Notes on Complexity Theory Last updated: November, 2011

Lecture 25

Jonathan Katz

1 Time-Bounded Derandomization

Randomization provides unconditional benefits in many settings; examples include cryptography (where random keys are used to provide protection against an adversary) and distributed computing (where randomness can be used as a means to break symmetry between parties). Randomness also appears to help in algorithm design. But is it possible that, from a complexity-theoretic perspective, randomness does *not* help? E.g., might it be the case that every problem that can be solved in randomized polynomial time can also be solved in deterministic polynomial time? (That is, is $\mathcal{P} = \mathcal{BPP}$?) Historically, guided by progress in designing efficient randomized algorithms, most researchers believed that randomness does help. Research over the past 25 years on (time-bounded) *derandomization* has now led many to change their views; the consensus nowadays is that randomization does not help.[1]

One natural approach to derandomize algorithms is to use a *pseudorandom generator* (PRG) that expands a small, truly random input into a larger, random-looking output. In the next section we define PRGs and then describe their application to derandomization. The remainder of these notes will focus on constructing a PRG based on a (plausible) complexity assumption.

2 Pseudorandom Generators

A pseudorandom generator G is a deterministic algorithm that expands a short input (often called a "seed") into a larger output. The output of G should "look random"; formally, $G(s)$ (for s chosen uniformly) should be indistinguishable from a uniform string of length $|G(s)|$. We give a formal definition next. (A word on notation: When we write $G : \{0,1\}^{\ell(t)} \to \{0,1\}^t$ we mean that for every integer t and every $s \in \{0,1\}^{\ell(t)}$, we have $|G(s)| = t$.)

Definition 1 *A function $G : \{0,1\}^{\ell(t)} \to \{0,1\}^t$ is a* (complexity-theoretic) pseudorandom generator *if G can be computed in exponential time (i.e., $G(s)$ can be computed in time $2^{O(|s|)}$) and if for all sufficiently large t the following holds: for any distinguisher (i.e., circuit) C of size at most t,*

$$\left| \Pr_{r \leftarrow \{0,1\}^t} [C(r) = 1] - \Pr_{s \leftarrow \{0,1\}^{\ell(t)}} [C(G(s)) = 1] \right| < 1/t.$$

It is worth pointing out several differences between the above definition and that of *cryptographic* pseudorandom generators. (Those who have not seen cryptographic PRGs can skip to the next section.) The primary difference is with respect to the *running time of the PRG* vs. the *running time*

[1] Note that even if randomness "does not help" from a complexity-theoretic standpoint, it may still be the case that it helps from an algorithmic standpoint. Namely, even if $\mathcal{P} = \mathcal{BPP}$ there may exist problems whose solution requires, say, deterministic quadratic time but randomized linear time.

of the distinguisher. Simplifying things a bit, in the cryptographic setting honest parties evaluate the PRG and an adversary plays the role of the distinguisher; we would like to keep the running time of the honest parties as small as possible, while simultaneously protecting them against the most powerful class of adversaries possible. In particular, we certainly want to offer protection against adversaries who are at least as powerful as the honest parties; thus, when defining a cryptographic PRG we will always want to consider distinguishers that run in time greater than the time to evaluate the PRG itself. In contrast, we will see that this is not needed for complexity-theoretic applications; thus, it is still meaningful in our context to consider PRGs where the distinguisher runs in less time than is required to evaluate the PRG.

We mention a few other differences; the reason for these differences should become clear in the following section:

- Here we only require that the distinguisher cannot distinguish the pseudorandom distribution from the uniform distribution "too well", i.e., with advantage better than $1/t$. In the cryptographic setting we require the distinguisher's advantage to be negligible.
- A complexity-theoretic PRG may require *exponential* time (in the input length) to compute. In the cryptographic setting, as noted above, evaluating the PRG should be as efficient as possible; we at least require the PRG to be computable in *polynomial* time.
- In the present definition we consider non-uniform distinguishers, while in the usual cryptographic setting one considers only uniform distinguishers.

With the exception of the last point, complexity-theoretic PRGs are weaker than cryptographic PRGs; thus they can be constructed from milder assumptions.

2.1 Using a PRG to Derandomize Algorithms

We now show how a complexity-theoretic PRG can be used to derandomize algorithms.

Theorem 1 *If there is a (complexity-theoretic) pseudorandom generator* $G : \{0,1\}^{\ell(t)} \to \{0,1\}^t$, *then* $\text{BPTIME}(t(n)) \subseteq \text{TIME}(2^{O(\ell(t^2(n)))})$.

Proof Fix a language $L \in \text{BPTIME}(t(n))$, and a probabilistic algorithm A running in time $T(n) = O(t(n))$ for which

$$\begin{aligned} x \in L &\Rightarrow \Pr[A(x) = 1] \geq 2/3 \\ x \notin L &\Rightarrow \Pr[A(x) = 1] \leq 1/3. \end{aligned}$$

We construct a deterministic algorithm B deciding L. We focus on input lengths n sufficiently large; the behavior of B on a finite number of shorter inputs can be hard-coded into the algorithm.

On input $x \in \{0,1\}^n$, algorithm B sets $t = t(n)$ and does:

1. For each $s \in \{0,1\}^{\ell(t^2)}$, compute $A(x; G(s))$. (Note that A uses at most $T(n)$ random bits, which is less than $|G(s)| = t^2(n)$ for n sufficiently large.)
2. Output the majority value computed in the previous step.

Each iteration of step 1 takes time[2] at most $2^{O(\ell(t^2(n)))} + T(n) = 2^{O(\ell(t^2(n)))}$, and there are $2^{\ell(t^2(n))}$ iterations; thus, B runs in time $2^{O(\ell(t^2(n)))}$. We now show that B correctly decides L.

Correctness of B follows once we show that

$$\begin{aligned} x \in L &\Rightarrow \Pr[A(x; G(s)) = 1] > 1/2 \\ x \notin L &\Rightarrow \Pr[A(x; G(s)) = 1] < 1/2, \end{aligned}$$

where the probabilities are over random choice of $s \in \{0,1\}^{\ell(t^2)}$. Fix $x \in \{0,1\}^n$ with $x \in L$. (The argument if $x \notin L$ is analogous.) Consider the distinguisher $C(\cdot) = A(x; \cdot)$. Since A runs in time $T(n)$, there is a circuit of size $o(T^2) = o(t^2)$ computing C. But then for n sufficiently large

$$\begin{aligned} \Pr[A(x; G(s)) = 1] &= \Pr[C(G(s)) = 1] \\ &> \Pr[C(r) = 1] - 1/t^2 \qquad \text{(by pseudorandomness of } G) \\ &= \Pr[A(x) = 1] - 1/t^2 \;\geq\; 2/3 - 1/t^2 > 1/2. \end{aligned}$$

This completes the proof. ■

It is worth noting that non-uniformity comes into play in the preceding proof because we want B to be correct on *all* inputs; if there exists an input x on which B is incorrect then we can "hard-wire" x into the distinguisher C. The theorem would hold even if we only required G to be indistinguishable for $T = O(t)$-time algorithms taking $n \leq T$ bits of advice. In a different direction, if we only required B to be correct for *efficiently sampleable* inputs then we could work with a uniform notion of PRGs.

Corollary 2 *If there is a (complexity-theoretic) pseudorandom generator $G : \{0,1\}^{\ell(t)} \to \{0,1\}^t$ with $\ell(t) = O(\log t)$, then $\mathcal{P} = \mathcal{BPP}$.*

Proof Take arbitrary $L \in \mathcal{BPP}$. Then $L \in \text{BPTIME}(n^c)$ for some constant c. By the previous theorem, $L \in \text{TIME}(2^{O(\ell(n^{2c}))}) = \text{TIME}(2^{O(\log n)}) = \text{TIME}(n^{O(1)}) \subset \mathcal{P}$. ■

3 The Nisan-Wigderson PRG

3.1 Some Preliminaries

We collect here some results that are used in the next section, but are tangential to the main thrust.

The first lemma follows by a standard hybrid argument.

Lemma 3 *Fix $G : \{0,1\}^\ell \to \{0,1\}^t$ and suppose there is a circuit C of size at most t such that*

$$\left| \Pr_{r \leftarrow \{0,1\}^t} [C(r) = 1] - \Pr_{x \leftarrow \{0,1\}^\ell} [C(G(x)) = 1] \right| \geq 1/t.$$

Then there exists an $i \in \{1, \ldots, t\}$ and a circuit C' of size at most t such that

$$\Pr_{x \leftarrow \{0,1\}^\ell} [C'\left(G(x)_1 \cdots G(x)_{i-1}\right) = G(x)_i] - \frac{1}{2} \geq 1/t^2.$$

I.e., C' can predict the ith bit of the output of G.

The next lemma is a standard fact we have seen before.

Lemma 4 *Any function $f : \{0,1\}^k \to \{0,1\}$ can be computed by a circuit of size at most 2^k.*

[2]Note that $\ell(t) = \Omega(\log t)$ (so $2^{O(\ell(t^2))} = \Omega(T)$); otherwise, there is a trivial distinguisher and G is not a PRG.

3.2 From Hardness to Randomness

We will construct a PRG starting from any "suitably hard" computational problem. The starting point here is simple: if a boolean function f is hard to compute (on average) for algorithms running in time t — we formalize this below — then, by definition, $x \| f(x)$ "looks random" to any t-time algorithm given x. This does not yet give a PRG, but at least indicates the intuition. We first formally define what it means for a function to be hard.

Definition 2 *A function $f : \{0,1\}^m \to \{0,1\}$ is* S-hard *if for all circuits C of size at most S,*

$$\left|\Pr_{x \leftarrow \{0,1\}^m}[C(x) = f(x)] - 1/2\right| < 1/S\,.$$

The key to the construction of a PRG is a combinatorial object called a *design*.

Definition 3 *Fix integers k, m, ℓ. A collection of sets $\{S_1, \ldots, S_t\}$ with $S_i \subset \{1, \ldots, \ell\}$ is a* (k, m)-design *if (1) $|S_i| = m$ for all i, and (2) $|S_i \cap S_j| \leq k$ for all $i \neq j$.*

We can specify a set system $\{S_1, \ldots, S_t\}$ with $S_i \subseteq \{1, \ldots, \ell\}$ by a $t \times \ell$ matrix, where row i of the matrix is the characteristic vector for S_i. We say such a matrix A is a (k, m)-design if the corresponding set system is.

Given a function $f : \{0,1\}^m \to \{0,1\}$, an ℓ-bit string $x = x_1 \cdots x_\ell$, and a set $S = \{i_1, \ldots, i_m\} \subset \{1, \ldots, \ell\}$, define $f_S(x) = f(x_{i_1} \cdots x_{i_m})$. Given a $t \times \ell$ matrix A corresponding to a set system $\{S_1, \ldots, S_t\}$ with $S_i \subset \{1, \ldots, \ell\}$, define $f_A : \{0,1\}^\ell \to \{0,1\}^t$ as

$$f_A(x) = f_{S_1}(x) \cdots f_{S_t}(x).$$

In the following theorem we construct a "PRG" $G : \{0,1\}^\ell \to \{0,1\}^t$ for some fixed values of ℓ, t. (It is not quite a PRG since it is not yet a construction for arbitrary outputs length t.) We will observe later that, as t varies, the construction is computable in exponential time as required by Definition 2.

Theorem 5 *Fix integers t, m, ℓ. Suppose $f : \{0,1\}^m \to \{0,1\}$ is t^2-hard, and let A be a $t \times \ell$ matrix that is a $(\log t, m)$-design. Let $f_A : \{0,1\}^\ell \to \{0,1\}^t$ be as above. Then for all circuits C of size at most t we have*

$$\left|\Pr_{r \leftarrow \{0,1\}^t}[C(r) = 1] - \Pr_{x \leftarrow \{0,1\}^\ell}\left[C\left(f_A(x)\right) = 1\right]\right| < 1/t. \tag{1}$$

Proof Denote the design corresponding to A by $\{S_1, \ldots, S_t\}$. Fix a circuit C of size at most t, and assume toward a contradiction that (1) does not hold. By Lemma 3, this implies the existence of an $i \in \{1, \ldots, t\}$ and a circuit C' of size at most t for which

$$\Pr_{x \leftarrow \{0,1\}^\ell}[C'\left(f_{S_1}(x) \cdots f_{S_{i-1}}(x)\right) = f_{S_i}(x)] - \frac{1}{2} \geq 1/t^2. \tag{2}$$

That is, C' can predict $f_{S_i}(x)$ given $f_{S_1}(x), \ldots, f_{S_{i-1}}(x)$. We construct a circuit D of size at most t^2 that computes f with probability better than $1/2 + 1/t^2$, contradicting the assumed hardness of f.

For notational convenience, let us assume that $S_i = \{1, \ldots, m\}$. Rewriting (2), we have

$$\Pr_{x_1, \ldots, x_\ell \leftarrow \{0,1\}}[C'\left(f_{S_1}(x) \cdots f_{S_{i-1}}(x)\right) = f(x_1 \cdots x_m)] - \frac{1}{2} \geq 1/t^2\,,$$

where $x = x_1 \cdots x_\ell$. By a standard averaging argument, this implies that there exist some fixed values $\bar{x}_{m+1}, \ldots, \bar{x}_\ell$ for the variables $x_{m+1}, \ldots, x_\ell$ for which

$$\Pr_{x_1,\ldots,x_m \leftarrow \{0,1\}}[C'\left(f_{S_1}(x) \cdots f_{S_{i-1}}(x)\right) = f(x_1 \cdots x_m)] - \frac{1}{2} \geq 1/t^2\,,$$

where now $x = x_1 \cdots x_m \bar{x}_{m+1} \cdots \bar{x}_\ell$. We can express C' as a function D of $x_1, \ldots, x_m$ only by defining $D(x_1 \cdots x_m) = C'\left(f_{S_1}(x) \cdots f_{S_{i-1}}(x)\right)$ (with x as just defined). The size of D is at most the size of C' plus the sizes of the circuits required to compute all the $f_{S_j}(x)$. To get an upper bound on the latter, we use the fact that A is a $(\log t, m)$-design. This implies that each $f_{S_j}(x)$ is a function of at most $\log t$ of the bits $x_1, \ldots, x_m$ (since S_j intersects $S_i = \{1, \ldots, m\}$ in at most $\log t$ positions). Thus, by Lemma 4, each f_{S_i} can be computed using at most t gates; hence D requires at most $t + (t-1) \cdot t = t^2$ gates. This gives the desired contradiction, and completes the proof. ■

To finish the construction we need only show how to build a design with the required parameters. Treating the hard function f as being given, we have some fixed t, m and we want a $(\log t, m)$-design $\{S_1, \ldots, S_t\}$ with $S_i \subset \{1, \ldots, \ell\}$ and where $\ell = \ell(t, m)$ is as small as possible. For $\log t \leq m \leq t$, designs with $\ell = O(m^2)$ exist; see [2]. For $m = O(\log t)$, a better construction is possible.

Theorem 6 *Fix a constant c. There is an algorithm that, given t, constructs a $(\log t, c \log t)$-design $\{S_1, \ldots, S_t\}$ with $S_i \subset \{1, \ldots, \ell\}$ and $\ell = O(\log t)$. The algorithm runs in time $\mathsf{poly}(t)$.*

Proof Define $m = c \log t$, and set $\ell = d \log t$ where the exact constant d can be derived from the analysis that follows. A greedy algorithm, where we exhaustively search for the next set S_i (with the required properties) given $S_1, \ldots, S_{i-1}$, works. To see this, look at the worst case where $S_1, \ldots, S_{t-1}$ have all been fixed. Consider a random subset $S_t \subset \{1, \ldots, \ell\}$ of size m. The expected number of points in which S_t and, say, S_1 intersect is $m^2/\ell = O(\log t)$; setting d appropriately, the probability that they intersect in more than $\log t$ points is at most $1/t$. A union bound over the $t-1$ sets that have already been fixed shows that the probability that any one of the existing sets intersects S_t in more than $\log t$ points is less than 1; thus, there exists a set S_t that intersects each of the fixed sets in at most $\log t$ points.

Each set can be chosen in time $\mathsf{poly}(t)$ (because $\ell = O(\log t)$, we can enumerate all m-size subsets of $\{1, \ldots, \ell\}$ in time $\mathsf{poly}(t)$); since we choose t sets, the overall running time of the algorithm is polynomial in t. ■

The only remaining piece is to let t vary, and observe conditions under which the above pseudorandom generator can be computed in the required time. We say a function $f : \{0,1\}^* \to \{0,1\}$ is $S(n)$-hard if for all sufficiently large n the function f restricted to inputs of size n is $S(n)$-hard.

Corollary 7 *Suppose there is a function f that can be computed in time $2^{\alpha n}$ but is $2^{\beta n}$-hard for some constants α, β. Then there is a pseudorandom generator $G : \{0,1\}^{\ell(t)} \to \{0,1\}^t$ with $\ell(t) = O(\log t)$, and $\mathcal{P} = \mathcal{BPP}$.*

Proof Set $c = 2/\beta$, and let $\ell(t) = O(\log t)$ be as obtained using Theorem 6 for this value of c.

Define G as follows: On input $x \in \{0,1\}^\ell$, where $\ell = \ell(t)$ for some t, set $m = c \log t$ and let $f^{(m)}$ denote the restriction of f to inputs of length m. Do:

1. Construct a $(\log t, m)$-design $\{S_1, \ldots, S_t\}$ with $S_i \subset \{1, \ldots, \ell\}$. Let A be the matrix corresponding to this design.

2. Compute $f_A^{(m)}(x)$.

By the assumptions of the theorem, for all sufficiently large t the function $f^{(m)}$ is t^2-hard. Thus, Theorem 5 implies that the output of G is pseudorandom and all that is left is to analyze the running time of G. By Theorem 6, step 1 can be done in $\mathsf{poly}(t) = 2^{O(\ell)}$ time. Step 2 requires $t = 2^{O(\ell)}$ evaluations of $f^{(m)}$, and the assumptions of the theorem imply that each such evaluation can be done in time $2^{O(m)} = 2^{O(\ell)}$. We thus see that the entire computation of G can be done in time $2^{O(\ell)}$, as required. ■

Bibliographic Notes

The results here are due to Nisan and Wigderson [2], whose paper is very readable. A good starting point for subsequent developments in this area is [1, Chapter 20].

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.

[2] N. Nisan and A. Wigderson. Hardness vs. Randomness. *J. Computer & System Sciences* 49(2):149–167, 1994. Preliminary version in FOCS '88.