80240233: Complexity of Computation **Lecture 9**
ITCS, Tsinghua Univesity, Fall 2007 6 November 2007

Instructor: Andrej Bogdanov **Notes by:** Andrej Bogdanov

Pseudorandomness is the study of how randomized algorithms can be simulated deterministically. Every randomized algorithm admits a natural deterministic simulation which can be obtained by running the algorithm over all possible settings of the randomness and looking at the distribution of outputs. However such simulations are very inefficient; can we do anything better?

A priori, it might seem that the answer should be no. Just as we believe that the best deterministic simulation of NP should require one to look at all possible witnesses, it might seem that simulating a BPP algorithm should require one to look at all possible settings of the randomness. Somewhat surprisingly, we show that this is *not* the case — if we assume a very plausible conjecture about the existence of certain hard functions:

Theorem 1. *Suppose there is a decision problem g that can be decided in time $2^{O(n)}$, but cannot be decided by any circuit family of size $2^{o(n)}$ (on almost all input lenghts). Then* $\mathrm{P} = \mathrm{BPP}$.

We think of g as a "hard" function, like the parity function is hard for AC^0 circuits. The theorem says that if a certain type of hard problem exists, then every BPP computation can be performed deterministically. In fact, as we will see the proof of Theorem 1 gives something even stronger. It shows that there is a universal efficient deterministic simulation for all efficient randomized algorithms of the BPP type.

The proof of Theorem 1 splits in two parts, which we state as separate theorems below.

Theorem 2. *Suppose there is a decision problem g that can be decided in time $2^{O(n)}$ but cannot be decided by any circuit family of size $2^{o(n)}$. Then there is a decision problem f that can be decided in time $2^{O(n)}$ and such that for some $\delta > 0$ and every circuit family C of size $2^{\delta n}$ and sufficiently large n*

$$\mathrm{Pr}_{x\sim\{0,1\}^n}[C(x) = f(x)] < 1/2 + 2^{-\delta n}.$$

This theorem gives a way to turn a hard problem g into a very hard problem f. We assume that g cannot be computed by sufficiently small circuits, meaning that on every input length there is *some* input on which g is not computed correctly. Given such an g, we obtain a f such that no circuit can determine f on much more than half the inputs. In other words, no small circuit can do much better at predicting the value $f(x)$ than guessing at random.

Theorem 3. *Suppose there is a decision problem f that can be decided in time $2^{O(n)}$ and such that for every $\delta > 0$ and every circuit family C of size $2^{\delta n}$ and sufficiently large n*

$$\mathrm{Pr}_{x\sim\{0,1\}^n}[C(x) = f(x)] < 1/2 + 2^{-\delta n}.$$

Then $\mathrm{P} = \mathrm{BPP}$.

In this lecture we prove Theorem 3. So let's assume the existence of the function f with the desired properties.

1 Pseudorandom generators

The basic strategy of the proof is as follows. To prove that BPP = P, we show a way to simulate every BPP algorithm A deterministically in an efficient way. In this setting, it will be helpful of A not as a function of its actual input x, but as a function of its randomness r. The condition that A is a BPP algorithm for decision problem L tells us that

$$\begin{aligned} x \in L &\implies \Pr_{r\sim\{0,1\}^m}[A(x,r)=1] \geq 2/3 \\ x \notin L &\implies \Pr_{r\sim\{0,1\}^m}[A(x,r)=1] \leq 1/3. \end{aligned}$$

When the input is of length n, we assume that the algorithm uses $m = m(n)$ bits of randomness. Since A runs in polynomial time, m grows polynomially in n.

The idea is to replace the random string r used by A by a pseudorandom string $G(s)$ which uses much less randomness to generate, yet doesn't affect by much the success probability of A. If $G(s)$ can be generated efficiently using $k = O(\log m)$ bits of randomness, then we could simulate A on input x deterministically by enumerating all possible outputs of $G(s)$ and observing what fraction of the time $A(x, G(s))$ accepts. If A were to accept, a majority of the outputs $G(s)$ should yield accepting computations; if A were to reject, the majority of them should yield rejecting computations. That is, we want

$$\begin{aligned} x \in L &\implies \Pr_{s\sim\{0,1\}^k}[A(x,G(s))=1] > 1/2 \\ x \notin L &\implies \Pr_{s\sim\{0,1\}^k}[A(x,G(s))=1] < 1/2. \end{aligned}$$

To achieve this, it is sufficient that for all x,

$$\left|\Pr_{r\sim\{0,1\}^m} A(x,r)=1] - \Pr_{s\sim\{0,1\}^k}[A(x,G(s))=1]\right| < 1/6.$$

The next step is a bit unusual, but crucial: Instead of looking to construct a specific G that works for our algorithm A, we will give a generic G that satisfies the above condition with respect to all algorithms that run in fixed polynomial time $t(n)$. In fact, it will be even more convenient to think of $A(x,r)$ as a collection of circuits $C_x(r)$ – where the input x of A is hardwired into the circuit C_x – of size $S(n)$, where $S(n)$ grows polynomially in n. Then the above condition can be written as

$$\left|\Pr_{r\sim\{0,1\}^m} C_x(r)=1] - \Pr_{s\sim\{0,1\}^k}[C_x(G(s))=1]\right| < 1/6.$$

This motivates the following definition.

Definition 4. $G : \{0,1\}^k \to \{0,1\}^m$ *is an* ϵ-pseudorandom generator *(in short,* ϵ-pseudorandom*) against size* S *circuits if for every circuit* C *of size* S*,*

$$\left|\Pr_{r\sim\{0,1\}^m} C(r)=1] - \Pr_{s\sim\{0,1\}^k}[C(G(s))=1]\right| < 1/6.$$

So to simulate every BPP algorithm deterministically in an efficient way, it is sufficient to give an "efficient" construction of a family of pseudorandom generators G against circuit families of every polynomial size $S(n)$ with "short" seed length k. What does it mean for k to be short and G to be efficient? If we want to enumerate all 2^k outputs of G in time polynomial in m, we must have

$k = \Theta(\log m)$, and G must run in time polynomial in m. Since $m = 2^{O(k)}$, this means that G runs in time $2^{O(k)}$ – exponential in the length of its input s.

To summarize, for every polynomial growing size bound $S(n)$ we want a family of 1/6-pseudorandom generators $G : \{0,1\}^k \to \{0,1\}^m$ against size $S(n)$, where $k = O(\log m)$ and G is computable in time $2^{O(k)}$.

2 Turning hardness into pseudorandomness

We want to construct G just based on the assumption that there exists a certain "hard" function f. How do we turn a hard function into a pseudorandom generator? The key idea here is that since the function f is hard to compute – with very small advantage over random – for circuits of a given size, the output of f must "look random" to such circuits. Another way of saying this is that conditioned on s, $f(s)$ is a random-looking string for such circuits, or equivalently, the string $(s, f(s))$ behaves much like a random string of length $k+1$ for such circuits. So it seems that the function $G(s) = (s, f(s))$ should be a good pseudorandom generator. This is indeed the case.

Lemma 5. *Suppose that for every circuit C of size S,*

$$\Pr_{s\sim\{0,1\}^k}[C(s) = f(s)] < 1/2 + \epsilon.$$

Then the function $G : \{0,1\}^k \to \{0,1\}^{k+1}$ given by $G(s) = (s, f(s))$ is ϵ-pseudorandom against size $S - O(1)$.

The proof of this lemma is not technically difficult, but it contains interesting conceptual ideas.

Proof. We argue by contradiction – we assume that G is not ϵ-pseudorandom against size S'; that is, there is a circuit C of size S' such that

$$\big|\Pr[C(s, f(s)) = 1] - \Pr[C(s, b) = 1]\big| \geq \epsilon.$$

Here, s is chosen randomly from $\{0,1\}^k$ and b is a random bit independent of s. We want to use C to construct another circuit C'' of size $S' + O(1)$ that computes f with advantage ϵ over random.

The first step is to get rid of the absolute value. By taking either $C'(s) = C(s)$ or $C'(s) = \overline{C(s)}$, we have that

$$\Pr[C'(s, f(s)) = 1] - \Pr[C'(s, b) = 1] \geq \epsilon.$$

where C' has the same size as C (recall that NOT gates don't count towards circuit size).

This equation can be interpreted in the following way: If we think of $C'(s, b)$ in terms of the input b, then C' is more likely to accept when $b = f(s)$ than when b is completely random. So it seems that the output of $C'(s, b)$ when b is random should make a good predictor for the value $f(s)$: If $C'(s, b) = 1$, then b is more likely to be $f(s)$, and if $C'(s, b) = 0$, then b is more likely to be $\overline{f(s)}$. This suggests the following randomized procedure C'' for predicting f:

```
$C''$ : On input $s$,
    Choose $b \sim \{0,1\}$ at random
        If $C'(s,b) = 1$, output $b$
        If $C'(s,b) = 0$, output $\overline{b}$.
```

We will show that

$$\Pr_{s,b}[C''(s,b) = f(s)] \geq 1/2 + \epsilon \tag{1}$$

so in particular there exists a choice of b for which

$$\Pr_s[C''(s,b) = f(s)] \geq 1/2 + \epsilon.$$

If we fix this choice of b into the circuit C'', we obtain a deterministic circuit C'' of size $S' + O(1)$ that contradicts the assumption of the lemma.

It remains to prove (1). For this, it is sufficient to show that for every fixed $s \in \{0,1\}^k$:

$$\Pr_b[C''(s,b) = f(s)] \geq 1/2 + \Pr[C'(s,f(s)) = 1] - \Pr_b[C'(s,b) = 1]. \tag{2}$$

This can be checked by a somewhat tedious case analysis. We fix s and look $C'(s,b)$ as a function of b. There are four possibilities for what this function can be: 0, 1, b or $\overline{b}$.

$C'(s,b)$	$\Pr[C'(s,f(s)) = 1]$	$\Pr_b[C'(s,b) = 1]$	$\Pr_b[C''(s) = f(s)]$
0	0	0	1/2
1	1	1	1/2
b	$f(s)$	1/2	$f(s)$
$\overline{b}$	$\overline{f(s)}$	1/2	$\overline{f(s)}$

We can check that the condition (2) holds in all four cases. □

The pseudorandom generator of Lemma 5 falls short on one crucial requirement: It saves only one bit of randomness, namely $k = m - 1$. To achieve $k = O(\log m)$, we need more work.

3 The Nisan-Wigderson generator

One way to improve the construction of Lemma 5 is to use several copies of the generator: Partition the input s into several disjoint substrings and run different copies of the generator on each one of these strings. This idea can save us more bits of randomness, but still falls very short of what we aim to achieve. However, if we allow the substrings of s to have some intersection instead of making them disjoint, then the output of the resulting generator could be potentially much longer than its input.

The problem is that as the substrings of s begin to intersect, the outputs corresponding to them are no longer independent. However if we keep the intersection sizes small, we could hope that there is enough independence left among the outputs so that it remains pseudorandom. This is the idea behind the Nisan-Wigderson generator, which we describe next. First we need to quantify what we mean by keeping the intersection sizes small.

Definition 6. *A collection of sets $T_1, \dots, T_m \subseteq \{1, \dots, k\}$ is a* combinatorial design *with set size t and intersection size $t_\cap$ if $|T_i| = t$ for every i and $|T_i \cap T_j| \le t_\cap$ for every $i \ne j$.*

Given a combinatorial design, we define a pseudorandom generator $G : \{0,1\}^k \to \{0,1\}^m$ by

$$G(s) = (f(s|_{T_1}), \dots, f(s|_{T_m}))$$

where $f : \{0,1\}^t \to \{0,1\}$ is the "hard" function and s_T is the substring of s indexed by the elements of the set T – for instance if $s = s_1 s_2 s_3 s_4$, then $s|_{2,4} = s_2 s_4$. It turns out that combinatorial designs with good parameters can be computed efficiently.

Claim 7. *For every constant $\gamma > 0$ there is a family of combinatorial designs with*

$$k = O(\log m/\gamma) \qquad t = \log m/\gamma \qquad t_\cap = \log m.$$

Moreover, this family can be constructed deterministically in time $m^{O(1/\gamma)}$.

In particular, for every fixed γ we have $k = O(\log m)$, so the seed size is exactly what we were aiming for. Let's now check that G can be computed efficiently (in time poly(m)). To compute G, we first construct the design in time $m^{O(1/\gamma)}$. We then need to evaluate m copies of f, each on an input of size $t = \log m/\gamma$. Since we assumed that f is computable in time $2^{O(t)} = m^{O(1/\gamma)}$, the whole computation can be done in time polynomial in m.

It remains to show that G is 1/6-pseudorandom against circuits of every polynomial size $S(n)$. (Looking ahead, the choice of γ will depend on the polynomial bound $S(n)$.) We argue by contradiction: If G is not 1/6-pseudorandom, then for some circuit C of size $S(n)$ we have

$$\Pr_{s\sim\{0,1\}^k}[C(G(s)) = 1] - \Pr_{r\sim\{0,1\}^m}[C(r) = 1] > 1/6.$$

(As before, we can remove the absolute value in the definition of pseudorandom without loss of generality.) Let's expand this definition:

$$\Pr_{s\sim\{0,1\}^k}[C(f(s|_{T_1}), \dots, f(s|_{T_m})) = 1] - \Pr_{r_1,\dots,r_m\sim\{0,1\}}[C(r_1, \dots, r_m) = 1] > 1/6. \tag{3}$$

This formula does not appear all that useful. To see what is happening, we introduce of the following way of "slowly" going from the pseudorandom distribution $G(s)$ to the random distribution r: At each step, we change one input of C from pseudorandom to random. If C can distinguish $G(s)$ from r, then at some step there must be a noticeable change in the behavior of C.

More formally, we consider the following sequence of *hybrid distributions* on inputs of C:

$$\begin{array}{rcccc}
D_m : & f(s|_{T_1}), & \dots, & f(s|_{T_{m-1}}), & f(s|_{T_m}) \\
D_{m-1} : & f(s|_{T_1}), & \dots, & f(s|_{T_{m-1}}), & r_m \\
 & \vdots & & \vdots & \vdots \\
D_0 : & r_1, & \dots, & r_{m-1}, & r_m.
\end{array}$$

These distributions are not "real"; we merely use them to help us in the analysis. Condition (3) tells us that $\Pr_{r\sim D_m}[C(r)] - \Pr_{r\sim D_0}[C(r)] > 1/6$. Then there must be some j between 1 and m for which $\Pr_{r\sim D_j}[C(r)] - \Pr_{r\sim D_{j-1}}[C(r)] > 1/6m$, that is

$$\begin{aligned}\Pr_{s\sim\{0,1\}^k, r_{j+1},\ldots,r_m\sim\{0,1\}}&[C(f(s|_{T_1}),\ldots,f(s|_{T_j}),\ldots,r_m)=1]\\ &- \Pr_{s\sim\{0,1\}^k, r_j,\ldots,r_m\sim\{0,1\}}[C(f(s|_{T_1}),\ldots,r_j,\ldots,r_m)=1] > 1/6m.\end{aligned}$$

There must then exist a fixing of the values $r_{j+1},\ldots,r_m$ that maximizes the above difference in probabilities. If we hardwire this fixing into the circuit C, we obtain a circuit C_1 of the same size such that

$$\Pr_s[C_1(f(s|_{T_1}),\ldots,f(s|_{T_j}))=1] - \Pr_{s,r_j}[C_1(f(s|_{T_1}),\ldots,r_j)=1] > 1/6m.$$

Let $s' = s|_{T_j}$ (this is a string of length t). There is now a fixing of all the bits of s outside s' that maximizes the above difference in probabilities. Let's hardwire these bits into C_1 and call the resulting circuit C_2. With respect to this fixing, for every $i < j$, $f(s|_{T_i})$ becomes a function of at most $\log m$ bits in s' (because s' intersects $s|_{T_i}$ in at most $t_\cap = \log m$ positions). Let's call this function $g_i(s')$. We then have

$$\Pr_{s'}[C_2(g_1(s'),\ldots,g_{j-1}(s'),f(s'))=1] - \Pr_{s',r_j}[C_2(g_1(s'),\ldots,g_{j-1}(s'),r_j)=1] > 1/6m.$$

Since each g_i is a function of at most $\log m$ bits, it can be computed by a circuit of size $O(2^{\log m}) = O(m)$. If we compose the circuit C' with the circuits for $g_1,\ldots,g_{j-1}$, we obtain a single circuit C_3 of size $S(n)+O(jm) = S(n)+O(m^2)$ such that

$$\Pr_{s'}[C_3(s',f(s'))=1] - \Pr_{s',r_j}[C_3(s',r_j)=1] > 1/6m.$$

In words, $(s', f(s'))$ is not $1/6m$-pseudorandom for size $S(n)+O(m^2)$; by Lemma 5 it follows that there is a circuit C_4 of size $S(n)+O(m^2)$ such that

$$\Pr_{s'}[C_4(s') = f(s')] > 1/2 + 1/6m.$$

Recall that f is a function on $t = \log m/\gamma$ bits, so we have that

$$\Pr_{s'}[C_4(s') = f(s')] > 1/2 + 1/6 \cdot 2^{-\gamma t}$$

where C_4 is a circuit of size $S(n)+O(m^2) \le 2^{c\gamma t}$ for some constant $c > 0$. If we choose $\delta = c\cdot\gamma$ we have that C_4 is a circuit of size $2^{\delta t}$ that predicts f with advantage $2^{-\delta t}$, contradicting the assumed hardness of f.