases, without problems, the global optimizazion, even at coarse level, is much more exacting.

The remainder of this chapter concerns only global optimization problems. In addition we shall not treat the traditional solution methods but only some based on evolutionary algorithms.

5. The three-dimensional space and the problem space

Solving a structure of N atoms means finding N positions in the 3D space. Unfortunately there are no algorithms for finding positions one by one (unless rather simple cases occurring in inorganic chemistry). Atomic positions must be find simultaneously.
An alternative way to define the terms of problem is to refer to another space, the so called *problem space*, having $3N$ dimensions instead of three. To catch a structure means to locate a *single point* in this hyper-space. Simple to say, but very difficult to do. As explained later the EA follow this crazy idea.
Small-medium size problems, are solved using deterministic algorithms, provided that the available diffraction measurements are numerous, accurate, and complete (i.e. including all the intensities with the diffraction angles ϑ lower than the assigned $2\vartheta_m$). For these methods, not discussed in this article, there are many excellent textbooks (e.g. Giacovazzo et al., 2002; Stout & Jensen, 1995).[4] The EA are indicated instead in difficult cases: limited $2\vartheta_m$, poor quality of data, low number of reflections.
One could ingenuously believe that all structural problems can be solved moving the representative point systematically in the whole problem-space and examining all solutions. Atomic coordinates are however real and continuous quantities; so points are infinite. Well, even though finite (and coarse) intervals are considered to render the problem-space discontinuous, one obtains astronomic numbers also in the simple cases (see later). The systematic exploration of the whole problem-space is not a practical tool, unless some tricks are adopted. Possible strategies are:

i) to reduce substantially the number of variables adopting other kinds of coordinates rather than crystallographic (see section 6) and choosing coordinate systems for which a good fraction of variables are predictable;

ii) to render the problem-space discontinuous;

iii) to reduce as much as possible the range of variation for each variable.

We shall suppose that the crystal under study is a molecular crystal with known composition (*chemical formula*) and known interatomic connection (*structural formula*); consider that the chemical formula is obtained by chemical analysis; structural formula is the result of a number of physical observations (e.g. NMR, IR, UV spectroscopy) and chemical observations (relations of the unknown substance to other already known).
Using diffraction techniques one can establish size, shape, and symmetry of the unit cell. From the experimental density of crystals one obtains the atom content of the unit cell. Using the diffractometers one obtains an appropriate number of diffracted intensities.

6. Coordinates

Structures are commonly described by using the crystallographic coordinates (c.c.) to define atom positions. The c.c. (collectively indicated with p_i) are $3N$ if atoms are N. This choice is the most natural and also practical because the simple analytical relationship with the structure factors $F(h,k,\ell)$, see equation (1).

[4] Among deterministic methods we include the so called direct methods, actually the most used. Also direct methods are somewhat probabilistc and aleatory, in modest amount however. Also using direct methods the risk of finding false solutions exists.

As a matter of fact, infinite alternative systems for describing structures are conceivable: if new coordinates q_i are introduced, so that one can transform from p_i to q_i and viceversa *using analytic and biunivocal relationships*, the q_i can be used as alternative stuctural variables. The number of q_i must be, of course, again $3N$.[5]
Which are the alternative useful descriptions among the infinite possible ones? To reply it is important to point out that atomic positions are not fully impredictable. The systematic study of molecular crystals did demonstrate that atomic positions are not random quantities. Any structural hypothesis must obey to the following rules:

i) atoms must be appropriately separated among them; for each atom pair distance is anything but random: if the two atoms are chemically bound, atom separation is a bond-length (b-l) and must be close to the sum of the atomic radii (see e.g. Cotton, 1999); if not bound and spaced by three bonds or more, the distance must be higher than the sum of the so called van der Waals radii (Bondi, 1964); the same applies when the two atoms belong to different molecules (packing distances).

ii) the bond-angles (b-a) must be close to the "canonical" values prescribed by the rules of orbital hybridation (see e.g. Cotton, 1999): angles close to 109.5° on carbon atoms with sp^3 hybridation, close to 120° on carbon atoms with sp^2 hybridation, etc. The examination of "molecular models" suggest in much cases angles in rather restricted intervals.

iii) the molecular conformation must obey to the rules of stereochemistry: aromatic rings must have D_{6d} symmetry, the sequences C–C=C–C have to be planar, the torsion-angles (t-a) about single bonds must have values close to those typical of ethane-like molecules (-60, 60, 180°), etc. These limitations can be thus "rigid" in some cases (double bonds), "flexible" in other cases (single bonds).

The above restrictions for b-l, b-a, and t-a are of simple mathematical formulation if the crystallographic coordinates p_i are employed; there is however the disadvantage that the restrictions (*constraints*) apply *not to the p_i themselves, but to a number of mathematical functions of the p_i*. Indeed there are procedures for doing minimizations in presence of constraints: the Lagrange method (Goldstein, 1980) with the drawback that only a few constraints can be accounted for, while in molecular building there are numerous. Instead, introducing alternative coordinates q_i *chosen with cleverness*, it is possible to impose the above constraints not on the functions but directly on the q_i. If so happens one simply removes the q_i from the list of variables.

7. The internal coordinates. Eyring algorithm

It is customary studying complicate problem by means of the so called *internal coordinates* (i.c.), used in performing molecular building in various other contexts (e.g. spectroscopy, theoretical chemistry). The pioneer work (in chemistry) was that of Eyring (1932) who devised the procedure later called *Z-matrix* . Really the idea is well older; it was skilfully described by Lagrange (1796) who coined the term *generalized coordinates*. Lagrange studied not molecules but machines; things are rather similar after all: Lagrange's machines are made by rigid objects

[5] $3N$ coordinates are necessary for describing the *crystal structure*. If only the *molecular structure* is of interest, the coordinates are $3N - 6$ being six the rigid body coordinates (Goldstein 1980).

connected to each other with limited freedom of movement; in molecules the rigid objects are the chemical bonds and movements are limited to torsion and (in small amount) to bending.[6] Eyring's algorithm is simple, provided that there are neither rings nor polyhedra. The idea is to build molecules first defining three atoms, (`C1` , `C2` , and `C3` in Fig. 1) then inserting the other atoms stepwise as function of three i.c. in each step: a b-l b, a b-a τ and a t-a ϑ. Of course, in each step one must specify which atoms are to be considered.
Three i.c. are needed for starting a molecular building, viz. two b-l (b_1, b_2) and a b-a (τ_3). It is customary to put the 1st atom on the origin, the 2nd atom along x axis, the 3rd atom in the x, y plane with $y > 0$.
Fig. 1 shows how the building starts (`chbe` command, green atoms) and the Eyring construction (red atom). The construction of the 5th atom (blue) is discussed later.

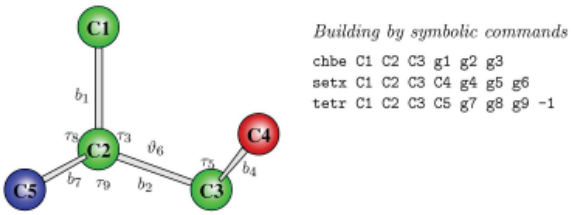


Fig. 1. Building a 5 atoms molecule

The Eyring's machinery for attaching a new atom after a given sequence A, B, C, is as follows: one computes the vector $\mathbf{v}$ with components $b\cos\tau$, $b\sin\tau\cos\vartheta$, $b\sin\tau\sin\vartheta$, the vector product $\mathbf{u} = (B - A) \times (C - B)$, and the orthogonal transformation matrix $\mathbf{T}$ aligning A and B points along z' and making $y'_C = x'_C = 0$; then adds to C atom the product $\mathbf{T} \cdot \mathbf{v}$. The analytical expression of $\mathbf{T}$ matrix is:

$$\mathbf{T} = \begin{vmatrix} x_B - x_A & x_C - x_B & u_x \\ y_B - y_A & y_C - y_B & u_y \\ z_B - z_A & z_C - z_B & u_z \end{vmatrix}$$

with each column normalized to length 1.
Besides the Eyring's construction (`setx`) needing three variables b, τ, ϑ, another construction is practical when atoms are inserted on tertiary carbon atoms *of known chirality*; this construction (`tetr`) makes use of two b-a τ_1, τ_2 (instead of one b-a and one t-a); the advantage is the restricted range of the b-a compared with the wide range of the t-a. The above 5-atom skeleton (Fig. 1) has been built just with 3 constructions: `chbe` (for atoms C1–C2–C3, in green), `setx` (for atom C4, in red), and `tetr` (for atom C5, in blue).
In Fig. 1 the three constructions are indicated "symbolically" according to a conventional syntax (see later).
Of course six other variables must be added to the molecular i.c. for defining the actual position and orientation of the molecule in the unit cell: three translations and three rotation angles. Rototranslation i.c. can be less than six because of crystal symmetry.

[6] This representation applies when bonds are considered rigid and bond-angles semirigid . This situation applies in studying molecular crystals (at a coarse level). In other contexts (e.g. spectroscopy) things are different.

7.1 Redundant coordinates

A serious drawback encountered in building molecules using Eyring method is the possible redundancy of the i.c. In simpler words *they might be not independent*. While independent i.c. can be varied each in turn preserving the *stuctural formula*, this does not apply in case of redundancy. When molecular building is performed within random search and genetic procedures, the preservation of the structural formula is of basic importance. Also regarding reduncancy there is nothing new after Lagrange; he said all the necessary: the *generalized coordinates* are exactly the same thing.

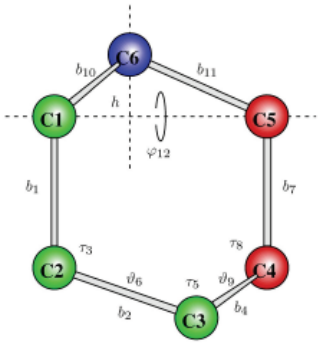
As illustrated below, non-redundancy in molecular building can be fulfilled by selecting the $3N-6$ i.c. of a molecule with shrewdness among b-l, b-a, t-a, and *also bending angles if necessary* (see later). Bending angles become necessary in two cases: cyclic molecules and polyhedric molecules.

7.2 Cyclic molecules

A possible solution overcoming redundancy in cyclic molecules is shown in Fig. 2 considering the case of cyclohexane.

Since the g_i must be, at a molecular level, $3N-6$, considering N bond-lengths, only other $2N-6$ g_i must be assigned, necessarily, angular. The method adopted (Immirzi, 2007a,b) integrates Eyring's procedure for building atoms C1, C2, ...C5 (9 g_i are employed altogether, 4 b-l, 3 b-a, and 2 t-a) with a new machinery for building atom C6 (proposed by Goto and Osawa, 1989) using two b-l and one angle only, a so called *bending angle* (φ); with this construction the molecule is bent about a line crossing two atoms separated by one bond only. Such construction (termed `flap`) has been added to the above ones. The machinery for computing C6 as a function of φ and the two b-l C1–C6 (g_{11}) and C5–C6 (g_{12}) is very simple (see Immirzi, 2005a).

The construction shown in Fig. 2, termed `flap` resolves the problem *for rings of any dimension* and also every polyciclic molecule (e.g. decaline, steroids, etc.) and resolves also intricate multicyclic molecules like norbornane, pinene, spirocompounds, etc.



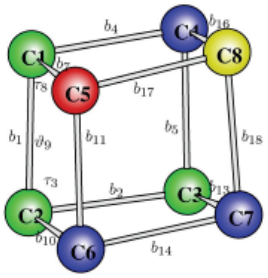
Building by symbolic commands

```
chbe C1 C2 C3 g1 g2 g3
setx C1 C2 C3 C4 g4 g5 g6
setx C2 C3 C4 C5 g7 g8 g9
flap C2 C1 C5 C6 g10 g11 g12
```

Fig. 2. Building a ring without redundancy, cyclohexane C_6H_6, (the blue atom is computed by a `flap` instruction).

7.3 Polyhedric molecules

A possible solution overcoming redundancy in polyhedric molecules is shown in Fig. 3 considering the simplest polyhedron: the cubane C_8H_8, represented as a bare carbon atom



Building by symbolic commands

```
chbe C1 C2 C3 g1 g2 g3
flap C2 C1 C3 C4 g4 g5 g6
setx C3 C2 C1 C5 g7 g8 g9
flap C1 C2 C5 C6 g10 g11 g12
flap C2 C3 C6 C7 g13 g14 g15
cage C4 C5 C7 C8 g16 g17 g18
```

Fig. 3. Building a polyhedric molecule without redundancy (cubane C_8H_8). The green atoms are computed by `chbe`, the red atom by `setx`, the blue atoms are computed by `flap`, and the yellow atom by `cage`.

skeleton. Remember that the g_i must be, at the molecular level, $3 \times 8 - 6 = 18$ and 12 g_i are b-l. In order to build without redundancy one must use *only 6 angles*. Building the first 7 atoms can be done as indicated symbolically in the figure. After the 5th command one atom is lacking, C8, and *all the 6 angles have been used* besides 9 over 12 bond-lengths. To complete the molecule one can find x_8, y_8, z_8 imposing C8 has assigned distances b16, b17, b18 from points C4, C5, C7. This is a simple (and classical) problem of 2nd degree having an unique solution provided that the polyhedron is convex. The command `cage` computes the lacking atom. Note that `cage` allows modeling also giant polyhedra like the fullerenes.

8. Other constructions and symbolic building

Molecular and crystal building through internal coordinates is not used only in performing structural analysis by EA. It can be used instead in all conventional procedures. In particular for doing interactive modeling, trial-and-error calculation, and at last for doing the least-square refinement of structures based on the i.c. themselves as optimized variables. In the latter case the non-redundancy of the variables plays a crucial role since only in absence of redundancy normal matrices are always non-singular.[7]

For these reasons all the mentioned constructions have been programmed devising an unique subroutine (termed `LAGR` , an homage to Lagrange, the authentic, and ignored, discoverer of the i.c.) performing the whole construction at each call. To simplify the input data preparation `LAGR` has been structured so that the various constructions invoked are specified symbolically according to a conventional syntax with one line of data for each construction step. Examples of this syntax are indicated in the figures and in the input data for procedure validation (Tables 1 and 2).

Besides the mentioned four basic constructions needed for building molecules of any kind (`chbe`, `setx`, `tetr`, `flap`, and `cage`, `tetr` is really not indispensable) there are numerous other constructions, not strictly necessary but very useful. Among them there are `phen` (building phenyl groups), `metl` (building a CH_3 group) and others. `LAGR` subroutine includes also the commands necessary for placing the built molecule in the unit cell, for doing orthogonal transforms, for referring the molecule to its inertial axes, and others. A clever

[7] A persuasive argument is that normal matrix needed for carrying out the least-square procedure is computed evaluating the first-order partial derivatives of structure factors vs. the variables. Such derivatives are computed wrong if the variables are not strictly independent to each other.

usage of these commands brings to a further reduction of the effective number of i.c. by imposition of *local symmetry*, i.e. the identity of structurally independent i.c. but chemically indistinguishable. At last a number of special commands were implemented for the case of linear polymers: to create regular helices and chains with glide-planes, to orient the chain parallel to the crystallographic *c* edge, etc.

9. General layout of a EA based procedure. Metaheuristic approach

The evolutionary (or genetic) algorithms for resolving global optimization problems are based on the idea of finding the "true" solution starting from a number of more or less arbitrary solutions and performing combinations of the variables analogous to the ones occurring in the cellular reproduction.
There is an analogy between chromosomes and *molecular structures*. As a chromosome can be described by a sequence of genes, a molecular structure can be described by a sequence of structural variables. For these reasons the terms are used as synonyms.
In order to limit the aleatority of the procedure it is practical to follow the so called *metaheuristic approach* (Weise, 2009) assuming that the g_i vary as multiple of a small increments. With this contrivance the i.c. (real and continuous quantities) become integer numbers whose upper limit is limited by the range of excursion of the g_i themselves. "Solutions" can binary-coded using a finite number of bits. Consider anyway that the number of points in the problem-space remains astronomic; for instance a 10-variable structure codified in 60-bits may assume $2^{60} \approx 10^{18}$ values. If, for simplicity, a 5-gene case is considered, and 7, 6, 3, 4, 6 bits are dedicated to the 5 genes respectively, the binary code will be (braces groups the bits of given g):

$$\underbrace{b_6b_5b_4b_3b_2b_1b_0}_{g_1}\quad\underbrace{b_5b_4b_3b_2b_1b_0}_{g_2}\quad\underbrace{b_2b_1b_0}_{g_3}\quad\underbrace{b_3b_2b_1b_0}_{g_4}\quad\underbrace{b_5b_4b_3b_2b_1b_0}_{g_5}$$

9.1 Planning the complete structural building

It is propedeutical to write-up a complete schema for the structural building based on the symbolic language.[8] This important step is anything but "automatic"; rather it requires much attention and cleverness. One must assign the non-redundant i.c. and prepare the full list of the symbolic building commands which, using the aforementioned subroutine `LAGR`, will provide the full structural building as function of the current i.c. This schema serves not only for carrying out the EA procedure, but for *all the numerous options of TRY program* (calculation of structure factor, model adjustment, crystal packing, least-square refinement, geometrical computations, etc.). Among the symbolic instructions there is one indispensable for carrying out the random search in the problem space, termed `xcon`, and discussed in details later.

9.2 Binary encoding

Now the user must: i) decide which g_i must be kept fixed and which varied (it is customary to exclude the bond-lengths and, possibly, also the bond-angles); ii) assign to each variable g_i a convenient increment Δg_i (the g_i will be varied by multiple of Δg_i), iii) assign to each

[8] These schemas are familiar to people using the Eyring's mechanism for molecular building. In the present case things are a little bit more difficult for the variety of constructions, which are necessary to avoid redundancy.

variable g_i how many bit (m_i) are to be used in binary encoding. Δg_i and m_i determine the range of excursion of the g_i (range it is exactly $\Delta g_i \times 2^{m_i}$).
It is evident that some common sense must be exerted in assigning Δg_i and m_i with a compromise between fine steps in exploring the problem-space and wide spanning intervals. For instance, for a bond angle on a carbon atom one could assign $\Delta = 1.0^\circ$ and $m_i = 4$; the angle will span a sufficient interval of 16°. Torsion angles (about single bonds), have instead higher uncertainity; giving them e.g. 5 bits and $\Delta g_i = 3^\circ$ will permit a span of 64°. A large number of bits, e.g. 7 or 8, must be assigned to the molecular rotation angles: using $m = 7$ and $\Delta = 2.8^\circ$ will ensure a 360° span; with $m = 8$ the same is fulfilled for finer step: 1.4°. Using common sense is no doubt necessary; but also experience turns out to be useful.
In this mechanism the initial value of each variable g_i° (to be defined in the input data) plays a role, since the values assumed by g_i are ($k = 1, 2 \ldots 2^m$)

$$g_i^\circ,\ g_i^\circ - \Delta g_i,\ g_i^\circ + \Delta g_i,\ g_i^\circ - 2\Delta g_i,\ g_i^\circ + 2\Delta g_i,\ g_i^\circ - 3\Delta g_i, \ldots$$

Note however that the g_i° are not critical quantities, provided they are internal to the span intervals.

10. Creating the initial population. Constraints

The first step of an evolutionary procedure is the creation of a *population of tentative structures* by means of a Montecarlo method (using e.g. pseudo-random numbers).[9]
One must assign the size of the population desired (e.g. 100-300 items) and the selection criteria (see below). Size and selection criteria control, in a very unforeseeable manner, the duration of the search, from few minutes to days. Anyway the search can be in any moment interrupted and restarted with new parameters, either creating a new list of solutions or queuing to the existing one.
If K is the number of g_j searched (K is of course well lower than $3N$), the random search is performed generating, for each tentative structure, K integer random numbers each in the range $0 \div 2^{m_j} - 1$ to obtain a random combination of g_i. By using the cited subroutine `LAGR`, the corresponding model is created. It is evident that most of these models will be unfeasible. There are two alternatives: i) to retain all solutions, attaching to each solution a proper "penalty function", ii) to apply some selection criteria and reject immediately unfeasible items. The latter has been chosen in our procedure to avoid endless populations. Much experience is anyway necessary for calibrating the rejection parameters, actually under user control.

10.1 Selection criteria. The `xcon` command

The computer program implemented in our laboratory includes the following selection criteria (*filters*):

1) the 1st filter, most incisive and mandatory, is the one checking the connectivity of the trial structure versus the known connectivity (remember that the chemical formula is supposed known!). We define as *connectivity* a group of eight integer codes (`connv`) to be inserted

[9] True random numbers should be preferable. Some hardware devices are produced, but we do not have direct experience. Such devices should be of course *very fast* as billions random numbers are necessary.

in the xcon command: the first is the number of atom-pairs separated by one bonds, the second the same number for two bonds, etc. up to 8 bonds.[10]

2) the 2nd filter eliminates structures with unfeasible bond angles. Non bonded pairs must have distances exceeding more than sepn2 (atoms separated by two bonds) or sepn3 (atoms separated by three or more bonds) the sum of covalent radii. Unrealistic structures with acute bond angles can so be eliminated. (In few cases, eg. cyclobutane, acute angles occur). The quantities sepn2 and sepn3 are supplied in the command xcon.

3) the 3rd filter checks the connections of the trial molecule with the neighbour (in molecular crystals no connections should be present; in linear polymer, instead, two connections take place); the number of external connections (linkno) is also assigned in xcon command. Connection are illegal if distance is lower than the sum of covalent radii augmented by tolnk, also supplied in xcon.

4) the 4th filter consists in computing the lattice energy, again rejection occurs if energy exceeds the assigned threshold, chosen by the user at the beginning of search.

5) the 5th filter is based on the R_2 index defined as the residual χ^2 (equation 2) divided by the sum of intensities. the tentative solution is rejected if R_2 exceeds an assigned value. Also for R_2 the threshold is chosen by the user.

Parameters sepn2, sepn3, tolnk, linkno and connv are defined in command xcon. Note that the same filters apply also in the subsequent *breeding* (see later).
It is evident the importance of assigning the above values *cum grano salis*, avoiding both excess or lack of severity. Indeed the ratio accepted-structures / generated-structures leans to be low (e.g. 10^{-5}); the conformational freedom of the molecule has of course a critical role.

10.2 Optimizing of the initial population

We have introduced (as an optional) the *local optimization* of the random solutions found with the above procedure. Each random solution is a point in the (discontinuous) problem-space. The idea is to examine the nearest points in the problem-space and find points possibly more promising, based e.g. on R_2 index. In a K-dimension problem space points are $3^K - 1$, a value prohibitively high if K is large.
On belief that the local optimizazion is a good idea (experience seems to confirm) we have exploited a trick: to assign a reasonable value to the number of neighbouring points P to be examined, say 500-1000, and select, at random, P points among the $3^K - 1$ ones.[11] With this contrivance the duration of an optimization step is reasonable.

11. Genetic combinations

The heart of the genetic procedure is the systematic combination (*breeding*) of the solutions belonging to the initial population following the rules of genetics. In this context two kinds of

[10] Establishing the connectivity codes is rather bothering, but the problem is easily resolved using TRY. One considers a dummy unit cell without symmetry and with large lattice constants. If the molecule is well constructed and displayed on the monitor the eight connection codes will appear on the top of the screen. Keep note and insert in xcon.

[11] For each integer p running from 1 to P one computes a real random number in the range 0-1, multiply it by 3^K and truncate. The point to be considered is given by the *digits of the product above written in the basis 3*. If desired 5 or 7 points can be considered instead 3.

combinations are considered: the *cross-over* and the *mutation*. A cross-over (*binary combination*) consists of an interchange of a selected gene belonging to the two mating items; a mutation (*unary combination*) is the change of a single bit (chosen at random) belonging to the bit-string. Considering a 9-gene example and a cross-over between genes g_2, g_7 one obtains:

Parent sequence (father):	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9
Parent sequence (mother):	g'_1	g'_2	g'_3	g'_4	g'_5	g'_6	g'_7	g'_8	g'_9
Child sequence:	g_1	g'_7	g_3	g_4	g_5	g_6	g'_2	g_8	g_9
Mutated sequence:	g_1	g'_7	g_3	g^*_4	g_5	g_6	g'_2	g_8	g_9

the symbol $*$ has been used for a mutation (the g_4 has been chosen by chance). In each case a new sequence is born for each crossover, two if also a mutation takes place. If the crossover takes place for all the pairs of K genes, $K \times (K+1)/2$ new strings occur; twice as many if also mutation is done.

Of course if two feasible structures are coupled, it is not warranted that child structures are meaningful. The above described "filtering" could be used again to eliminate unfeasible structures. At present the structures having R_2 higher than an assigned limit or a lattice energy higher than an assigned limit are eliminated. The limits, of course, should be fixed using experience. The new selected structures, anyway, are added to the initial ones.

The crossover can be done either considering a single gene, selected at random, or considering all genes; in the last case there are K combinations, for each pair of mated structures. When the genetic combination is ended the population of solution is extended by an amount depending on the used filters. The new population will be again sorted using an appropriate figure like the R_2 index.

Examining the sorted list one observes, frequently, that *the first solutions happen to be very similar to each other*. That is no doubt an indication of success. If this is not the case a good idea is to repeat the breeding as many times as it is necessary. If however the breeding has resulted ineffectual in finding new solutions, it is better to repeat the whole process in more appropriate conditions. The exploration of a wide fraction of the problem-space is no doubt essential for the success.

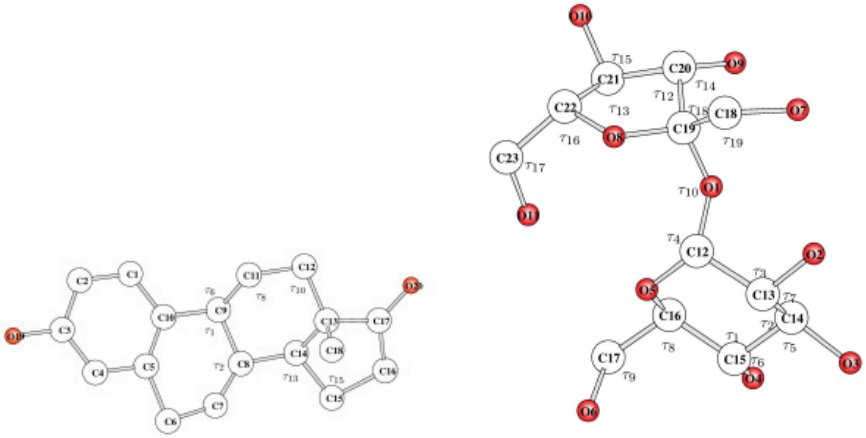
12. Procedure validation

For testing purposes two already reported structures (single-crystal X-ray diffraction studies, molecular models are shown in Fig. 4) have been considered using published data. Only the reflections having $d > 1.2$ Å have been considered to show that the method needs few reflections and that resolution can be modest.

The first test concerns the steroid equilin (a 20-atom molecule, ignoring hydrogen atoms) with restricted conformational freedom (Sawicki et al. 1999), the second test considers sucrose, a 23-atom molecule with high conformational freedom (Hynes et al., 1991). The latter was already considered in the first publication describing the method (Immirzi et al., 2008); the analysis has been repeated however in more challenging conditions. The input data for doing the two tests are given in Table 1 and Table 2; many comments are added for reader's convenience.

All computations were done using the program TRY (Immirzi, 2007) and assuming fixed bond-lengths in both cases, also fixed bond-angles in the case of equilin. The computing scheme is the simplest one: no hydrogen atoms, isotropic thermal vibration (an unique B_{iso}), unitary weight factors. In both cases the EA procedure consists of three phases: i) formation of

the initial population, ii) improvement of solutions looking at adjacent points, iii) breeding by cross-over and mutation. The reliability of the resulting structures has been tested performing some cycles of full-matrix least-square refinement optimizing the same i.c. considered in the EA phase. The resulting R indices are, of course, not competitive with the ones published because of the crude simplicity of the followed computation scheme and because of the reduced number of reflections. The LS convergency is anyway excellent.



(a) Equilin (b) Sucrose

Fig. 4. Molecular models of the two test molecules: equilin and sucrose. Bond angles (τ_n) are shown. Torsion angles (ϑ_n) are listed in Tables 1 and 3.

The following scheme summarizes the essential data for the two tests.

	equilin	sucrose
reflections considered ($d > 1.2$ Å)	715	589
number of non-hydrogen atoms	20	23
number of internal coordinates	60	68
varied internal coordinates	15	28
bits used in binary encoding	75	123
random structures generated	50×10^3	141×10^6
initial population	300	150
lowest R_2 index after search	0.43	0.74
lowest packing energy	-3Kcal	-11Kcal
search duration	3 min.	29 hr.
mated structures	20	30
resulting R_2 after breeding	0.39	0.43

Note how the formation of initial population is performed in a very short time in the former case (75-bit encoding), and longer time in the latter (123 bit encoding).

```
Equilin by Sawicki 1999
6.5429  9.0345  23.894  0  0  0  !  lattice constants
C=C  O=O                          !  atomic species
4 0.72080 56.55 4  4  6           !  3 non-operative codes + crystal extension
-x+1/2, -y, z+1/2,                !  symmetry (P2_12_12_1 space group)
-x, y+1/2, -z+1/2, *
!  here begin lines processed by LAGR subroutine
xcon  0.10 0.10 0.10 0  23  36  39  30  22  16  12  7
!  1st ring (aromatic)  C1-C2-C3-C4-C5-C10 (conventional geometry)
chbe  C1  C2  C3   1.40  1.40  1.20
setx  C1  C2  C3   C4    1.40  1.20  0.0
setx  C2  C3  C4   C5    1.40  1.20  0.0
setx  C3  C4  C5   C10   1.40  1.20  0.0
!  2nd ring (cyclohexene) C6-C7-C8-C9
setx  C2  C1  C10  C9   1.52 1.20 1.80
setx  C1  C10 C9   C8   1.52 g1  g2
setx  C10 C9  C8   C7   1.33 g3  g4
flap  C8  C7  C5   C6   1.52 1.52 g5
!  3rd ring:  (cyclohexane)
setx  C1  C10 C9   C11  1.54 g6   g7
setx  C10 C9  C11  C12  1.54 g8   g9
setx  C9  C11 C12  C13  1.54 g10  g11
flap  C9  C8  C13  C14  1.54 1.49 g12
!  4th ring (cyclopentane)
setx  C9  C8  C14  C15  1.54 g13  g14
setx  C8  C14 C15  C16  1.54 g15  g16
flap  C15 C16 C13  C17  1.52 1.52 g17
tert  C12 C14 C17  C13  C18  1.54
!  O atoms
setx  C5  C4  C3   O19  1.36 1.20  1.80
trig  C13 C17 C16  O20  1.22 0
iner  C1  Q31  20   0   !  refer the 20-atom molecule to inertial axes
rtax  1   C1  20  g18   !  Rx rotation
rtax  2   C1  20  g19   !  Ry rotation
rtax  3   C1  20  g20   !  Rz rotation
move  C1  20  g21 g22 g23  0  !  translation
end
1.0800     1.6900     1.2000     0.0800     1.7000  !  starting values
1.0900    -0.6400     1.0900     1.7800     1.0600  !  for i.c.
0.5450     1.2700     1.2200     1.7700     1.0200  !
1.6600     1.7200    -0.9521    -0.5334    -0.5628
2.0146     3.5320     9.8812
*
*
180 -90 180  50  1.0 1  !  screen projection and scale
5.2  5.2  !  B-iso      !  thermal parameters
0.0  0.0  !  B-33       !  anisotropic component of ditto
!  the following 15 lines define the g(i) kept variable in EA
!  and the binary-encoding scheme:  no.  of bit and step; span intervals
!  (sp) are also given as comments
 2  4  0.028  0  0  !  sp for C1-C10-C9-C8   t-a =45deg
 4  4  0.028  0  0  !  sp for C10-C9-C8-C7   t-a =45deg
 5  4  0.028  0  0  !  sp for C8-C7-C5-C6    t-a =45deg
 7  4  0.028  0  0  !  sp for C1-C10-C9-C11  t-a =45deg
 9  4  0.028  0  0  !  sp for C10-C9-C11-C12 t-a =45deg
11  4  0.028  0  0  !  sp for C9-C11-C12-C13 t-a = 45deg
14  4  0.028  0  0  !  sp for C9-C8-C14-C15  t-a = 45deg
16  4  0.028  0  0  !  sp for C8-C14-C15-C16 t-a = 45deg
17  4  0.028  0  0  !  sp for C15-C16-C13-C17 flap =45deg
18  7  0.028  0  0  !  sp for Rx rotation = 360deg
19  7  0.028  0  0  !  sp for Ry rotation = 360deg
20  7  0.014  0  0  !  sp for Rz rotation = 360deg
21  5  0.200  0  0  !  sp for Tx translation = 6 A
22  6  0.200  0  0  !  sp for Ty translation = 12 A
23  7  0.200  0  0  * !  sp for Tz transl.  = 25 A
0 1.0 10.0 0.1 1.0 1.0 1.0 1.0 0.8 1.0 0.0
EQUILIN.DAT ! file containing the F2hkl
```

Table 1. Input data for equilin test

```
Sucrose by Hynes (1991)
10.8631  8.7044  7.7624  90 102.94 90  !  lattice constants
C=C  O=O  H=H                          !  atomic species
8 1.5418 56.55  4  4  4                !  3 dummy codes and cell extension
-x, 1/2+y, -z, *                       !  symmetry (P2_1 space group)
!  here begin lines processed by LAGR subroutine
xcon  0.30  0.30 0.40  0 24 36 43 36 32 30 26 17 !  connections
chbe C16  C15  C14       1.521   1.521   g1   !  build piranose
setx C16  C15  C14  C13  1.521   g2      g21
setx C15  C14  C13  C12  1.521   g3      g22
flap C15  C16  C12  O5   1.427   1.427   g23
setx C16  O5   C12  O1   1.427   g4      g24  !  O atom bridge to furanose
setx C16  C15  C14  O3   1.427   g5      g25  !  O3
setx C13  C14  C15  O4   1.427   g6      g26  !  O4
setx C15  C14  C13  O2   1.427   g7      g27  !  O2
setx C14  C15  C16  C17  1.521   g8      g28  !  lateral CH2OH group
setx C15  C16  C17  O6   1.427   g9      g29
setx C13  C12  O1   C19  1.427   g10     g30  !  build furanose
setx C12  O1   C19  C20  1.521   g11     g31
setx O1   C19  C20  C21  1.521   g12     g32
setx C19  C20  C21  C22  1.521   g13     g33
flap C21  C22  O19  O8   1.427   1.411   g34
setx O8   C19  C20  O9   1.427   g14     g35  !  O lateral to furanose
setx C19  C20  C21  O10  1.427   g15     g36  !  O lateral to furanose
setx C19  O8   C22  C23  1.521   g16     g37  !  lateral CH2OH
setx O8   C22  C23  O11  1.427   g17     g38
setx C21  C20  C19  C18  1.521   g18     g39  !  lateral CH2O
setx C20  C19  C18  O7   1.427   g19     g40
iner C1   Q50  23   0  !  compute the inertial axes and orient accordingly
rtax 1    C1   23   g41 !  overall Rx rotation
rtax 2    C1   23   g42 !  overall Ry rotation
rtax 3    C1   23   g43 !  overall Rz rotation
move C1 23 g44 0 g45 0  !  translation
end                                    !  i.c.  starting values follow
   1.1072   1.0725  1.1207   1.1010  1.0771   1.1249   1.0982   1.1219   1.1124
   1.1381   1.0818  1.0225   1.0285  1.1583   1.1181   1.0999   1.1284   1.1479
   1.1137   0.0000 -0.5680   0.5729  1.2994  -0.6819 -1.7724  -1.7409   1.7851
   1.7344 -0.6413   1.3017   1.6018 -0.8762  -0.3520  1.7662   1.5806  -1.5559
  -1.3286  0.7047   1.4875   0.7208  0.7006  -0.1187   0.5964  2.4590    3.1670
*
-90  0  0  50  1.0  1    !  molecular orientation, and image scale factor
1.3  1.3  1.3            !  Isotropic thermal parameters
0.0  0.0  0.0            !  anisotropic component of ditto
!  binary-encoding scheme for g(i):  no.  of bit and step; span intervals  (sp) after !
4  3  0.00500 0  0  !  sp for O5-C12-O1  b-a = 4 deg
10  3  0.00500 0  0  !  sp for C12-O1-C19 b-a = 4 deg
11  3  0.00500 0  0  !  sp for O1-C19-C20 b-a = 4 deg
21  3  0.02812 0  0  !  sp for C16-C15-C14-C13 t-a = 22.4 deg
22  3  0.02812 0  0  !  sp for C15-C14-C13-C12 t-a = 22.4 deg
23  3  0.02812 0  0  !  sp for C15-C16-C12-O5  flap = 22.4 deg
24  3  0.02812 0  0  !  sp for C16-O5-C12-O1   t-a = 22.4 deg
25  3  0.02812 0  0  !  sp for C16-C15-C14-O3  t-a = 22.4 deg
26  3  0.02812 0  0  !  sp for C13-C14-C15-O4  t-a = 22.4 deg
27  3  0.02812 0  0  !  sp for C15-C14-C13-O2  t-a = 22.4 deg
28  4  0.02812 0  0  !  sp for C14-C15-C16-C17 t-a = 44.8 deg
29  4  0.02812 0  0  !  sp for C15-C16-C17-O6  t-a = 44.8 deg
30  7  0.02812 0  0  !  sp for C13-C12-O1-C19  t-a = 358 deg
31  7  0.02812 0  0  !  sp for C12-O1-C19-C20  t-a = 358 deg
32  7  0.02812 0  0  !  sp for O1-C19-C20-C21  t-a = 358 deg
33  3  0.02812 0  0  !  sp for C19-C20-C21-C22 t-a = 22.4 deg
34  3  0.02812 0  0  !  sp for C21-C22-C19-O8 flap = 22.4 deg
35  4  0.02812 0  0  !  sp for O8-C19-C20-O9   t-a = 44  deg
36  4  0.02812 0  0  !  sp for C19-C20-C21-O10 t-a = 44  deg
37  4  0.02812 0  0  !  sp for C19-O8-C22-C23  t-a = 44  deg
38  4  0.02812 0  0  !  sp for O8-C22-C23-O11  t-a = 44  deg
39  4  0.02812 0  0  !  sp for C21-C20-C19-C18 t-a = 44  deg
40  4  0.02812 0  0  !  sp for C20-C19-C18-O7  t-a = 44  deg
41  8  0.01400 0  0  !  sp for Rx rotation = 358  deg
42  8  0.01400 0  0  !  sp for Ry rotation = 358  deg
43  8  0.01400 0  0  !  sp for Rz rotation = 358  deg
44  5  0.17000 0  0  !  sp for Tx translation = 5.5 A
45  5  0.13600 0  0  *  !  sp for Tz translation = 4.4 A
0 3.0 10.00 0.25 0.80 0.20 0.10 0.70 0.005 0.0 0.0 !  powder data
SUC1.dat !  file contaning F2hkl
```

Table 2. Input data for sucrose test

13. Conclusions

The power of evolutionary algorithms in resolving difficult crystal structures from diffraction data has been discussed evidencing the convenience of basing the approach on internal coordinates. A specific procedure has been implemented, having the following main features: i) it is based on internal coordinates (this reduces considerably the number of variables and their uncertainty); ii) it uses discretized coordinates and binary structure encoding (metaheuristic approach); iii) the procedure is designed as a constraint satisfaction problem so incorporating the numerous a priori information available; the constraints are considerd both in the initial step of the procedure (formation of the initial population of solutions based on Montecarlo methods) and in the subsequent steps (breeding of the population).
The ideas above afforded a Fortran-language computer program suitable for any kind of molecular structure and available free of charge. The evolutionary procedure has been inserted into a general-purpose program, entirely based on internal coordinates. The program, presently tested on known structures (in difficult conditions), runs well. It has however the disadvantage of needing numerous parameters (presently assigned by the user) which could be assigned automatically in more evolute versions of the program.

14. References

Arnott, S. & Wonacott, A. J. (1966). *Polymers*, 7, 157-166

Back, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategy, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, USA.

Bondi, A. J. (1964). *J. Phys. Chem.* 68, 441.

Cotton, A., Wilkinson G., Murillo, C.A., & Bochmann, M. (1999). *Advanced Inorganic Chemistry*, 6th Ed. J. Wiley & Sons, New York.

Davis, L (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold.

Eyring, H. (1932) *Phys. Rev. 39*, 746-748.

Feng, Z. J. & Dong, C. (2006) *J. Appl. Crystallogr.*, 39, 615-617.

Giacovazzo, C. (2002). *Fundamentals of Crystallography*, Oxford University Press, Oxford.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Berkeley.

Goldstein, H. (1980) *Classical Mechanics* 2nd Ed. Addison-Wesley, Berkeley.

Goto, H, & Osawa, E. (1989) *J. Am. Chem. Soc., 111*, 8950-8951.

Hanson, A. J., Cheung, E. J. & Harris, K. D. M. (2007) *J. Phys. Chem. B*, 111, 6349-6356.

Harris, K. D. M., Johnston, R. L. & Kariuki, B. M. (1998) *Acta Crystallogr.*, A54, 632-645.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems: an introductory analysis with applications to Biology, Control, and Artificial Intelligence*, MIT press, USA.

Hynes, R. C., Le Page J. (1991). *J. Appl. Cryst.* 24, 352-354.

Ionita, M., Breaban, N., & Croitoru C. (2010) *Evolutionary Computation in Constraint Satisfaction* in *New Achievements in Evolutionary Computing*, edt. P. Kovosec, Intech edition, Vukovar, Croatia.

Immirzi, A. (2007) (*a*). *Chem. Inf. Model*, 47, 2263-2265.

Immirzi, A. (2007) (*b*) *J. Appl. Cryst.*, 40, 1044-1049

Immirzi, A., Erra, L., & Tedesco, C. (2008). *J. Appl. Cyrst.* 41, 784-790.

Immirzi, A., L. Erra, & Tedesco, C. (2009) *J. Appl. Cryst.*, 42, 810-814.

Lagrange, J. L. (1797). *Théorie des Fonctions Analytiques*, Imprimerie de la République, Paris.

Oganov, A. R., & Glass, C. W. (2006). *J. Chem. Phys.*, 124, art. 244704.
Okada, K., Noguchi, K., Okuyama, K., & Arnott, S. (2003). *Comput. Biol. Chem.*, 265-285
Polya, G. (1971). *How to solve it*. Princeton University Press.
Rietveld, H. M. (1967). *Acta Crystallogr.* 22, 151.
Rietveld, H. M. (1969). *J. Appl. Crystallogr.* 2, 65.
Sawicki, M. W., Li, N., & Gosh. D. (1999). *Acta Cryst. C*, C55, 1999, 425-427.
Smith, P. J. C. & Arnott, S. (1978). *Acta Cryst.*, A34, 3-11.
Stout, G. H., & Jensen, L. H. (1989). *X-ray structure determination, a practical guide*. J. Wiley & Sons.
Tadokoro, H. (1979). *Structure of Crystalline Polymers* J. Wiley & Sons, New York.
Weise, T. (2009). *Evolutionary Algorithms. Theory and Application*. URL : http://www.it-weise.de
Young, R. A. (1995) *The Rietveld Method*, IuCr Monograph on Crystallography, no. 5

29

Evolutionary Enhanced Level Set Method for Structural Topology Optimization

Haipeng Jia[1], Chundong Jiang[1], Lihui Du[2], Bo Liu[1] and Chunbo Jiang[2]
[1]*Department of Mechanical Engineering, Hebei University of Technology, Tianjin 300130*
[2]*State Key Laboratory of Hydroscience and Engineering of, Tsinghua University, Beijin 100084, P. R. China*

1. Introduction

During the last 20 years, structural optimization has become one of the most important topics of engineering applications. Design optimization of structure has been an interesting area of research in the field of engineering design for its ability to short the design cycle and to enhance product quality. Significant research activity has occurred in the area of structural optimization in the last decade. Especially for topology optimization of structure, many new theoretical, algorithmic, and computational contributions have resulted by researchers and engineers. Topology optimization is a powerful tool for global and multi-scale design of macrostructures, microstructures, and the cell of prescribed composite materials.

The population based evolutionary algorithms have emerged as powerful mechanism for finding optimum solutions of complex optimization problems in engineering during the last two decades. Evolutionary computation is the study of computational systems which use ideas and get inspiration from natural evolution and adaptation [1]. The thinking has wide application in various engineering fields, such as computer science, artificial intelligence, operations research. Genetic algorithm is another kind of bio-inspired optimization method and it is playing an increasingly important role in studies of complex adaptive systems. Its application ranges from adaptive agents in economic theory to the use of machine learning techniques in the design of complex devices and structures, such as aircraft turbines and integrated circuits [2].

Optimization of structures can be classified into three categories: sizing, shaping, and topology optimization. In the topology optimization, it is concerned with the structure members and connectivity between members. In general, it is easily represented by discrete variables rather than by those used for continuous optimization problems. Topology optimization is the most difficult and complex among three categories and it is special useful in developing innovative conceptual designs. Structural optimization, in particular the topology optimization, has been identified as one of the most challenging tasks in structural design. Various techniques and approaches have been established during the last two decades. Topology optimization usually referred to as layout optimization or general shape optimization [3]. It lets engineers get the optimal topology of structure or new configurations during product design phase, as they are implementing the design of the size

and shape of structure. In the last two decades, topology optimization has been becoming increasingly popular in industrial applications [4-6]. For in many cases, tremendous cost savings have been achieved due to the impact of this design tool in the early stage of the design procedure. However, because of the complicacy of the mathematical formulation and the difficulties in solving it, topology optimization is considered as one of the most challenging research field.

In order to improve the efficiency in global optimization search the topology of engineering problems, many heuristic algorithms [7, 8, 9] have been developed, such as evolutionary algorithm [8, 9], genetic algorithms[10, 11], ant algorithm, simulated annealing algorithm,. In recent years, many biologically inspired methods come to be used in topology optimization of structure. Evolutionary algorithms are a popular and robust strategy for structure optimization. Especially the evolutionary structure optimization (ESO) has been applied widely in solving structural topology optimization problems [12, 13, 14]. These methods have special characteristics such as parallel computing and globally optimum searching. Based on hole image interpretation techniques, Lin et al. [7] gave two-stage artificial neural networks for topology and shape optimization, which contains improved template variety and recognition reliability. Salami and Hendtlass [8] proposed a "fast evolutionary algorithm" that does not evaluate all new individuals, in which fitness and associated reliability value are assigned to each new individual that is evaluated using the true fitness function only if the reliability value is below a threshold. In 1992, Xie and Steven [9, 12, 13] proposed the ESO and bidirectional ESO (BESO) approach [13] for topology optimization and applied to the optimization of structures successfully. During implementation of the method, elements are gradually removed from the structure by altering the material properties. The removal or additional criterion is based on the comparison of the Von Misses stress, principal stress or the deformation energy of the candidate element. Considering the low efficiency of the usage of the material in the low value of stress or strain energy, the element can be removed from the structure, or added in high value region for the need of material. For its simplicity in implementation and convenience in coping with the local buckling, displacement constraint and local stress constraint, ESO algorithm has wide applications in the dynamic modification, topology optimization thermal-structure coupling problems with different criteria, for further please see refernces [14, 15, 16]. Mariano Victoria etc. [17] gives the isolines topology design algorithm, and in essence it is a variant of Evolutionary Structure Topology optimization approach. Based on the thinking of perfect state of harmony in musical processing, Lee and Z. W. Geem [18] give the implementation of harmony searching algorithm for structural optimization. The merit is independent of the initializing design variable and derivative information of objective and constraint function.

To get good topology of the final structure, in 1994 Eschenauer, et al. [19] put forward the Bubble Method. The main idea is first to introduce a new small circle and then implement a shape optimization by a conventional fixed topology shape optimization to get the size, shape and position of the hole. Osher and Sethian [20, 21, 23] proposed the concept of level set, it has been proven to be phenomenally successful as a numerical device, and since its appearance it has wide applications ranging from capturing multiphase fluid dynamical flows to special effects in hollywood to visualization, image processing, topology optimization of structure[24, 25], computer vision and many more. Wang and his coauthors [26, 27] Proposes level set method for structural topology optimization and many other variant of this algorithm. Wei [27] proposed piecewise constant level set method to nucleate

holes during optimization and some benchmark problems show the validity of the algorithm.

There are several advantages to this approach for topology optimization of continuum structures. Firstly, we solve the elastic strain equations on a fixed grid through a version of the immersed interface method (IIM), which avoids the complications that come from using distorted and convoluted unstructured meshes. Secondly, the Level Set Method allows us to perturb the shapes of the interface, without worrying about changes in topology, such as how many holes are required. Thirdly, the entire method carries over to three dimensions, if desired. This method attracts many interests of engineers and researchers in optimization field for the smoothness of the boundary and having no intermediate density in the final topology [24-30].

The studies all aimed at developing a robust and efficient algorithm for searching global optimum solution for engineering applications.

The demanding computational cost for engineering optimization is often very high for each iteration needs at least one finite element analysis. Because the finite element analysis for engineering model takes lots of time in finding required data for calculating the parameters of objective function in optimization problem and that of constraint function. Various mathematical programming methods have been used to solve engineering optimization problems. But these methods need calculation of the first or second order differentiation that will increase the difficulty in searching optimum solution. In another hand, the mathematical programming methods are easily to fall into local optimum for non-convexity of topology optimization problems.

The Traditional level set method algorithms for topology optimization use a Hamilton Jacobi equation to connect the evolution of the scalar function with the boundary of the topology contours. For this reason, it can hardly create new holes during evolving otherwise other measures has been taken.

This paper proposes an improved LSM algorithm. The newly modified method integrates the ESO inspired hole-inserting technique in LSM method and overcome the shortcomings of traditionally approach. Using this algorithm, new holes can be inserted at different positions during the optimization to determine the optimal topology. From the point of view of "ground structure", the proposed method of topology optimization enlarged the searching space of Level Set Method.

2. Mathematical formulation of the evolutionary optimization algorithm

Traditionally, the problem of topology optimization of structure to maximize stiffness can be specified as (1)-(4):

Minimize: $$J(u) = \int_{\Omega} F(u) d\Omega \tag{1}$$

s.t.: $$a(u, v) = L(v) \tag{2}$$

$$u|_{\Gamma_d} = u_0 \forall v \in U \tag{3}$$

$$V = \int d\Omega \leq V_{\max} \tag{4}$$

Here, the design domain of the structure is represented byΩ, $J(u)$ is the objective function, $F(u)$ is specific physical or geometric type on design domain. In this paper, $F(u)$ is the compliance of structure and the objective is to find the minimum of it, let the structure be the stiffest. In terms of the energy bilinear form $a(u,v,\phi)$, $L(v)$ and $\varepsilon_{ij}(u)$ described by (5)-(7) respectively,

$$a(u,v,\phi)=\int_{\Omega}E_{ijkl}\varepsilon_{ij}(u)\varepsilon_{kl}(v)d\Omega\,, \tag{5}$$

$$L(v)=\int_{\Omega}pvd\Omega+\int_{\Gamma}\tau vds\,, \tag{6}$$

$$\varepsilon_{ij}(u)=\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right). \tag{7}$$

The purpose of the topology optimization is to optimize the objective function by layout of the material in design domain.

In the Level Set Method, the boundary of structure is described by zero level set and it can easily represent complicated surface shapes that can form holes, split to form multiple boundaries, or merge with other boundaries to form a single surface. Zero level sets are decided by the objective function such as energy of deformation, stress, eigenvalue etc., and the optimal structure can be gotten through the movement, amalgamation of the external boundary of the structure.

Compared with the homogenization method and SIMP (Solid Isotropic Material Penalty) [30] method, the LSM has some excellent aspects: no chessboard, no mesh-dependency problems, and good numerical stability.

The LSM describes the topology of structure implicitly, and the course of the topology optimization of continuum is achieved by solving the Hamilton-Jacobi equation (8).

$$\frac{\partial\phi}{\partial t}+\vec{V}\cdot\phi=0 \tag{8}$$

Here, according to objective function how to descend, $\vec{V}$ is chosen to let level set function change. Time variable is length that satisfies Courant-Friedrichs-Levy (CFL) condition which makes difference calculation stability. The model of optimization can be specified as (9)-(12):

Min: $$J(u,\phi)=\int_{\Omega}F(u)H(\phi)d\Omega \tag{9}$$

s.t.: $$a(u,v,\phi)=L(v,\phi) \tag{10}$$

$$u|_{\Gamma_d}=u_0\forall v\in U \tag{11}$$

$$V=\int H(\phi)d\Omega\le V_{\max} \tag{12}$$

In terms of the energy bilinear form $a(u,v,\phi)$, the load linear form $L(v,\phi)$, and the volume $V(\phi)$ of the structure, respectively described by (13)-(15):

$$a(u,v,\phi)=\int_{\Omega}E_{ijkl}\varepsilon_{ij}(u)\varepsilon_{kl}(v)H(\phi)d\Omega \tag{13}$$

$$L(v,\phi)=\int_{\Omega} pvH(\phi)d\Omega+\int_{\Omega}\tau vH\left|\nabla\phi\right|\delta(\phi)d\Omega \tag{14}$$

$$V(\phi)=\int_{\Omega}H(\phi)d\Omega \tag{15}$$

Where $\delta(x)$ is Dirichlet function， and $H(x)$ Heaviside function, see paper [22].

As known to all, only the moving and merging of holes can be implemented during the LSM topology optimization, no new holes can be generated through the optimization. The disadvantage of level set based topology optimization is apparent for some engineering problems. To conquer the difficulty, one method is to initialize the guess design with enough holes in order to include as more topologies as possible. To get a good result, we should comply with the following two fundamental principles to initialize the guess configuration before carrying out the optimization:

a. The number of holes must be enough to include all the possible topology;
b. The layout of the holes should be rationally positioned.

Cantilever beam is a benchmark problem in topology optimization. As shown in Figure 1, it has a length of 64mm and a height of 40mm, thickness of the plate t =1 and is subjected to a concentrated load of 80N at the middle of its free end. The objective function of the problem is the strain energy of the structure with a material volume constraint. The Young's Modulus and Poisson's Ratio of the material used in the example are 200GPa and 0.3, respectively. Parameters $\alpha=10^{-9}$, $\Delta=1.0$ are used in the numerical approximation of $\delta(x)$ and $H(x)$. The volume ratio is limited to 25%. A mesh including 64 x 40 4-node-isoparametric elements is used, and the problem is dealt with as a plane stress problem. As shown in Figure 2, the guess topology configuration is initialized with 4 x 6 holes in level set function.

Figure 3 gives the topology evolving procedure during optimization progress. The final result in Figure 3(f) shows that good topology can be obtained if the initialized configuration includes sufficient number of holes, see in Figure 2. It consists with the result in paper [31] in Figure 5 and that from the optimization criteria(OC) approach in Figure 6. For more detailed description on the theory and numerical computation of the level set based topology optimization, see [26].

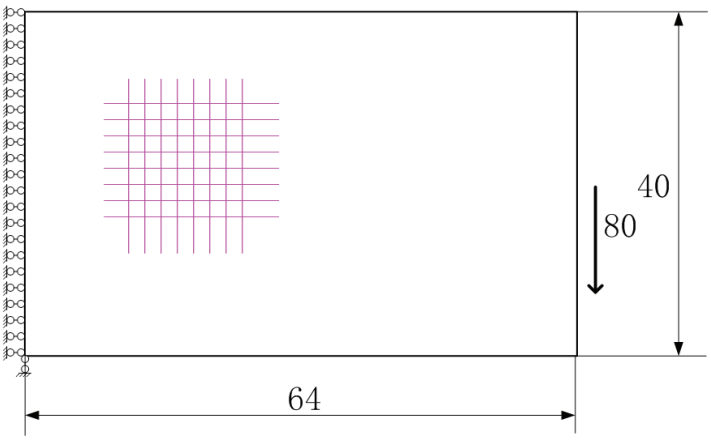


Fig. 1. Geometry parameters and boundary conditions of cantilever plate

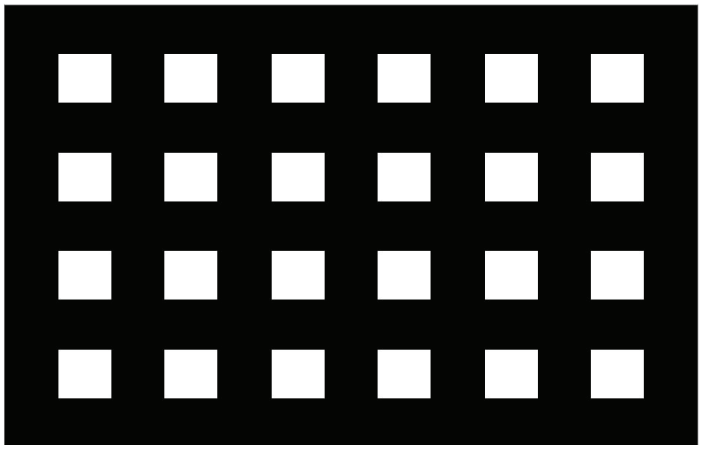


Fig. 2. Topology Initialization with evenly distributed 4X6 holes

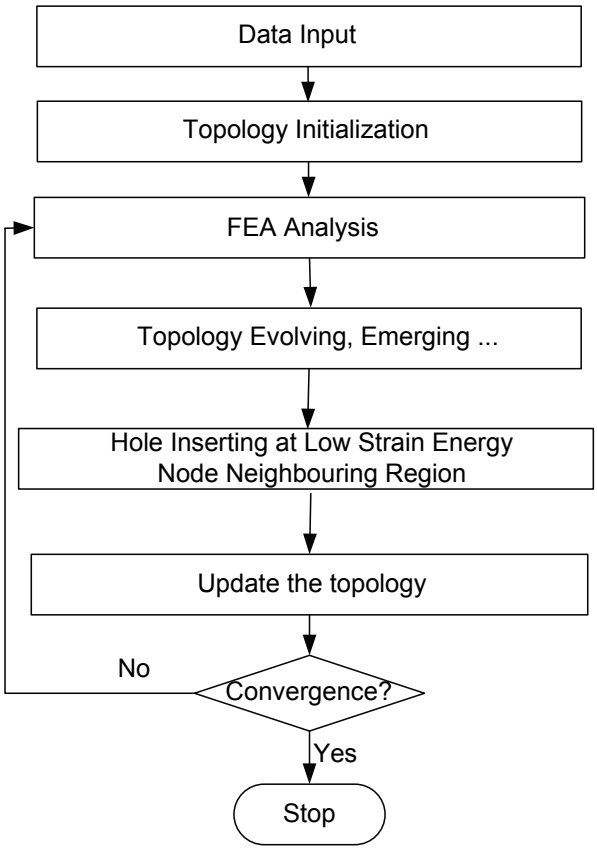


Fig. 3. Computational flow of the structural topology optimization

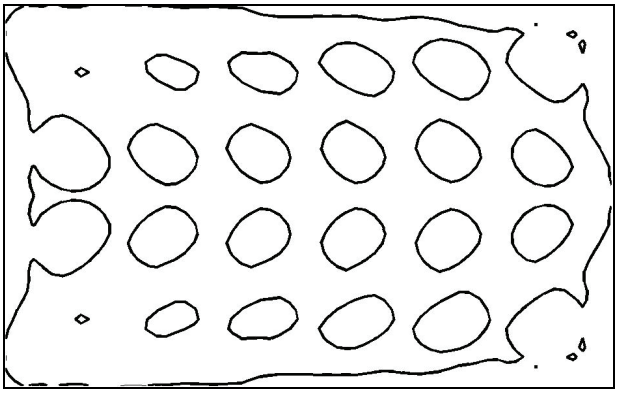


Fig. 4(a) Iteration number 15

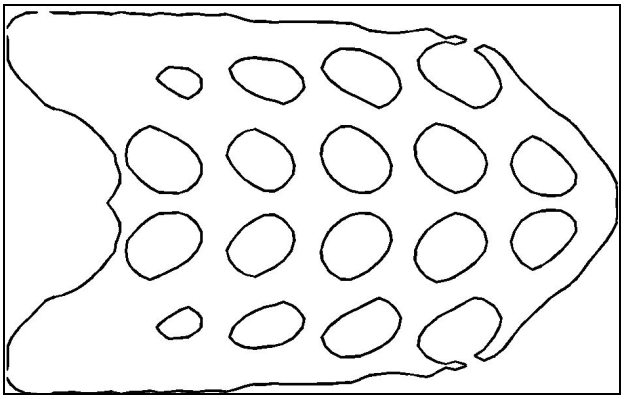


Fig. 4(b) Iteration number 30

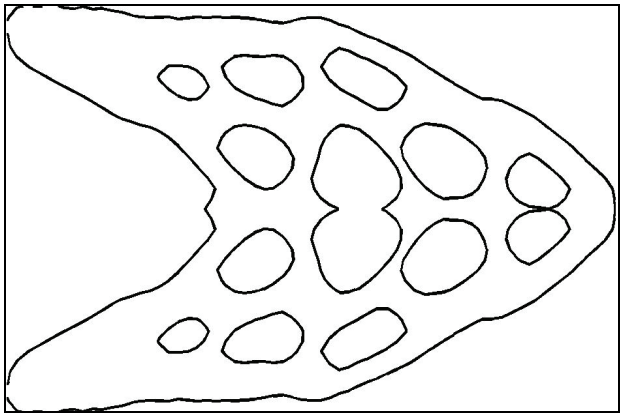


Fig. 4(c) Iteration number 45

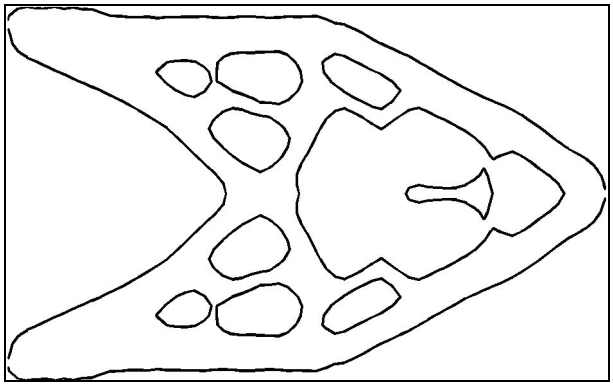


Fig. 4(d) Iteration number 60

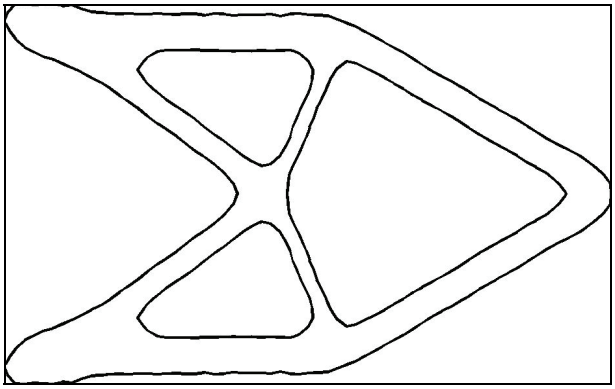


Fig. 4(e) Iteration number 75

Fig. 4. Traditional Level set method for topology optimization, initialized with uniformly distributed holes

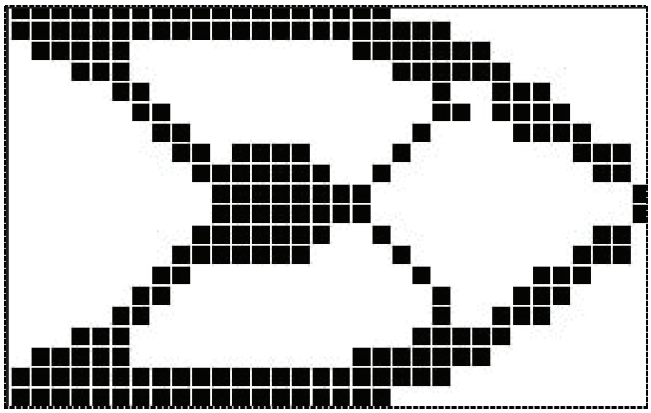


Fig. 5. Resultant topology in paper [32]

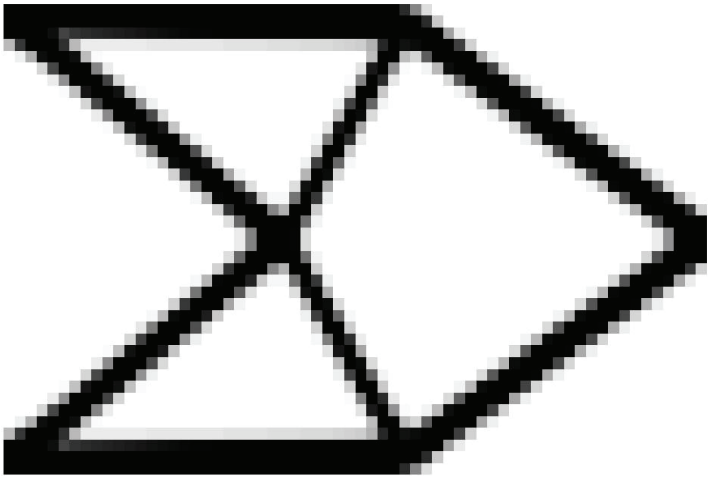


Fig. 6. Final topology through optimization criteria (OC) method

Numerical experiments show that the optimal topology depends on the initialization considerably. In fact, the final topology is only a subset of the candidate topology set of initialization. The more the topologies are included, the higher the possibility a good design can be obtained.

To illustrate the invalidity of the level set based topology optimization algorithm, let us design the topology of a cantilevered plate, a classical benchmark problem for topology optimization, from an initial guess topology with no hole. The design result indicates that the optimal topology is a two-bar-truss-like structure, which is apparently different from the real optimum topology. From this example, we can safely come to the conclusion that the optimal topology highly depends on the initial guess design, and that LSM can only find a best topology in the given topology sets in advance.

To circumvent the obstacle of independence of initialization, new criteria are needed to insert new holes at the right position during the right iteration. This is the emphasis of this paper.

The proposed method is based on the node neighboring strain energy as illustrated in Figure 7, and calculated through (16),

$$\alpha_i = \int_{\Omega^e} E_{ijkl}\varepsilon_{ij}(u)\varepsilon_{kl}(v)d\Omega \tag{16}$$

In (16), Ω^e indicate the node neighboring region as shown in Figure 8, and α_i is the Performance Index of the ***i-***th node relative to the whole structure, this value indicate the effect of the i-th node on strain energy of whole structure when it is removed from structure. For each iteration of optimization, the algorithm finds small percentage of the lowest strain energy of all nodes within solid material region, see Figure 7. For different problems, the initial value can be changed a little to get better result accordingly.

The implementation of the proposed algorithm as follows:

Step 1. Initialization of the guess topology of the structure with signed distance function in terms of the external boundary;

Step 2. Solving the equilibrium equation of the structure. FEA (Finite Element Analysis) is adopted to compute the displacement field and the adjoint displacement field through the linear elastic system;

Step 3. Computing the sensitivity of the candidate node. The value is the strain energy of a node-neighboring region α_i;

Step 4. Hole inserting, in the material region, according to the value, remove the low energy element (generally the remove rate is 2-3% of those violating the volume constraint);

Step 5. Evolving of the topology of the structure. Solve the level set equation to update the embedding function. Same as that of the Level Set Method.

Step 6. Convergence checking. If volume constraint met, then the iteration finished; or repeats Step2 - Step6 until convergence.

In Step3 and Step4, the proposed method can control the position of the inserted hole adaptively. Apart from that, the number of holes and the iteration number can be carried out individually in code implementation. For different fields of topology optimization problems, corresponding parameters should be adjusted accordingly.

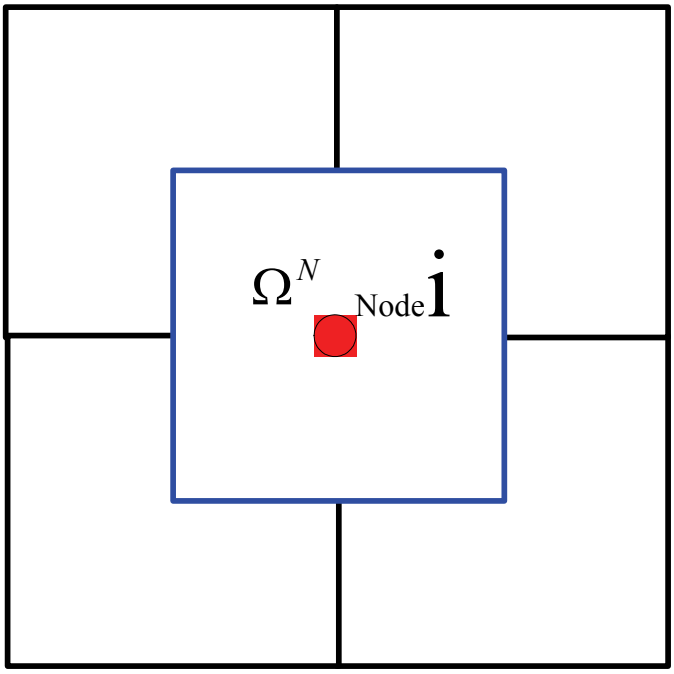


Fig. 7. Computational diagram for strain energy of node neighboring region

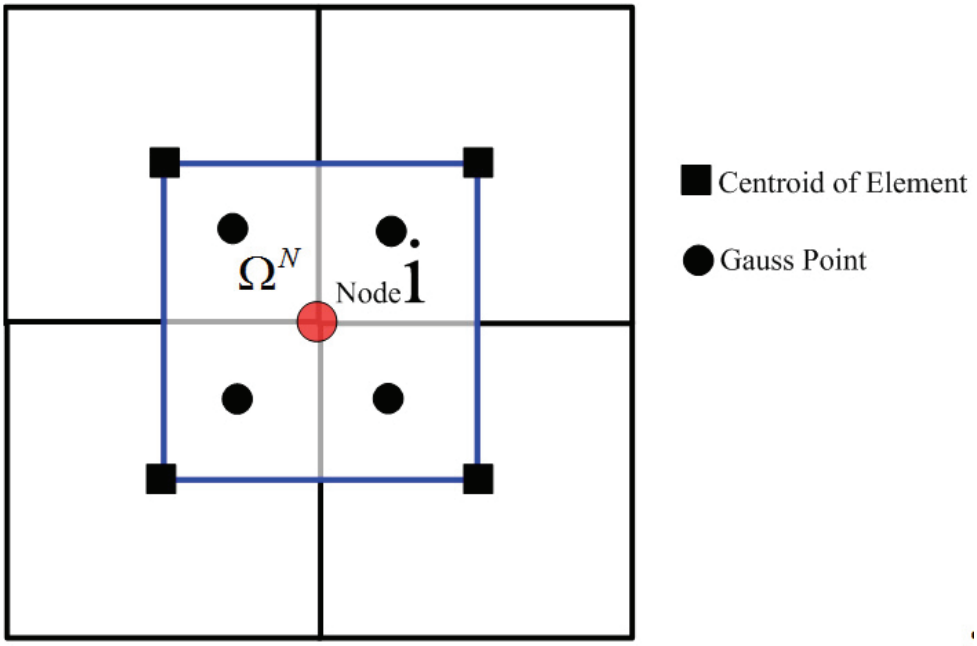


Fig. 8. Averaged strain energy of 4 neighboring gauss point

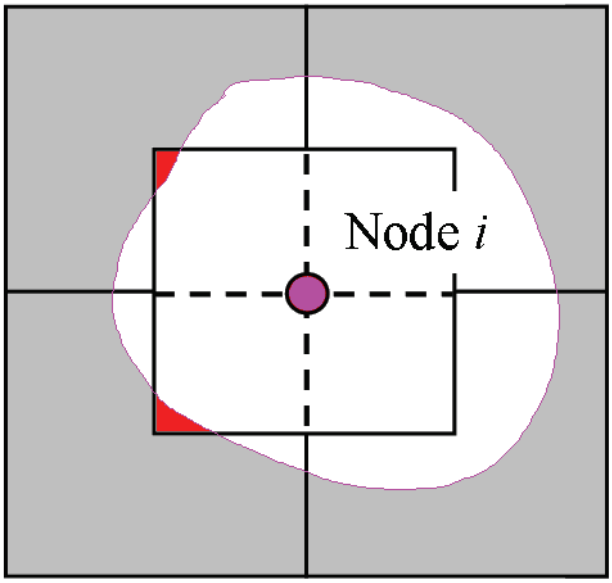


Fig. 9. Updated topology boundary after inserting a new hole during evolving optimization

3. Numerical examples

To illustrate the reliability and the validity of the nodal ESO hole-inserting LSM topology optimization method, the classical cantilever beam in Figure 1 is optimized and gets a good result. At the same time, to show the efficiency of the improved algorithm, guess topology has no hole is computed to show the characteristics.

Case 1

To solve the problem with no hole the initialization configuration has, one cannot get the optimal topology using traditional LSM algorithm easily.
According to the theory of Evolutionary Structure Optimization method, this paper gives the automated hole-inserting approach. The evolution procedure of structural topology is shown from Figure 10(a) to Figure 10(f). The topology optimization of the cantilever shows the validity of the proposed method.
Figure 11 gives the structural strain energy variation history during optimization. Figure12 shows the iteration history of material usage within the design domain during topology evolving.

Case 2

To illustrate the efficiency of the proposed algorithm, the same benchmark problem is solved. but the initialization has initialized holes and inserting holes when impossible during the optimization iteration. Figure 13(a) until Figure 13(f) gives the key intermediate topology during optimization. Optimization history shows that the iteration number decreased from 72 to 39.
Figure14 gives the structural strain energy variation history during optimization. Figure15 shows the iteration history of material usage within the design domain during topology evolving.

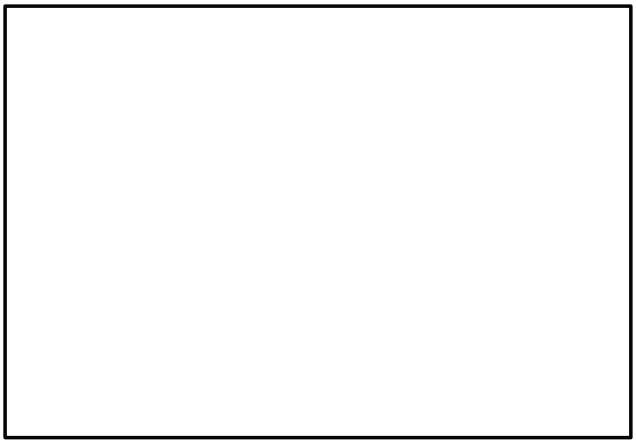


Fig. 10(a) Topology initialization without holes

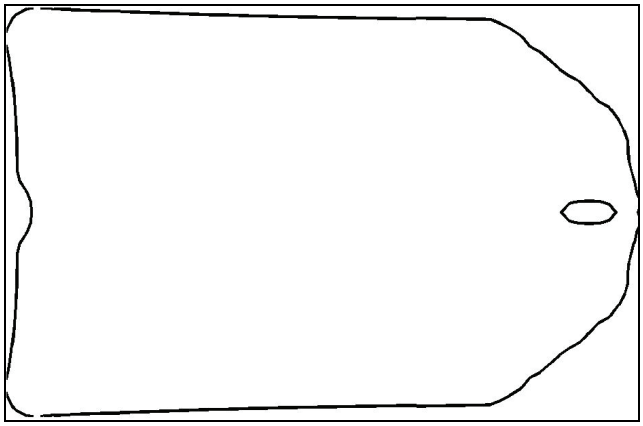


Fig. 10(b) Iteration number 14

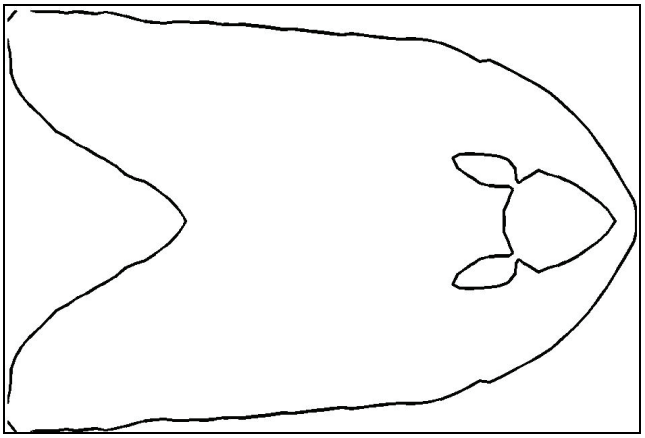


Fig. 10(c) Iteration number 30

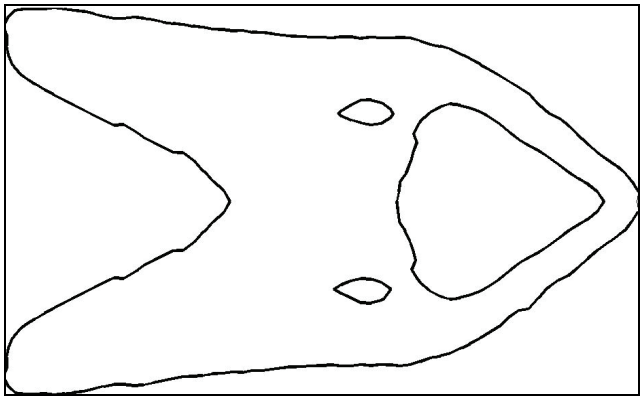


Fig. 10(d) Iteration number 46

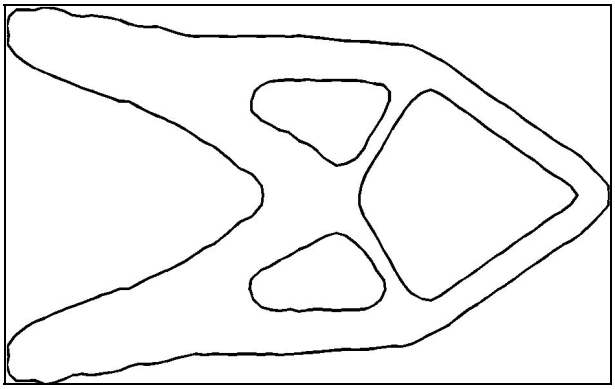


Fig. 10(e) Iteration number 61

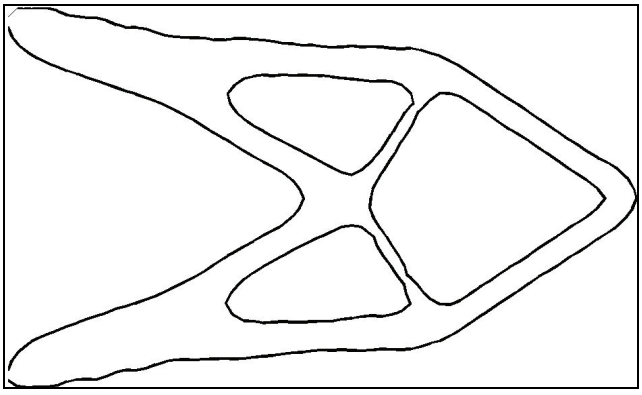


Fig. 10(f) Iteration number 72

Fig. 10. The proposed algorithm: Topology evolving process with initialization having no holes

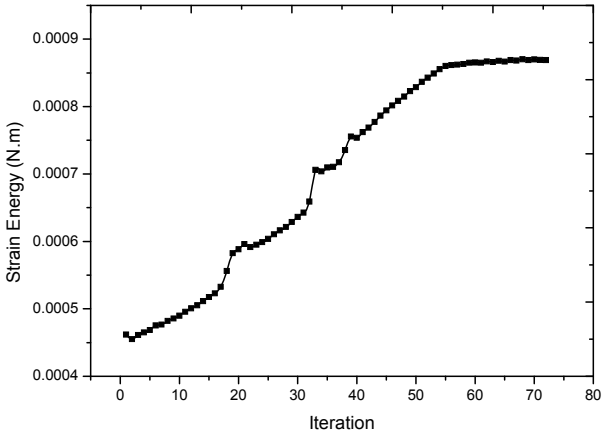


Fig. 11. Strain Energy of the structure V.S. iteration number

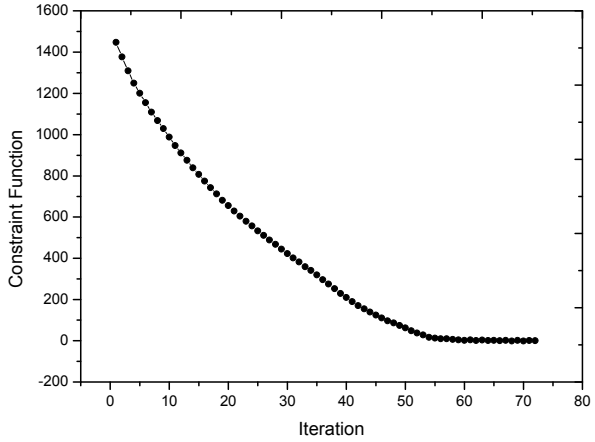


Fig. 12. Value of constraint function v.s. iteration number, the value indicted the gross material usage during the optimization

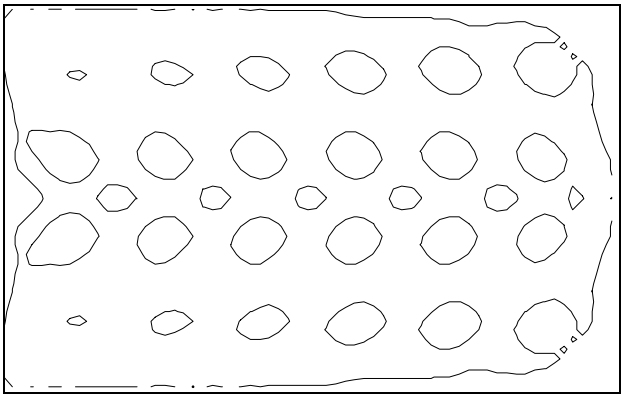


Fig. 13(a) Iteration number 6

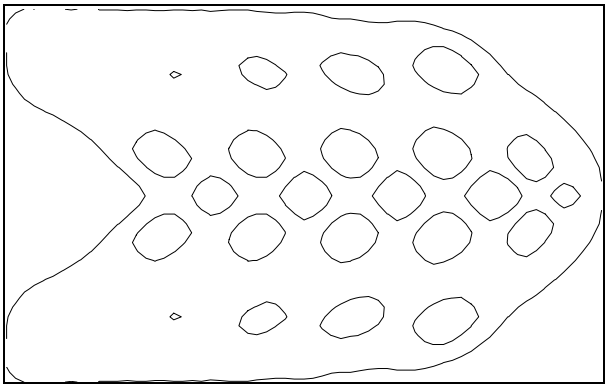


Fig. 13(b) Iteration number 13

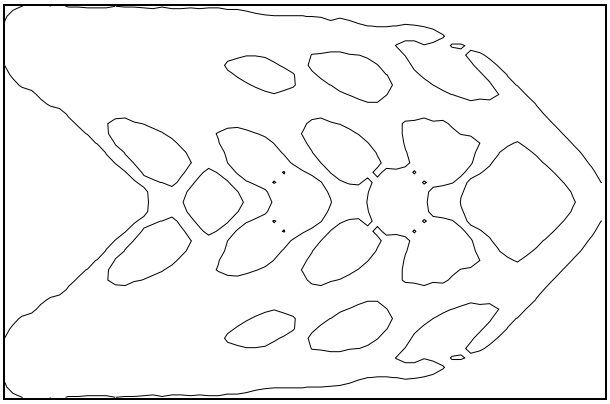


Fig. 13(c) Iteration number 20

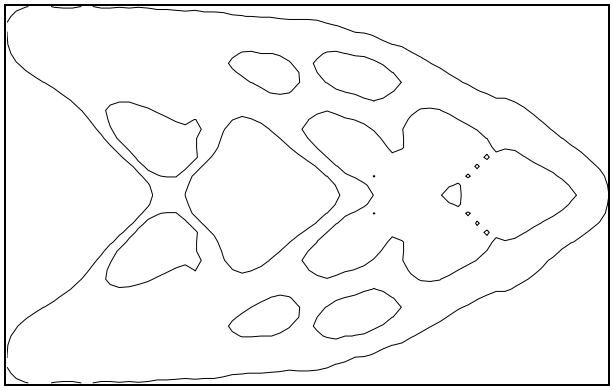


Fig. 13(d) Iteration number 27

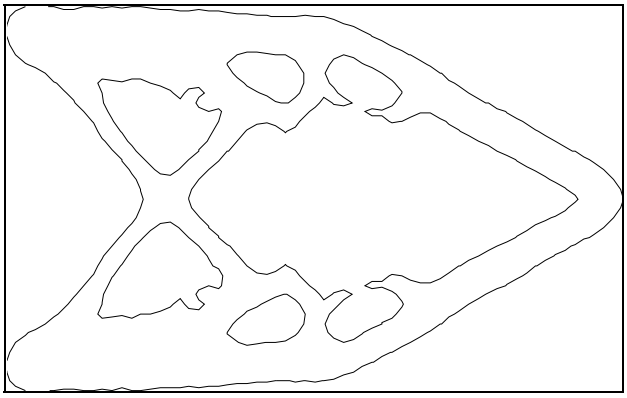


Fig. 13(e) Iteration number 33

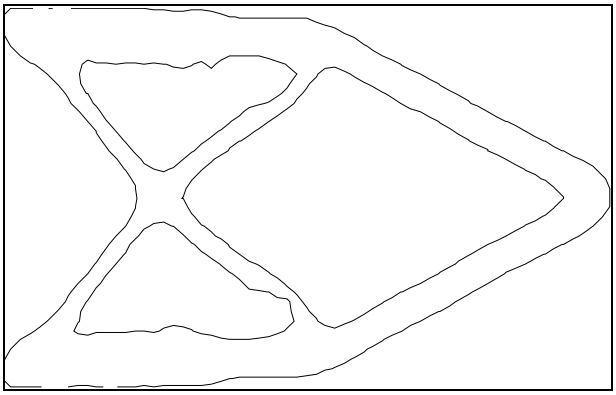


Fig. 13(f) Iteration number 39

Fig. 13. The proposed algorithm: Topology evolving process with initialization having holes

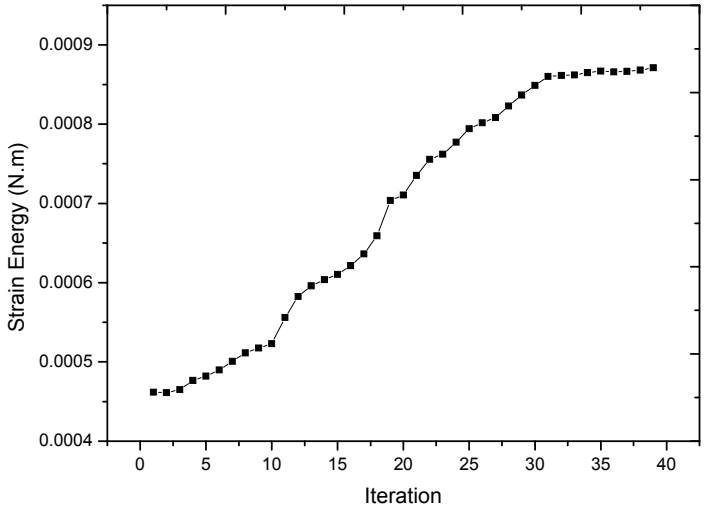


Fig. 14. Strain energy of the structure v.s. iteration number

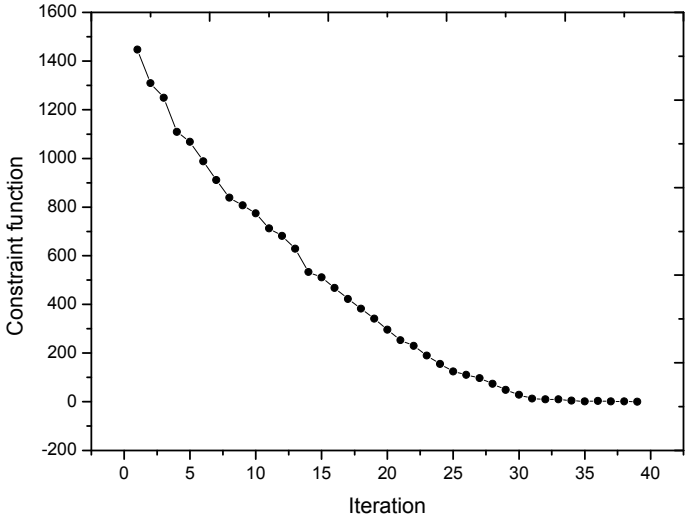


Fig. 15. Value of constraint function V.S. iteration number, the value indicted the gross material usage during the optimization

4. Conclusions

This paper proposed a LSM combined ESO hole-inserting algorithm for topology optimization. The algorithm integrated the merits of two methods and eliminated the weaknesses of conventional Level Set Method. Smooth boundary of the final topology can be gotten and need no post-processing for the manufacturability. The optimization iteration needs no explicit description of the variation of the topology, all the merits of LSM methods are kept and implemented in the new algorithm so that it makes the computation convenient and improves the efficiency accordingly. In conclusion, the nodal ESO integrated level set methods for topology optimization has the following characteristics:

1. Enlarged the optimum searching scope of the ground structure, solved the topology optimization without holes in initialization of guess configuration. which cannot get satisfied topology within proper iteration for traditional LSM method.
2. With the proposed algorithm in this paper to solve benchmark problems, the computational efficiency can be improved considerably, it can get the optimal topology in less iteration for initialization with enough initialized holes.
3. Additionally, the proposed algorithm can be used to solve other engineering problems easily if given different optimization criteria, such as local stress constraint, eigenvalue optimization and design of compliant mechanism.

Further work includes the parallelization of the genetic algorithm with the aim of reducing the iteration times as well as the extension of the proposed approach to 3D structures. Moreover, availability of the bi-direction evolutionary structure optimization algorithm the automatic mechanism of inserting and removing hole will be further implemented and integrated with the level set method to tackle more complex engineering problems.

5. Acknowledgments

This work is partially supported by National Science Foundation of Hebei (grant NO.2008000087), State Key Laboratory of Hydroscience and Engineering of Tsinghua University (grant NO.2010-TC-2) and Tsinghua University Initiative Scientific Research Program (grant NO.2009THZ07060). Their financial contributions are gratefully acknowledged.

6. References

[1] John. H. Holland. Adaptation in Nature and Artificial System: An Introductory Analysis with Applications to Biology Control and Artificial Intelligence. Cambridge, MA: MIT Press, 1975.

[2] Xie Y. M., Steven G. P. Evolutionary Structural Optimization. Springer-Verlag, Berlin, German, 1997.

[3] M. P. Bendsøe. Optimal shape design as a material distribution problem. Structural and Multidisciplinary Optimization1989; 1:193-202.

[4] Eschenauer H. A., Olhoff N. Topology optimization of continuum structures: a review. Applied Mechanics Review 2001; 54:331-390.

[5] Bendsoe M. P., Kikuchi N. Generating optimal topologies in structural design using a homogenization method. Computer Methods in Applied Mechanics and Engineering 1988; 71:197-224.

[6] Makoto Ohsaki. Simultaneous optimization of topology and geometry of a regular plane truss. Computers & Structures 1998; 66: 69-71.

[7] Chyi-Yeu Lin, Shin-Hong Lin. Artificial neural network based hole image interpretation techniques for integrated topology and shape optimization. Computer Methods in Applied Mechanics and Engineering 2005; 194: 3817-3837.

[8] Mehrdad Salami, Tim Hendtlass. A fast evaluation strategy for evolutionary algorithms. Applied Soft Computing 2003; 2(3): 156-173.

[9] Y. M. Xie, G. P. Steven. A simple evolutionary procedure for structure optimization. Computers and Structures 1993; 49: 885-896.

[10] S. Y. Wang, K. Tai, M. Y. Wang. An enhanced genetic algorithm for structural topology optimization. International Journal for Numerical Methods in Engineering 2006; 65:18-44.

[11] Soon Yu Woon, Liyong Tong, Osvaldo M. Querin and Grant P. Steven, Effective optimization of continuum topologies through a multi-GA system. Computer Methods in Applied Mechanics and Engineering 2005; 194(30-33): 3416-3437.

[12] X.Y. Yang, Y.M. Xie and G.P. Steven, Evolutionary methods for topology optimization of continuous structures with design dependent loads. Computers and Structures 2005; 83(12-13): 956-963.

[13] X.Y. Yang, Y.M. Xie, G.P. Steven and O.M. Querin. Bi-directional evolutionary method for stiffness optimization. American Institute of Aeronautics and Astronautics Journal 1999. 37(11):1483-1488.

[14] Xiaodong HUANG, Yi Min XIE and Mark Cameron BURRY. A new algorithm for bi-directional evolutionary structural optimization. JSME International Journal Series C. 2006. 49 (4): 1091-1099

[15] G. P. Steven, Q. Li, Y. M. Xie. Multi-criteria optimization that minimizes maximum stress and maximizes stiffness. Computers and Structures 2002; 80: 2433-2448.
[16] G. Marckmann P. Bettess and B. Peseux. Self designing structures: a new evolutionary rule for thickness distribution in 2D problems. Communications in Numerical Methods in Engineering 2002; 18:743-755.
[17] Mariano Victoria, Pascual Martí, Osvaldo M. Querin. Topology design of two-dimensional continuum structures using isolines. Computers and Structures 2008. 87: 101-109.
[18] K. S. Lee, Z. W. Geem. A new structural optimization method based on the harmony search. Computers & Structures 2004; 82(9-10): 781-798.
[19] H. A. Eschenauer, H. A. Kobelev. A. Schumacher. Bubble method for topology and shape optimization of structures. Structural and Multidisciplinary Optimization 1994; 8:142-151.
[20] Stanley Osher, J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithm based on hamilton-jacobi formulations. Journal of Computational Physics 1988; 79:12-49.
[21] J. A. Sethian. Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry. Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press, 1999.
[22] Stanley J. Osher, Ronald P., Fedkiw. Level Set Methods and Dynamic Implicit Surfaces. Springer-Verlag, 2000.
[23] Stanley Osher, and Ronald P. Fedkiwy. Level set methods: an overview and some recent results. Journal of Computational Physics 2001; 169: 463-502.
[24] J. A. Sethian and Andreas Wiegmann. Structural boundary design via level set and immersed interface methods. Journal of Computational Physics, 2000, 163: 489-528.
[25] Stanley J. Osher, Fadil Santosay. Level set methods for optimization problems involving geometry and constraints. Journal of Computational Physics2001; 171: 272-288.
[26] Michael Yu Wang, Xiaoming Wang, Dongming Guo. A level set method for structural topology optimization. Computer Methods in Applied Mechanics and Engineering 2003; 192: 227-246.
[27] Wang SY, Wang MY. Radial basis functions and level set method for structural topology optimization. International Journal for Numerical Methods in Engineering 2006; 65(12):2060-2090.
[28] Peng Wei, Michael Yu Wang. Piecewise constant level set method for structural topology optimization, International Journal for Numerical Methods in Engineering, 2009; 78:379-402.
[29] G. Allaire, F. Jouve. A level-set method for vibration and multiple loads structural optimization. Computer Methods in Applied Mechanics and Engineering 2005; 194: 3269-3290.
[30] A. Rietz. Sufficiency of a finite exponent in SIMP (power law) methods. Structural and Multidisciplinary Optimization 2001; 21:159-163.
[31] C. Y. Lin, L. S. Chao. Automated image interpretation for integrated topology and shape optimization. Structural and Multidisciplinary Optimization 2000; 1:125-137.
[32] Klaus-Jurgen Bathe, Finite Element Procedures. Prentice Hall, 1995.
[33] Robert D. Cook. Concepts and Applications of Finite Element Analysis. John Wiley & Sons, New York, 2001.