CS225: Pseudorandomness Prof. Salil Vadhan

Lecture 22: Black-Box PRG Constructions and Extractors

May 1, 2007

Based on scribe notes by Hamilton Chong.

1 Putting Together the PRG Construction

Let's put together the steps of the PRG construction we have seen so far. Starting from a worst-case hard function f, we can construct a pseudorandom generator G_f in the following steps.

$$f : \{0,1\}^{\ell} \to \{0,1\}$$

$$\Downarrow \ (1)$$

$$\hat{f} : \{0,1\}^{\hat{\ell}} \to \{0,1\}$$

$$\Downarrow \ (2)$$

$$G_f : \{0,1\}^{O(\hat{\ell}^2/\log m)} \to \{0,1\}^m$$

$\hat{f}$ is the encoding of f in a locally $(1/2-\varepsilon/m)$-decodable code (Reed-Muller/Low-Degree Extension concatenated with Hadamard). This yields $\hat{\ell} = O(\ell + \log(m/\varepsilon))$. (In our current application to PRGs, the setting of parameters will have $\log(m/\varepsilon) < \ell$, so $\hat{\ell} = O(\ell)$, but later we will refer to the more general parametrization.) G_f is obtained by the Nisan–Wigderson PRG construction using an $(\hat{\ell}, \log m)$ design.

$\hat{f}$ can be evaluated in time $2^{O(\ell)}$ given the truth table of f. And G_f can be evaluated in time $\mathrm{poly}(\hat{\ell}, m)$ plus m evaluations of $\hat{f}$. Thus, if $f \in \mathbf{E}$, then G_f is computable in time $\mathrm{poly}(2^{\ell}, m)$.

The correctness of the construction is proved via reduction. Given an efficient nonuniform test T distinguishing the output of G_f from uniform with advantage ε, we obtain a nonuniform algorithm $\hat{A}$ computing $\hat{f}$ on random inputs with error probability $1/2 + \varepsilon/m$, with $\mathrm{time}(\hat{A}) = \mathrm{time}(T) + \mathrm{poly}(m)$. Applying the local $(1/2 - \varepsilon/m)$ decoding algorithm to $\hat{A}$, we get a nonuniform algorithm A computing f on all inputs, with $\mathrm{time}(A) = \mathrm{poly}(\ell, m/\varepsilon) \cdot \mathrm{time}(\hat{A}) = \mathrm{poly}(\ell, m/\varepsilon) \cdot \mathrm{time}(T)$.

Thus, if f is worst-case hard for nonuniform time $t = t(\ell)$, we have a contradiction provided $\mathrm{time}(T), \ell, m, 1/\varepsilon < t^{1/c}$ for a sufficiently large constant c.

Theorem 1 *If* $\mathbf{E}$ *has nonuniform worst-case hardness* ℓ^c $\forall c$,[1]

$$\mathbf{BPP} \subseteq \mathbf{SUBEXP}$$

- *If* $\mathbf{E}$ *has nonuniform worst-case hardness* $2^{\ell^{\Omega(1)}}$, $\mathbf{BPP} \subseteq \tilde{\mathbf{P}}$
- *If* $\mathbf{E}$ *has nonuniform worst-case hardness* $2^{\Omega(\ell)}$, $\mathbf{BPP} = \mathbf{P}$

[1]Equivalently, **E** or **EXP** is not contained in **i.o.-P/poly**, where 'i.o.' means for infinitely many input lengths.

This result provides strong evidence that **BPP** is close to **P**.

Throughout our analysis, we've been using nonuniformity a lot, but it is important to stress that these constructions give us *uniform derandomizations*. That is, given a uniform algorithm computing f in time $2^{O(\ell)}$, we get a uniform algorithm computing G_f in time $2^{O(\ell)}$, and thus a uniform derandomization of **BPP**. The nonuniformity comes into play only when we talk about the hardness of our functions and the type of tests our PRGs fool. In particular, this is a very different result from the simply proven result we saw earlier in the class that $\mathbf{BPP} \subset \mathbf{P/poly}$.

Theorem 1 says that circuit lower bounds (for **E**) imply derandomization of **BPP**. An interesting question is whether the converse is true. As you will see on Problem Set 6, it can be shown that pseudorandom generators that fool nonuniform algorithms imply circuit lower bounds. Moreover, this relationship between circuit lower bounds for **E** and pseudorandom generators is quantitatively very tight. The only real looseness in the relationship we have seen is the fact that our PRG has seed length $O(\ell^2/\log m) = O(\ell^2/\log t)$ for hardness t, but it is known how to achieve the 'correct' bound of $O(\ell)$ with a different construction.

However, this still leaves open the question of whether *derandomization* of **BPP**, which could conceivably be done without using pseudorandom generators, is equivalent to circuit lower bounds. On one hand, there are results showing that nontrivial 'average-case' derandomizations of **BPP** follow from the uniform hardness assumption $\mathbf{EXP} \neq \mathbf{BPP}$. On the other hand, there are results showing that $\mathbf{BPP} = \mathbf{P}$ imply circuit lower bounds for **NEXP**, nondeterministic exponential time.

2 Black-Box Constructions

Like we observed for the Nisan–Wigderson generator, this construction is very general. The construction is well-defined for *every* function f; the only place that we use the fact that f is in **E** is to deduce that G_f is computable in **E**. And the reduction works for works for every test T. The only place we use the fact that T is an efficient nonuniform algorithm is to deduce that the algorithm A computing f, which uses T as a subroutine, is an efficient nonuniform algorithm.

A construction of this type is referred to as a *black-box construction*, because it only uses the assumed function f and the 'adversary' T as 'black boxes'. Formally:

Definition 2 *Let $G^f : \{0,1\}^d \to \{0,1\}^m$ be an algorithm that is defined for every oracle $f : \{0,1\}^\ell \to \{0,1\}$. We say that G is a (t,k,ε)* black-box PRG construction *if there is an oracle algorithm R, called the* reduction, *running in time t such that for every function $T : \{0,1\}^m \to \{0,1\}$ such that*

$$|\Pr[T(G(U_d)) = 1] - \Pr[T(U_m) = 1]| > \varepsilon,$$

there exists an advice string $z \in \{0,1\}^k$ such that

$$\forall x \qquad R^T(x,z) = f(x)$$

What we have shown above is the following:

Theorem 3 *For every $\ell, m \in \mathbb{N}$, $\varepsilon > 0$, there is a (t,t,ε) black-box PRG construction $G^f : \{0,1\}^d \to \{0,1\}^m$ expecting an oracle $f : \{0,1\}^\ell \to \{0,1\}$ with $t = \mathrm{poly}(m,\ell,1/\varepsilon)$, $d = O((\ell + \log(1/\varepsilon))^2/\log m)$, and G^f computable in time $2^{O(\ell)}$.*

Most — but not all — results in complexity theory are 'black-box' in a similar sense. The reason is that it seems difficult to make use of an efficient algorithm in any way other than to run it. For those familiar with the term, black-box constructions are always *relativizing*, meaning that they work in the presence of any additional oracle.

Why is this generality interesting?

- It allows us to construct pseudorandom generators that fool randomized classes $\mathcal{C}$ that are higher than **BPP**, under the assumption that **E** cannot be solved efficiently by nonuniform algorithms with an oracle for $\mathcal{C}$. In particular, this provides evidence that **AM**, a randomized version of **NP**, actually equals **NP**. (**AM** contains important problems, like GRAPH NONISOMORPHISM, that are not known to be in **NP**.) See Section 4 below.

- Black-box constructions typically have information-theoretic interpretations. In particular, we will see below how any black-box construction of pseudorandom generators gives rise to an extractor.

3 Extractors vs. PRGs

In this section, we will see a surprising connection between extractors and pseudorandom generators and use it to get a very good extractor from our PRG construction. Why is the connection surprising? So far, we have seen extractors as information-theoretic objects, which we constructed and studied using techniques from combinatorics and probability. And pseudorandom generators are inherently computational objects, which we constructed based on constructed using complexity-theoretic assumptions and reasoned about using efficient reductions. But now let's look at both objects in a way that starts to identify their similarity:

PRG	Extractor
If f is "hard" then $G^f(U_d)$ is pseudorandom	If X is k-source (with enough min-entropy) then $\mathrm{Ext}(X, U_d)$ is ϵ-close to U_m

There are still two noticeable differences. First, syntactically the extractor takes two inputs (a weak random source and a seed), whereas the generator takes only a seed. But if we think of the boolean function f used by the generator not as fixed, but as another input, we fix that problem.

Idea: Define $\mathrm{Ext}(f, y) \stackrel{\text{def}}{=} G^f(y)$.

Here, we view the n-bit string from the source as the truth table of a function $f : \{0,1\}^{\log n} \longrightarrow \{0,1\}$. That way the function input to the PRG matches the n bit source given to the extractor.

But now we run into the second difference. To show that Ext is an extractor, we need to prove that its output is *statistically* close to uniform, but G_f is only "designed" to give an output that is computationally indistinguishable from uniform. However, note that the notion of a black-box construction allows us to reason about *any* statistical test.

Indeed, we have the following:

Theorem 4 *If $G^f : \{0,1\}^d \to \{0,1\}$ is an (∞, k, ε) black-box pseudorandom generator construction, then $\mathrm{Ext}(f, y) = G^f(y)$ is a $(k + \log(1/\varepsilon), 2\varepsilon)$-extractor. Conversely, if $\mathrm{Ext}(f, y)$ is a (k, ε)-extractor, then G^f is an $(\infty, k+1, \varepsilon)$ black-box pseudorandom generator construction.*

Proof: ∎

Combining this with Theorem 3, we get:

Theorem 5 *For every* $n, k \in \mathbb{N}$*, there is an explicit* $(k, 1/m)$*-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \longrightarrow \{0,1\}^m$ *with* $m = 1/\varepsilon = (k/\mathrm{polylog}(n))^{\Omega(1)}$ *and* $d = O((\log^2 n)/(\log k))$.

Note that when $k = n^{\Omega(1)}$, we have $d = O(\log n)$ and $m = n^{\Omega(1)}$, and the extractor is sufficient for simulating **BPP** with any weak random source. It is important to stress that this construction gives us an *unconditional* extractor. We do not have to assume a hard function because the function comes from7 the source.

The parameters of the extractor are not as good as the one we saw earlier (which achieved $d = O(\log(n/\varepsilon))$ and $m = \Omega(k)$), but the construction is much simpler. It looks like:

$$\mathrm{Ext}(f, y) = \hat{f}(y|_{S_1}) \cdots \hat{f}(y|_{S_m}),$$

where $\hat{f}$ is the encoding of f in a list-decodable error-correcting code.

The output length can be improved to $m = k^{1-\delta}$ for an arbitrarily small constant δ by noting that the running time t of our reduction does not matter for the extractor, only the number k of advice bits used does. Thus we should more careful in bounding k, instead of equating it with t with the running time, as we have been doing. Specifically, if we allow the intersection size used in the design to be a parameter a, then the amount of advice we need is

$$|z| = m \cdot 2^a + O\left(\log \frac{m}{\epsilon}\right)$$

Recall that a is the size of the intersection between sets of our design. The $m \cdot 2^a$ comes from having to nonuniformly provide truth tables in the NW reduction. The $O\left(\log(\frac{m}{\epsilon})\right)$ comes from picking the right codeword in the list-decoding (note: we don't need *efficient* list-decodability for the extractor). Also remember that from the design, our seed length is $d = O\left(\frac{\log^2(n/\varepsilon)}{a}\right)$. Taking $a = \delta \log k$ gives the claimed result. Thus we have:

Theorem 6 *For every* constant $\delta > 0$*, and every* $n, k \in \mathbb{N}$*,* $\varepsilon > 0$*, there is an explicit* (k, ε)*-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with* $m = k^{1-\delta}$ *and* $d = O((\log^2(n/\varepsilon))/\log k)$.

(The actual output length obtained is $m = (k - O(\log(m/\varepsilon)))^{1-\delta}$, but this is at least $k^{1-\delta'}$ except when $k = O(\log(m/\varepsilon))$. In that case, we can simply output the seed (since we are not claiming a strong extractor here).)

Computational World	Information-theoretic World
(black-box) PRG constructions	Extractors
$f : \{0,1\}^\ell \to \{0,1\}$ nonuniform hardness	random source with high min-entropy
pseudorandom	ε-close to uniform
worst-case to average-case hardness conversion	list-decodable error-correcting codes
proof of correctness requires *efficient* (local) decoding	*inefficient* decoding is ok for extractors

Additional notes:

1. Since the discovery of these connections, work on PRGs and extractors have progressed together.
2. The decoding framework also gives a nice way of thinking about extractors. Specifically, we can prove the correctness of an extractor algorithmically, by describing a decoding procedure. As computer scientists, we find this to be a very natural way of thinking.
3. This also helps identify which aspects and limitations of our PRG constructions are present for information-theoretic reasons, and which are due to computational efficiency concerns. In particular, we see that nonuniform hardness assumptions are necessary as long as we work with black-box PRG constructions. Why?

4 Derandomizing AM

AM is a probabilistic version of **NP**. $L \in \mathbf{AM}$ if $\exists$ a polynomial time algorithm V such that :

$$\text{completeness: } x \in L \Rightarrow \Pr_r[\exists y \text{ } s.t. \text{ } V(x,r,y) = 1] \geq \frac{2}{3}$$

$$\text{soundness: } x \notin L \Rightarrow \Pr_r[\exists y \text{ } s.t. \text{ } V(x,r,y) = 1] \leq \frac{1}{3},$$

where r and y are bitstrings of length $\text{poly}(|x|)$. (Side remark: it turns out that, for **AM** (unlike **BPP**), it is known that the fractions at the end can be replaced by 1 and $\frac{1}{2}$ respectively without loss of generality.) We can view **AM** as capturing interactive proof systems of the following form:

$$\begin{array}{ccccc} & & x & & \\ & \swarrow & & \searrow & \\ P & & \longleftarrow r & & V \\ & & y \longrightarrow & & \\ & & & & \text{output } 0/1 \end{array}$$

It is known that Graph Nonisomorphism is in **AM**, even though it is not known to be contained in **NP**.

A natural question about derandomization is: Does $\mathbf{AM} = \mathbf{NP}$?

As usual, our idea approach will be to replace randomness with pseudorandomness. More specifically, we'll replace r with $G_f(s)$ for a "hard" f. The hope is that the two probabilities ($\frac{2}{3}$ and $\frac{1}{3}$) don't change too much. If these probabilities do not change much and G has seed length $d = d(n)$, we get $L \in \mathbf{NTIME}(2^d \cdot \text{poly}(n))$ because

$$L = \{x \mid \exists y_1, \ldots, y_{2^d} \text{ s.t. } \text{maj}_s\{V(x, G_f(s), y_s)\} = 1\}.$$

In particular, if G_f has a logarithmic seed length, this implies $L \in \mathbf{NP}$.

Suppose that deciding L in this manner doesn't work. Then we want to break the PRG. For each x where this doesn't work, we define:

$$T_x(r) = \begin{cases} 1 & \text{if } \exists y \text{ s.t. } V(x,r,y) = 1 \\ 0 & \text{otherwise} \end{cases}$$

This isn't an efficient algorithm in the usual sense, but it is an efficient *nondeterministic* one (with x hardwired in nonuniformly). If the derandomization fails on x, then T_x distinguises $G_f(U_d)$ from U_m with constant advantage. By the definition of black-box reduction, there is an efficient nonuniform algorithm A such that A^T computes f everywhere. We note that A^T can be implemented efficiently given an oracle for any **NP**-complete problem (e.g. SAT). Thus we get a contradiction if we start with a function f that cannot be computed by an efficient **NP**-oracle algorithms.

Theorem 7 *If* $\mathbf{E}$ *has nonuniform worst-case hardness* $2^{\Omega(\ell)}$ *for* **NP***-oracle algorithms, then* $\mathbf{AM} = \mathbf{NP}$.

This result was quite surprising when discovered because it was the first evidence that $\mathbf{AM} = \mathbf{NP}$. It suggests that perhaps Graph Nonisomorphism$\in$ **NP**. One psychological barrier that delayed the discovery of this result was a misconception that one cannot have PRGs that fool nondeterministic adversaries because they can just guess the seed. However, since we're now talking about PRGs that can take more time than the tests they fool, a test that can nondeterministically guess the seed still doesn't have enough time to actually run the generator.