80240233: Complexity of Computation **Lecture 20**
ITCS, Tsinghua Univesity, Fall 2007 14 December 2007

Instructor: Andrej Bogdanov **Notes by:** Wei Yu

In Lecture 9 we saw how the Nisan-Wigderson (NW) generator can be constructed assuming assuming the existence of an "average-hard" function. In this lecture, we will show that one way functions give a Blum-Micali-Yao (BMY) type pseudorandom generator. Recall the main distinction between these two types of generators:

- NW: The random seed is very small, but the running time of the generator depends on size of distinguisher.
- BMY: The running time of the generator is independent of the size of the distinguisher, but the random seed is longer.

Definition 1 (Blum-Micali-Yao Pseudorandom Generator). *A Blum-Micali-Yao type pseudorandom generator is a collection of functions $G_n : \{0,1\}^n \to \{0,1\}^{m(n)}$ $(m(n) > n)$, such that,*

1. *On input $x \in \{0,1\}^n$, $G_n(x)$ is polynomial time computable.*
2. *For every family of polynomial size circuits C, and every polynomial $p(n)$,*
$$\left|\Pr_{x\sim\{0,1\}^n}\left[C(G_n(x)) = 1\right] - \Pr_{y\sim\{0,1\}^{m(n)}}\left[C(y) = 1\right]\right| < \frac{1}{p(n)}$$
for sufficiently large n.

Theorem 2 (Håstad, Impagliazzo, Levin, Luby). *One-way functions exist if and only if pseudorandom generators (with $m(n) = \mathrm{poly}(n)$) exist.*

In the last lecture we showed that every pseudorandom generator is a one-way function. It is more difficult to show that if one-way functions exist, then so do pseudorandom generators. In this lecture we will prove that if a special kind of one-way function called a "one-way permutation" exists, then pseudorandom generators also exist.

Definition 3 (One-way permutation). *A family of functions $f_n : \{0,1\}^n \to \{0,1\}^n$ is a* one-way permutation *if $\{f_n\}$ is a family of one-way functions and each f_n is a permutation.*

Theorem 4. *If f_n is a one-way permutation, then*
$$G_n(x, r) = (f(x), r, \langle x, r\rangle)$$
where x and r are of length n, and $\langle x, r\rangle = \sum_{i=1}^{n} x_i r_i \mod 2$ is the "inner product" of x and r modulo 2, is a pseudorandom generator from $\{0,1\}^{2n}$ to $\{0,1\}^{2n+1}$.

Since the input length will always be the same, we will write G for G_n and f for f_n.

1 Proof of Theorem 4

We prove the contrapositive. Suppose G is not a pseudorandom generator. Then there exists a family of circuits C of size $s(n) = \text{poly}(n)$ and a function $\epsilon(n) = 1/\text{poly}(n)$ such that

$$\left|\Pr_{x,r\sim\{0,1\}^n}\left[C\left(f(x), r, \langle x, r\rangle\right) = 1\right] - \Pr_{y\sim\{0,1\}^{2n+1}}\left[C(y) = 1\right]\right| > \epsilon(n)$$

We will construct a new family of polynomial-size circuits C' such that $\Pr_{x\sim\{0,1\}^n}[C'(f(x)) = x] \geq \epsilon(n)/2$, and conclude that f is not one-way.

Let's view C as a distinguisher between the distributions $(f(x), r, \langle x, r\rangle)$ and the uniform distribution U_{2n+1} on $2n+1$ bits. If we write $z = f(x)$, then C distinguishes between the distributions

$$\left(z, r, \langle f^{-1}(z), r\rangle\right) : z, r \sim U_n \quad \text{and} \quad U_{2n+1}.$$

We can now turn the distinguisher C into a predictor P (see Lemma 5 in lecture 9) of size $s(n)+O(1)$ which satisfies

$$\Pr_{x,r}\left[P(z, r) = \langle f^{-1}(z), r\rangle\right] > \frac{1}{2} + \epsilon$$

which is the same as

$$\Pr_{x,r}\left[P(f(x), r) = \langle x, r\rangle\right] > \frac{1}{2} + \epsilon$$

By Markov's inequality, it follows that

$$\Pr_x\left[\Pr_r\left[P(f(x), r) = \langle x, r\rangle\right] > \frac{1}{2} + \frac{\epsilon}{2}\right] > \frac{\epsilon}{2}$$

Let S be the set of all x such that

$$\Pr_r\left[P(f(x), r) = \langle x, r\rangle\right] > \frac{1}{2} + \frac{\epsilon}{2} \tag{1}$$

This suggests the following algorithm for inverting $f(x)$ when $x \in S$: On input $z = f(x)$, try to find all x' such that $\Pr_r\left[P(z, r) = \langle x', r\rangle\right] > 1/2 + \epsilon/2$. Since x satisfies (1), one of these x' must equal x. To find out which one, apply $f(x')$ to all of them and see which one maps to z. Since f is a permutation, if $f(x') = z$ it must be that $x' = x$.

Can we carry out this computation by a polynomial-size circuit? At first, the idea seems unreasonable: It looks like there might be exponentially many x' such that $\Pr_r\left[P(z, r) = \langle x', r\rangle\right] > 1/2+\epsilon/2$, so even listing all of them, much less computing them, may take too much time. However, this is not the case, and the search of x' can be carried out in polynomial-time, thanks to the following theorem:

Theorem 5 (Goldriech-Levin)**.** *There is a randomized algorithm A which on input ϵ and given oracle access to $g : \{0,1\}^n \to \{0,1\}$ runs in time polynomial in n and $1/\epsilon$ and with probability $2/3$ outputs a list that contains all x such that*

$$\Pr_r\left[g(r) = \langle x, r\rangle\right] \geq \frac{1}{2} + \epsilon.$$

2 Proof of the Goldreich-Levin theorem

We will start by proving a much weaker statement than what is required, and strenghten it in stages to derive the proof of the theorem. Our goal is to design an algorithm A that outputs all x such that

$$\Pr_r\left[g(r) = \langle x, r\rangle\right] \geq p.$$

where $p = 1/2 + \epsilon$. Let us however start with the case $p = 1$.

Case $p = 1$. In this case, A can evaluate $g(r) = \langle x, r\rangle$ for every r and wants to "recover" x. It is not hard to see that g uniquely determines x, and the ith bit of x is given by $x_i = \langle x, e_i\rangle = g(e_i)$, where e_i is the string that has 1 in the ith coordinate and 0 everywhere else. So in this way we can recover x bit by bit.

Case $p = 1 - \frac{1}{6n}$. Now we need to work a bit harder, since it might be the case that $g(e_i) \neq \langle x, e_i\rangle$, so querying g at e_i might be misleading. But we can deduce the value $\langle x, e_i\rangle$ by querying g at two random points, using a similar trick as when we did worst-case to average-case reductions for the permanent in Lecture 16. We know that for every r, $x_i = \langle e_i, x\rangle = \langle x, r\rangle + \langle x, r + e_i\rangle$. Moreover, for a random r, the strings r and $r + e_i$ are both uniformly random in $\{0, 1\}^n$. So, if we choose a random r and compute $g(r) + g(e_i + r)$, we have that

$$\Pr_r[g(r) + g(e_i + r) \neq x_i] \leq \Pr_r[g(r) \neq \langle x, r\rangle] + \Pr_r[g(e_i + r) \neq \langle x, e_i + r\rangle] < \frac{1}{6n} + \frac{1}{6n} < \frac{1}{3n}$$

By taking a union bound, we have that $\Pr_r[\exists i : g(e_i + r) + g(r) \neq x_i] < \frac{1}{3}$. So with probability 2/3 this randomized algorithm recovers all the bits of x.

Case $p = \frac{3}{4} + \epsilon$. In the above algorithm, we now have that

$$\Pr_r[g(r) + g(e_i + r) \neq x_i] \leq \Pr_r[g(r) \neq \langle x, r\rangle] + \Pr_r[g(e_i + r) \neq \langle x, e_i + r\rangle] < 2(1/4 - \epsilon) < 1/2 - 2\epsilon.$$

We cannot take a union bound of n such events anymore. However, by repeating this procedure several times, we can increase its success probability from $1/2 - 2\epsilon$ to $1/3n$, and then take the union bound.

In particular, consider the following algorithm: For each $1 \leq i \leq n$, compute the value $g(r) + g(e_i + r)$ for $t = O(\log n/\epsilon^2)$ independent values of r and let x_i equal the majority of the answers. Each trial gives the correct value for x_i with probability $1/2 - 2\epsilon$ and the trials are independent, so by the Chernoff bound the probability that a majority of the trials fails is at most $\exp((2\epsilon)^2 t/2) < 1/3n$.

Case $p = 1/2 + \epsilon$. We now give the proof of the theorem. It turns out that an interesting phenomenon happens when we try to take $p \leq 3/4$. By the analysis of the previous case, it follows that when $p > 3/4$, there is a unique x such that $\Pr[g(r) = \langle x, r\rangle] \geq p$. However, when $p \leq 3/4$, there may be two or more such xs. So we must introduce a way into the algorithm to disambiguate between the different solutions.

There is also an evident (and related) problem with the above analysis: If we attempt to use the same algorithm, we would get that $\Pr[g(r) + g(e_i + r) = x_i] < 1 - 2\epsilon$, so it appears that we do not obtain any information about the value x_i.

However, suppose that someone could tell us the values $h_r = \langle x, r\rangle$ needed by the algorithm, so we wouldn't have to query g to get them and make potential mistakes. Then we would have

$$\Pr_r[h_r + g(e_i + r) \neq x_i] \leq \Pr_r[g(e_i + r) \neq \langle x, e_i + r\rangle] < 1/2 - \epsilon. \tag{2}$$

so the previous algorithm would work – provided that we knew the values $h_r = \langle x, r\rangle$.

How can we get hold of the values h_r? One possibility is to simply guess them, and think of each possible guess as giving a candidate value for x. So to obtain a list of all x, one can simply go through all the choices for h_r. How many such choices are there? For each i, the algorithm uses $O(\log n/\epsilon^2)$ choices of r, and there are n possible values of i, so we need to guess $O(n \log n/\epsilon^2)$ different values h_r. It looks like going through all the choices would take time exponential in n!

One place in the algorithm where we can save immediately is this: Instead of using independent choices of r for the different coordinates i, we can in fact make the same choices. In the end, our analysis works by taking a union bound over i, so it does not matter if the randomness used for different coordinates is the same. This will reduce the number of random strings r needed by the algorithm to $O(\log n/\epsilon^2)$, so the number of possible choices for h_r becomes $2^{O(\log n/\epsilon^2)} = n^{O(1/\epsilon^2)}$. Recall that in our setting, $\epsilon = 1/\text{poly}(n)$, so this is still too large.

To further improve the algorithm, we introduce additional correlations among the rs. To amplify the success probability of (2) from $1/2 + \epsilon$ to $1 - 1/3n$, it is not really necessary that the rs are independent. It turns out that we can choose them in a dependent way so that we only need to guess the value h_r for a very small number of r, and this will automatically yield guesses for the other rs.

We choose the rs from the following distribution: First, choose a "basis" $r_1, \dots, r_m \sim \{0,1\}^n$ independently at random, where $m = O(\log(n/\epsilon^2))$. Then, for every subset $S \subseteq \{1, \dots, m\}$, set $r_S = \sum_{j \in S} r_j$. Notice that guesses h_j for the values $\langle x, r_j\rangle$ automatically yield guesses h_S for the values $\langle x, r_S\rangle$ via the formula $\langle x, r_S\rangle = \sum_{j\in S}\langle x, r_j\rangle$.

We can now give the algorithm A from the theorem:

```
$A^g$: Choose $r_1, \dots, r_m$ independently at random from $\{0,1\}^n$.
    For every choice of values $h_1, \dots, h_m \in \{0,1\}$:
        For every $1 \leq i \leq n$:
            For every $S \subseteq \{0,1\}^m$:
                Set $r_S = \sum_{j\in S} r_i$ and $h_S = \sum_{j \in S} h_j$.
                Compute $a_{i,S} = h_S + g(e_i + r_S)$.
            Set $x_i = \text{majority}_S(a_{i,S})$.
        Output $x = x_1 \dots x_n$.
```

We choose $m = \log(6n/\epsilon^2)$, so the number of possible choices for $h_1, \dots, h_m$ is $6n/\epsilon^2$ and the running time of the algorithm is polynomial in n and ϵ.

Claim 6. *For every x such that $\Pr_r[g(r) = \langle x, r\rangle] \geq 1/2 + \epsilon$, with probability $2/3$ over the choice of $r_1, \dots, r_m$, A^g outputs x.*

This claim almost proves the Goldreich-Levin theorem. The only difference is that it only guarantees each x satisfying the condition will appear in the list with probability $2/3$, while the theorem says that the list contains all such x with probability $2/3$. To take care of this, we run the algorithm 2^n times and take the union of all the lists output by it. Then each such x will appear in the list with probability $1 - 3^{-n}$, so by a union bound the list will contain all such x with probability $2/3$.

Proof. We will show that A^g outputs x when $h_i = \langle x, r_j\rangle$ for all $1 \leq j \leq m$, which also implies $h_S = \langle x, r_S\rangle$ for every $S \subseteq \{1, \dots, m\}$. Let us fix this choice for h_i. As before, is enough to show that for all i,

$$\Pr_r\bigl[\text{majority}_S(h_S + g(e_i + r_S)) = x_i\bigr] > 1 - \frac{1}{3n}.$$

Let

$$Y_S = \begin{cases} 1, & \text{if } h_S + g(e_i + r_S) = x_i, \\ 0, & \text{otherwise.} \end{cases}$$

Then for every S,

$$\Pr[Y_S = 1] = \Pr[h_S + g(e_i + r_S) = x_i] = \Pr[g(e_i + r_S) = \langle x, e_i + r_S\rangle] \geq \frac{1}{2} + \epsilon$$

The main observation here is that the random variables Y_S are pairwise independent, since the variables r_S are pairwise independent and Y_S is determined by r_S. We can therefore use Chebyshev's inequality to obtain a deviation bound on $Y = \sum_{S\subseteq\{0,1\}^m} Y_S$. Let us assume for simplicity that $\Pr[Y_S = 1] = 1/2 + \epsilon$. Then

$$\mathrm{E}[Y] = \sum_{S\subseteq\{0,1\}^m} \mathrm{E}[Y_S] = (1/2 + \epsilon)\cdot 2^m \qquad \text{and} \qquad \mathrm{Var}[Y] = \sum_{S\subseteq\{0,1\}^m} \mathrm{Var}[Y_S] \leq 2^m.$$

By Chebyshev's inequality, we have

$$\begin{aligned} \Pr[x_i \neq \text{majority}_S(a_{i,S})] &\leq \Pr\bigl[Y < \mathrm{E}[Y] - \epsilon\cdot 2^m\bigr] \\ &\leq \Pr\Bigl[Y < \mathrm{E}[Y] - \epsilon\cdot 2^{m/2}\cdot\sqrt{\mathrm{Var}[Y]}\Bigr] \\ &\leq \frac{1}{(\epsilon\cdot 2^{m/2})^2} < \frac{1}{3n}. \end{aligned}$$

□