CS221: Computational Complexity Prof. Salil Vadhan

Lecture 18: Randomized Computation and Circuit Complexity

11/04 *Scribe: Ethan Abraham*

Contents

1 Plan for Lecture **1**

2 Recap **1**

3 Randomized Complexity **1**

4 Circuit Complexity **4**

1 Plan for Lecture

- Randomized Computation (Papadimitrou §11.1 - §11.2, Thm. 17.12)
- Circuit Complexity (Papadimitrou §11.4)

2 Recap

Definition 1 $L \in \mathbf{BPP}$ *if* $\exists$ *a probabilistic polynomial-time Turing Machine* M *such that*

- $x \in L \Rightarrow \Pr[M(x) accepts] \geq 2/3$
- $x \notin L \Rightarrow \Pr[M(x) rejects] \leq 1/3$.

We also recall the following lemma:

Lemma 2 (BPP Amplification Lemma) *If* $L \in \mathbf{BPP}$, *then for all polynomials* p, L *has a* **BPP** *algorithm with error probability* $\leq 2^{-p(n)}$.

3 Randomized Complexity

3.1 Bounds on BPP

What do we know?

- $\mathbf{P} \subseteq \mathbf{RP}, \mathbf{P} \subseteq \mathbf{co\text{-}RP}$
- $\mathbf{RP} \subseteq \mathbf{NP}, \mathbf{RP} \subseteq \mathbf{BPP}$
- $\mathbf{co\text{-}RP} \subseteq \mathbf{BPP}, \mathbf{co\text{-}RP} \subseteq \mathbf{co\text{-}NP}$

What don't we know?

- We don't know the relationship between **BPP** and **NP**.
- We don't even know that $\mathbf{BPP} \neq \mathbf{NEXP}$. This is downright insulting.

However, we do have the following theorem, that gives some bound on how large **BPP** can be:

Theorem 3 (Sipser) $\mathbf{BPP} \subseteq \mathbf{\Sigma_2 P} \cap \mathbf{\Pi_2 P}$

Based on our knowledge of the polynomial hierarchy, we have the following immediate corollary:

Corollary 4 $\mathbf{P} = \mathbf{NP} \Rightarrow \mathbf{BPP} = \mathbf{P} \Rightarrow \mathbf{BPP} \neq \mathbf{EXP}$, *so either* $\mathbf{P} \neq \mathbf{NP}$ *or* $\mathbf{BPP} \neq \mathbf{EXP}$.

Proof of Theorem (Lautemann): Suppose $L \in \mathbf{BPP}$. By the BPP Amplification Lemma, we can assume that L has a **BPP** algorithm with error probability 2^{-n}. Thus $\exists$ a polynomial relation $R \in \mathbf{P}$ and a polynomial $p(n)$ such that, for a random $r \xleftarrow{\mathrm{R}} \{0,1\}^{p(n)}$:

$$\begin{aligned} x \in L &\Rightarrow \Pr_r [R(x,r) = 1] \geq 1 - 2^{-n} \\ x \notin L &\Rightarrow \Pr_r [R(x,r) = 0] \leq 2^{-n} \end{aligned}$$

Basically, the relation is hard-coding the possible random choices into a prechosen random string. Let $A_x = \{r \in \{0,1\}^{p(n)} : R(x,r) = 1\}$. A_x represents which random choices will give an accepting computation. In very non-technical terms, if $x \in L$, A_x is a large subset of $\{0,1\}^{p(n)}$, while if

$x \notin L$, A_x is very small. We would like to find a **PH** algorithm to distinguish between these two cases. If $x \in L$, it seems likely that we could cover the whole space of $\{0,1\}^{p(n)}$ using only a few shifted copies of A_x, while if $x \notin L$, this should not be possible. Now we will make this rigorous.

Definition 5 *A copy of A_x is a translation via some string $t \in \{0,1\}^{p(n)}$, ie $A_x \oplus t = \{r \oplus t \,:\, r \in A_x\}$.*

Claim 6 (1) $x \in L \Rightarrow \exists t_1, t_2, \ldots, t_{p(n)} \in \{0,1\}^{p(n)} \,:\, \bigcup_i A_x \oplus t_i = \{0,1\}^{p(n)}$

Claim 7 (2) $x \notin L \Rightarrow \forall t_1, t_2, \ldots, t_n \in \{0,1\}^{p(n)}, \,:\, \bigcup_i A_x \oplus t_i \neq \{0,1\}^{p(n)}$

To obtain a $\mathbf{\Sigma_2 P}$ algorithm from these claims, we note $\bigcup_i A_x \oplus t_i = \{0,1\}^{p(n)}$ can be rewritten as

$$\left[\forall s \in \{0,1\}^{p(n)}, (s \in A_x \oplus t_1) \vee (s \in A_x \oplus t_2) \vee \cdots \vee (s \in A_x \oplus t_{p(n)})\right].$$

Definition 8 *For a subset $S \subseteq \{0,1\}^{p(n)}$, $\mu(S) = \frac{|S|}{2^{p(n)}}$*

Proof: [Claim 7] We know that $\mu(A_x) \leq 2^{-n}$. Thus, by the injectivity of $\oplus$,

$$\mu(\bigcup_i A_x \oplus t_i) \leq \sum_i \mu(A_x \oplus t_i) = \sum_i \mu(A_x) \leq p(n) \cdot \frac{1}{2^n} < 1$$

■

Proof: [Claim 6] This proof is slightly harder, and will use the probabilistic method. We will show that the $t_1, t_2, \ldots, t_{p(n)}$ exist by showing that a randomly chosen sequence of t_i satisfy the desired properties with non-zero probability. This proof, then, is non-constructive. So, fix some $s \in \{0,1\}^{p(n)}$. Then, since all the t_i are randomly chosen,

$$\Pr_{t_1,\ldots,t_{p(n)}}\left[s \notin \bigcup_i (A_x \oplus t_i)\right] \leq \left(\Pr_{t_i}\left[s \notin A_x \oplus t_i\right]\right)^{p(n)} \leq (\mu(A_x))^{p(n)} \leq 2^{-np(n)}$$

Thus,

$$\Pr_{t_1,\ldots,t_{p(n)}}\left[\exists s \notin \cup_i (A_x \oplus t_i)\right] \leq (\text{\# of choices for s}) \cdot 2^{-np(n)} = 2^{p(n)} \cdot 2^{-np(n)} < 1$$

Therefore some set of t_i's must have $\cup_i A_x \oplus t_i = \{0,1\}^{p(n)}$. ■

By the claims, we have $\mathbf{BPP} \subseteq \mathbf{\Sigma_2 P}$, and thus also $\mathbf{BPP} \subseteq \mathbf{\Pi_2 P}$, since **BPP** is closed under complement. ■

3.2 ZPP

We have defined **RP**, **co-RP**, and**BPP** with respect to probabilistic algorithms which *always* halt within a polynomial number of steps (a.k.a. *strict* polynomial time). We can also consider algorithms that have *expected* polynomial running time and zero error.

Definition 9 *$L \in$ **ZPP** $\Leftrightarrow \exists$ a probabilistic, error-free TM M for L and a polynomial p such that $\forall x, \mathrm{E}(T(M(x))) \leq p(|x|)$, where $T(M(x))$ is the random variable denoting the number of steps taken by M on x.*

Proposition 10 **ZPP** $=$ **RP** $\cap$ **co-RP**.

For a sketch of the proof, if we have an **RP** and a **co-RP** algorithm for some language L, we can just run both algorithms in parallel. If they halt on the same answer (accept or reject) that is the answer. This will happen "almost all" of the time. Sometimes, one will make a mistake, and they will halt on different answers. In this case, we just repeat the run, using different random choices.

4 Circuit Complexity

So far, we have spoken only about the complexity of functions defined on an infiinite set of inputs. Can we talk about the complexity of finite functions? If we have a function $f : \{0,1\}^n \rightarrow \{0,1\}$, we define the *circuit complexity* of f to be the number of gates (including input gates) in the smallest circuit computing f. This definition, we note, is dependent on the basis, the set of gates, that we use to construct circuits. Any universal one, ie one that allows us to construct every possible 1-ary and 2-ary function, can be used.

- $S = \{\wedge, \vee, \neg\}$, our usual basis.
- $B_2 = \{\text{all 2-ary functions}\}$
- $\{\vee, \oplus\}$, which represents multiplication and addition mod 2.
- $\{\text{nand}\}$

Although the choice of basis affects the circuit complexity, it affects it by at most a constant factor.