CS760 Topics in Computational Complexity **2024-25 Sem I**

Lecture 19: 11-10-2024

Scribe: Priyanshu Singh *Lecturer: Rohit Gurjar*

Derandomization

It is an open question to determine whether every problem with an efficient randomized algorithm also has an efficient deterministic algorithm. In other words, whether

- BPP $\stackrel{?}{=}$ P
- RP $\stackrel{?}{=}$ P

It is widely believe that BPP = RP = P. The reason for this belief is the known construction of pseudorandom generators from hard functions. The essential idea is that output of a complicated function will look random to any algorithm using limited resources. Here, by hard functions we mean Boolean functions which cannot be computed by small Boolean circuits.

Showing hard functions. We know that $\exists f : \{0,1\}^n \to \{0,1\}$ which requires exponential-sized circuits. We wish to find such explicit functions. Explicit meaning function which can be computed in exponential time (or some smaller complexity class). The current best known circuit lower bound for explicit functions is $\Omega(n)$.

For the purpose of pseudorandom generators, what we aim to achieve is: a function $f \in$ EXP that has no circuit of size $2^{n/10}$. If we can find such a function, we can use f to construct pseudorandom generators (PRGs), which will imply BPP = P.

Pseudorandomness

Let D be a distribution on $\{0,1\}^m$. We say that D is pseudorandom for a randomized algorithm A, if:

$$\forall x \in \{0,1\}^m \quad \left| \Pr_{r \sim U_m}[A(x,r)=1] - \Pr_{r \sim D}[A(x,r)=1] \right| \leq \frac{1}{10}$$

Thus, we can say, if for $r \sim U_m$, A is correct with probability $\geq \frac{2}{3}$, then for $r \sim D$, A is correct with probability $\geq \frac{2}{3} - \frac{1}{10}$.

Trivial derandomization: A trivial deterministic equivalent for a randomized algorithm A is as follows: If A uses ℓ random bits, then consider an algorithm A' that runs A on each possible choice of random bits, i.e., 2^ℓ choices, and then output the majority answer. In particular, a randomized algorithm that uses $O(\log n)$ random bits can be simulated deterministically, with runtime blowing up only by a polynomial factor.

So, the idea is to construct a pseudorandom distribution that is generated by using only $O(\log n)$ random bits. More precisely, we want to design an algorithm G that takes $O(\log n)$ bit input and outputs n bits. When $r \in \{0,1\}^{O(\log n)}$ is chosen randomly, the distribution of $G(r)$ should look similar to a uniform distribution on n bits (to a bounded resource algorithm A). If the algorithm A runs in time $O(n^2)$ time, then we can expect to fool it by a generator algorithm G that runs in time sufficiently larger than $O(n^2)$.

Note that when we say that we have to fool an algorithm A, we mean we have to fool it with all possible inputs x of a particular size. Suppose an algorithm $A(x,r)$ takes input x and random bits r, then we can represent $A(x,\cdot)$ by a circuit of polynomial size (polynomial in the input size and running time of A). That is,

$$\text{Algorithm} + \text{Input} \to \text{Circuit}$$

To see this, recall that any algorithm $A(x, r)$ can be converted into a circuit C whose size is proportional to the running time. This circuit C has $|x| + |r|$ input gates. For any given x, we can fix the corresponding bits in circuit C and obtain another circuit C', which takes only r as input. The size of C' is clearly bounded by the size of C. To summarize, our pseudorandom generator will need to fool a circuit of polynomial size.

Definition 19.1 (Pseudorandom distribution)**.** *A distribution D on $\{0,1\}^m$ is said to be (S, ϵ)-pseudorandom if for all circuits C on m input bits and size $\leq S$:*

$$\left| \Pr_{y \sim U_m}[C(y) = 1] - \Pr_{y \sim D}[C(y) = 1] \right| \leq \epsilon$$

We refer the above as "distribution D fooling all circuits of size at most S". Now, we define pseudorandom generators.

Definition 19.2 (Pseudorandom Generator (PRG))**.** *An algorithm G with $|G(y)| = m(|y|)$ is called an $m(\ell)$-PRG (you can think of, $m(\ell) = 2^{\ell/10}$) if the following holds:*

1. *For all ℓ, for any $r \in \{0,1\}^\ell$, $G(r)$ can be computed in $2^{O(\ell)}$ time.*
2. *$G(U_\ell)$ is $(m(\ell)^3, 1/10)$-pseudorandom.*

Remark. There exists a circuit of size around $2^\ell \approx m(\ell)^{10}$ which can distinguish between U_m and $G(U_\ell)$. *How?* Take a circuit C' such that C' is precisely 1 on the range of G (i.e., hardcode the values of $G(U_\ell)$). Then, we have the following:

$$\Pr_{y \sim U_\ell}[C'(G(y)) = 1] = 1$$

and

$$\Pr_{z \sim U_m}[C'(z) = 1] \leq \frac{2^\ell}{2^m}$$

(it is an inequality because G may map different ℓ bits strings to the same m bit string.)

Theorem 19.3. *Existence of $2^{\ell/10}$-PRG Implies BPP = P.*

Proof. We know that $10 \log m$ random bits can be converted to m pseudorandom bits using the pseudorandom generator G. Let $A(x, r)$ be a randomized algorithm that for a particular input size takes m random bits r and runs in time m^2. For each $y \in \{0,1\}^{10 \log m}$, we will run A with $G(y)$ as the choice of random bits. Finally, we take the majority answer. The total time complexity becomes

$$O(2^{10 \log m}) \times (2^{O(\log m)} + \text{runtime}(A)),$$

which remains polynomial time.

We can say that $A(x, \cdot)$ has a circuit of size at most m^3. Then by the definition of PRG, we know that $\Pr_{r \in U_m}[A(x, r)] \approx \Pr_{y \in U_{10 \log m}}[A(x, y)]$. Hence, we will get the correct answer.

Note that this derandomization works, even if the running time of the algorithm A was a larger polynomial, say $O(n^6)$ for input size n. In that case, we simply choose m to be large enough, that is, $m^3 = n^6$, which implies $m = n^2$. If algorithm A actually requires fewer random bits than $m = n^2$, then we simply ignore the excess pseudorandom bits generated by G.

□

Definitions of Hardness

As mentioned earlier, we can construct pseudorandom generators via some hard functions – which cannot be computed by small circuits. Let us formally define the hardness of functions.

Definition 19.4 (Worst Case Hardness)**.** *For a function $f : \{0,1\}^n \to \{0,1\}$, the worst case hardness, $H_{worst}(f)$ is $S - 1$, where S is the minimum possible size of a circuit C for which,*

$$\textit{for all } x \in \{0,1\}^n, C(x) = f(x).$$

For PRGs we will actually need functions that are hard on average, as defined below.

Definition 19.5 (Average Case Hardness)**.** *For a function $f : \{0,1\}^n \to \{0,1\}$, the average case hardness, $H_{avg}(f)$ is $S-1$, where S is the minimum number such that there is a circuit C of size S for which:*

$$\Pr_{r \sim U_n}[C(x) = f(x)] \geq \frac{1}{2} + \frac{1}{S}.$$