1 Overview

In this lecture we will introduce and examine some topics of Pseudo-randomness and we will see some applications of coding theory to them. Especially we will define l-wise independent random number generator function G and construct it. And then we will define and examine δ-almost l-wise independent G, and ϵ-biased G. And finally we will give a construction of a ϵ-biased space G using some results of coding theory.

2 Use of randomness

Usually a randomized algorithm A takes (x, y) as input where x is "real" input and y is a random string independent from x. And we hope that for some desired function $f(x)$, $Pr[A(x, y) = f(x)]$ is higher than some criteria, where probability is taken over the distribution of $y \in \{0,1\}^n$. Usually we assume that each bit of y is uniformly and independently distributed. Then how can we obtain such random string y? We may obtain y by physical sources of randomness, for example, "Zener Diode". But in many situations generating randomness by physical source may be very expensive. So computer scientists try to design algorithm that use a few random inputs and generates 'Pseudo-random' string that is pretty longer in size than its input.

3 Pseudo-randomness

Suppose that we are given a randomized algorithm A that satisfies

$$Pr_{y\in\{0,1\}^n}[A(x, y) = f(x)] \geq \frac{3}{2} \tag{1}$$

One may hope to find a $G : \{0,1\}^t \to \{0,1\}^n$ satisfying

$$Pr_{s\in\{0,1\}^t}[A(x, G(s)) = f(x)] \geq \frac{2}{3} - \epsilon. \tag{2}$$

For small ϵ. Here, We assume that $s \in \{0,1\}^t$ has uniform distribution.

- Question: For sufficiently small $\epsilon > 0$, does there exist G satisfying (2) for every A?

- The answer is No.

(Fix $G : \{0,1\}^{n-1} \to \{0,1\}^n$. Then $\exists S \in \{0,1\}^n$ such that $|S| = 2^{n-2}$ and

$$Pr_{s\in\{0,1\}^{n-1}}[G(s) \in S] \geq \frac{1}{2}. \tag{3}$$

Let $x = \emptyset$ and Let $A(x, y) = 1$ if $y \in S$, and $A(x, y) = 0$ otherwise.
Then $Pr_{y\in\{0,1\}^n}[A(y) = 0] = \frac{3}{4}$ but $Pr_{s\in\{0,1\}^t}[A(G(s)) = 0] \leq \frac{1}{2}$.)

So we may try to pick a broad class of Algorithms W and have G work for every $A \in W$. If we can do that for $W = \{$all polynomial time algorithms$\}$ or $W = \{$all polynomial sized circuits$\}$, it would be nice. But we don't know whether they have such G. For next W's it is known that they have such G's.

- $C = \{$algorithms that depend on limited independence$\}$
- $C = \{$algorithms that perform "linear tests"$\}$

In this lecture, we will deal with the first case.

4 l-wise independence

Definition 1 *We say $G : \{0,1\}^t \to \{0,1\}^n$ is l-wise independent if $\forall T \subseteq [n]$, $|T| = l$, $\forall b_1, b_2, \ldots, b_l \in \{0,1\}$,*

$$Pr_{s\in\{0,1\}^t}[G(s)|_T = (b_1, b_2, \ldots, b_l)] = 2^{-l}. \tag{4}$$

When $W = \{$algorithms that depend on less than or equal to l independence$\}$, l-wise independent G works for every $A \in W$.

To construct G that is l-wise independent, Let C be a $[n, t, ?]_2$ linear code. s.t. $C^\perp$ is a $[n, n-t, l+1]_2$ linear code.

Claim 2 *$x \mapsto C(x)$ is a l-wise independent generator.*

(For the proof of claim 2, See problem set 1, problem 4.)

Let $C^\perp$ be a BCH code with distance $(l + 1)$. Then, $C^\perp$ is a $[n, n - \lfloor \frac{l}{2} \rfloor \log n, l + 1]$ code. So C is a $[n, \lfloor \frac{l}{2} \rfloor \log n, ?]$ code. And we obtain l-wise independent G s.t.

$$G : \{0,1\}^{\lfloor \frac{1}{2} \rfloor logn} \to \{0,1\}^n \tag{5}$$

For a fixed l, $t = \lfloor \frac{1}{2} \rfloor \log n$ is polynomial over n. So it gives a polynomial sized sample space $\{0,1\}^t$ for all constant l.

5 δ-almost l-wise independence & ϵ-biased space

Sometimes l-wise independence is "stronger" than what we need. Let δ be a positive real number.

Definition 3 *$G : \{0,1\}^t \to \{0,1\}^n$ is δ-almost l-wise independent if the following holds $\forall T \subseteq [n], |T| = l$ and $\forall A : \{0,1\}^l \to \{0,1\}$,*

$$|Pr_{s\in\{0,1\}^t}[A(G(s)|_T) = 1] - Pr_{y\in\{0,1\}^l}[A(y) = 1]| \le \delta \tag{6}$$

Definition 4 *G is ϵ-biased if for every non-trivial linear function $A : \{0,1\}^n \to \{0,1\}$, if is the case that*

$$|Pr_{y\in\{0,1\}^n}[A(y) = 1] - Pr_{s\in\{0,1\}^t}[A(G(s)) = 1| \le \epsilon. \tag{7}$$

Note that for every nontrivial linear A, $Pr_{y\in\{0,1\}^n}[A(y) = 1] = \frac{1}{2}$, and there exist $T_A \subseteq [n]$ s.t. $A(y) = \bigoplus_{i\in T_A} y_i$. So, (7) becomes

$$\frac{1}{2} - \epsilon \le Pr_{s\in\{0,1\}^t}[A(G(s)) = 1] \le \frac{1}{2} + \epsilon \tag{8}$$

Proposition 5 *Every ϵ-biased generator also yields a $2^l\epsilon$-almost l-wise independent generator for all l.*

We will not prove this proposition here. Now suppose that we want a $\frac{1}{n^2}$ -almost $\log n$ -wise independent family. For $\epsilon = \frac{1}{n^3}$, if we are given ϵ-biased G, by setting l=$\log n$,G is a $\frac{1}{n^2}$-almost $\log n$-wise independent generator as we desired. So now we need to construct a $\epsilon = \frac{1}{n^3}$-biased space G.

6 construction of ϵ -biased space G

Let $N = 2^t$ and suppose that we are given $[N, n, (\frac{1}{2} - \epsilon)N]_2$ linear code C with condition that its maximum weight(number of 1's) codeword has weight at most $(\frac{1}{2} + \epsilon)N$. Suppose further that $N = \frac{n}{\epsilon^3}$. Let $n \times N$ matrix F be the generator matrix of C. Let $j : \{0,1\}^t \to [N]$ be a 1-1 correspondence. For $s \in \{0,1\}^t, 0 \le i \le n$,define

$$G(s)_i = F_{j(s),i}. \tag{9}$$

Then by the property of C, for any nonempty $T \subseteq [n]$,

$$\frac{1}{2} - \epsilon \le Pr_{s\in\{0,1\}^t}[\bigoplus_{i\in T} G(s)_i = 1] \le \frac{1}{2} + \epsilon. \tag{10}$$

So, G is an ϵ-biased space.

For $\epsilon = \frac{1}{n^3}, N = \frac{n}{\epsilon^3} = n^{10}$ So, if $t =$log $N = 10$log n then we can obtain $\frac{1}{n^2}$ -almost log n -wise independent family.

On the contrary to the Pseudo-random generator, random number extractor extracts "pure" random strings from "contaminated" random sources. Here contaminated means that it is far from uniform distribution. It takes (x, y) as input where x is contaminated random string and y is pure but short random string. Using x and y, extractor tries to get its output z near to uniform distribution. Generally z is a rather shorter string than x. In the next lecture, we will talk about random number extractor.