6.842 Randomness and Computation February 23, 2022

Lectures on Derandomization

Lecturer: Ronitt Rubinfield *Scribe: Yuchong Pan*

1 Randomized Complexity Class

Definition 1. A *language* is a subset of $\{0,1\}^*$.

Definition 2. P is a complexity class that consists of all languages L with a polynomial time deterministic algorithm A.

Definition 3. RP is a complexity class that consists of all languages L with a polynomial time probabilistic algorithm A such that

$$\begin{aligned} &\mathbb{P}[A \text{ accepts } x] \geq 1/2, && \text{if } x \in L, \\ &\mathbb{P}[A \text{ rejects } x] = 1, && \text{if } x \notin L, \end{aligned}$$

This is called 1-*sided error*.

Definition 4. BPP is a complexity class that consists of all languages L with a polynomial time probabilistic algorithm A such that

$$\begin{aligned} &\mathbb{P}[A \text{ accepts } x] \geq 2/3, && \text{if } x \in L, \\ &\mathbb{P}[A \text{ rejects } x] \geq 2/3, && \text{if } x \notin L, \end{aligned}$$

This is called 2-*sided error*.

2 Derandomization via Enumeration

Consider a problem L in BPP. Given a randomized algorithm A that decides L with running time $t(n)$ and $r(n) \leq t(n)$ random bits, we can define a deterministic algorithm in Algorithm 1 that decides L. By the definition of BPP, the majority answer is the correct answer. The running time of Algorithm 1 is $2^{r(n)} \cdot t(n)$.

```
run A on every possible random string of length r(n)
output the majority answer
```

Algorithm 1: A deterministic algorithm that derandomizes a randomized algorithm A with running time $t(n)$ and $r(n) \leq t(n)$ random bits.

Definition 5. $\mathrm{EXP} = \bigcup_c \mathrm{EXP}(2^{n^c})$.

Corollary 6. BPP $\subseteq$ EXP.

3 Pairwise Independence

3.1 Maximum Cut

The maximum cut problem is formulated as follows:

Problem 7 (maximum cut)**.** *Given a graph $G = (V, E)$, output a partition of V into S, T to maximize $|\{(u, v) : u \in S, v \in T\}|$, i.e., the size of the (S, T)-cut.*

The maximum cut problem is NP-hard. We give a randomized algorithm in Algorithm 2 that approximates the maximum cut problem.

flip coins $r_1, \ldots, r_n \in \{0, 1\}$ $(n = |V|)$
put vertex i on side r_i (i.e., if $r_i = 0$ put in S, else in T) for each $i \in [n]$ to get S, T

Algorithm 2: A randomized algorithm that approximates the maximum cut problem.

For each $(u, v) \in E$, let

$$\mathbb{1}_{(u,v)} = \begin{cases} 1, & \text{if } r_u \neq r_v \text{ (i.e., } (u,v) \text{ crosses the } (S,T)\text{-cut)}, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore,

$$\begin{aligned} \mathbb{E}[\text{cut size}] &= \mathbb{E}\left[\sum_{(u,v)\in E} \mathbb{1}_{(u,v)}\right] = \sum_{(u,v)\in E} \mathbb{E}\left[\mathbb{1}_{(u,v)}\right] \\ &= \sum_{(u,v)\in E} \mathbb{P}[(u,v) \text{ crosses cut}] \\ &= \sum_{(u,v)\in E} \mathbb{P}\left[(r_u = 0, r_v = 1) \text{ or } (r_u = 1, r_v = 0)\right] \quad \text{(using that } r_u, r_v \text{ are independent)} \\ &= \frac{|E|}{2}. \end{aligned}$$

This implies that there exists a cut of size at least $|E|/2$. By derandomization via enumeration, we try all 2^n possible settings of coins and pick the best cut.

The plan to obtain a faster derandomized algorithm is to find a subset of settings of $r_1, \ldots, r_n$ which suffices, so we instead enumerate over this smaller subset.

3.2 Pairwise Independence

Definition 8. Let T be a domain such that $|T| = t$. Let $X_1, \ldots, X_n$ be n random variables such that $X_i \in T$ for each $i \in [n]$. We say that $X_1, \ldots, X_n$ are

- *independent* if for all $b_1, \ldots, b_n \in T$, $\mathbb{P}[(X_1, \ldots, X_n) = (b_1, \ldots, b_n)] = 1/t^n$;
- *pairwise independent* if for all $i, j \in [n]$ with $i \neq j$ and $b_i, b_j \in T$, $\mathbb{P}[(X_i, X_j) = (b_i, b_j)] = 1/t^2$;
- *k-wise independent* if for all distinct $i_1, \ldots, i_k \in [n]$ and for all $b_{i_1}, \ldots, b_{i_k} \in T$, we have $\mathbb{P}[(X_{i_1}, \ldots, X_{i_k}) = (b_{i_1}, \ldots, b_{i_k})] = 1/t^k$.

total independence				pairwise independence				
probability	r_1	r_2	r_3	probability	r'_1	r'_2	r'_3	
1/8	0	0	0	1/4	0	0	0	00
1/8	0	0	1	1/4	0	1	1	01
1/8	0	1	1	1/4	1	0	1	10
1/8	1	0	1	1/4	1	1	0	11
1/8	1	0	0					
1/8	1	0	1					
1/8	1	1	0					
1/8	1	1	1					

Table 1: An example of pairwise independence.

Consider the example given in Table 1. Note that r'_1, r'_2, r'_3 are not independent because, e.g., $\mathbb{P}[r'_1 r'_2 r'_3 = 000] = 1/4 \neq 1/8$ and $\mathbb{P}[r'_1 r'_2 r'_3 = 010] = 0 \neq 1/8$. However, r'_1, r'_2, r'_3 are pairwise independent because $\mathbb{P}[r'_i r'_j = b_i b_j] = \mathbb{P}[r_i r_j = b_i b_j] = 1/4$ for all $i, j \in [3]$ with $i \neq j$ and for all $b_i, b_j \in \{0, 1\}$. Note that each row on the right half can be represented by two bits, as indicated in the last column.

A *randomness generator* takes m totally independent random bits $b_1, \ldots, b_m$ as input and outputs n pairwise independent random bits $r_1, \ldots, r_n$. Suppose that we have a randomness generator. Then we can derandomize Algorithm 2 as follows:

(i) Construct a randomized algorithm MC' which, given m totally independent random bits $b_1, \ldots, b_m$ and a graph G, generates n pairwise independent random bits $r_1, \ldots, r_n$ from $b_1, \ldots, b_m$, and uses the r_i's to run Algorithm 2.

(ii) Derandomize MC' via enumeration, i.e., for all choices of $b_1, \ldots, b_m$, run MC', and output the best cut.

Note that the running time of this derandomized algorithm is

$$2^m \times (\text{time for randomness generator} + \text{time for MC'})\,.$$

Therefore, if $m = O(\log n)$, then we have a deterministic polynomial time 2-approxmation algorithm for the maximum cut problem.

3.3 Generating Pairwise Independence Random Variables

We consider the case that the random variables are bits and the case that the random variables are integers in $\{0, \ldots, q-1\}$, where q is a prime.

Bits. We use Algorithm 3. The correctness of the algorithm will be proved in homework. Therefore, k truly random bits can generate $2^k - 1$ pairwise independent random bits, so $\log n$ truly random bits can generate $n-1$ pairwise independent random bits.

choose k truly random bits $b_1, \ldots, b_k$
foreach $S \subset [k]$ *such that* $S \neq \emptyset$ **do**
    $C_S \leftarrow \bigoplus_{i \in S} b_i$
output all C_S's

Algorithm 3: A randomness generator for bits.

Integers in $\{0, \ldots, q-1\}$***, where*** q ***is a prime.*** The first idea is that if q can be represented via ℓ bits, then we run Algorithm 3 for ℓ times. The resulting algorithm requires $O(\log q \cdot \log q)$ truly random bits, where the first $\log q$ becomes from Algorithm 3 and the second $\log q$ is the number of repetitions. Nevertheless, there exists an algorithm which requires $O(\log q)$ truly random bits only, given in Algorithm 4.

pick truly random integers $a, b \in \mathbb{Z}_q$
foreach $i \in \{0, \ldots, q-1\}$ **do**
    $r_i \leftarrow a \cdot i + b \mod q$
output $r_0, \ldots, r_{q-1}$

Algorithm 4: A randomness generator for integers in $\{0, \ldots, q-1\}$, where q is a prime.

Definition 9. A family $\mathcal{H} = \{h_i\}_{i \in I}$ of functions such that $h_i : [N] \to [M]$ for each $i \in I$ is said to be *pairwise independent* if h is uniformly random in H,

- for all $x \in [N]$, $h(x)$ is uniformly distributed in $[M]$;
- for all $x_1, x_2 \in [N]$ with $x_1 \neq x_2$, $(h(x_1), h(x_2))$ is uniformly distributed in $[M]^2$.

For each $a, b \in \mathbb{Z}_q$, let $h_{a,b} : \{0, \ldots, q-1\} \to \mathbb{Z}_q$ be defined by

$$h_{a,b}(x) = a \cdot x + b \mod q.$$

Then we can show that $\mathcal{H} = \{h_{a,b} : a, b \in \mathbb{Z}_q\}$ is pairwise independent. Indeed, for each $x_1, x_2 \in \{0, \ldots, q-1\}$ with $x_1 \neq x_2$ and for each $c, d \in \mathbb{Z}_q$,

$$\mathbb{P}_{a,b}\left[h_{a,b}(x_1) = c \wedge h_{a,b}(x_2) = d\right] = \mathbb{P}_{a,b}\left[ax_1 + b = c \wedge ax_2 + b = d\right] = \frac{1}{q^2}.$$

To see this, note that the above probability equals

$$\mathbb{P}_{a,b}\left[\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}\right] = \frac{1}{q^2},$$

because $x_1 \neq x_2$ implies $\det \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \end{pmatrix} \neq 0$ and hence has a unique solution.

3.4 Using Pairwise Independence to Improve Confidence

Let $\mathcal{A}$ be a randomized polynomial time algorithm for some language $L \in \mathrm{RP}$ which uses a random string r such that

- if $x \in L$, then $\mathbb{P}_r[A(x,r) = \text{accept}] \geq 1/2$;
- if $x \notin L$, then $\mathbb{P}_r[A(x,r) = \text{accept}] = 0$.

The goal is to reduce the confidence error.

The old way is to repeat $\mathcal{A}$ for k times with *new* random bits each time. If we ever see "accept", output "accept;" else output "reject." If $x \in L$, then $\mathbb{P}[\text{accept}] = 0$. If $x \notin L$, then $\mathbb{P}[\text{reject}] \leq (1 - 1/2)^k = 1/2^k$. This approach uses $O(k|r|)$ random bits.

1 pick $h \in_R H$
2 **foreach** $i \leftarrow 1, \ldots, 2^{k+2}$ **do**
3 $\quad r_i \leftarrow h(i)$
4 $\quad$ **if** $\mathcal{A}(x, r_i) = \textit{accept}$ **then**
5 $\quad\quad$ **return** *accept*
6 **return** *reject*

Algorithm 5: The sampling algorithm for 2-point sampling.

We introduce 2-*point sampling*. Assume that given a family $\mathcal{H}$ of pairwise independent fnuctions $[2^{k+2}] \to \{0,1\}^r$, we can pick a random $h \in \mathcal{H}$ with $O(k+r)$ random bits and $\text{poly}(k,r)$ time. The sampling algorithm is given in Algorithm 5.

We analyze the behavior of Algorithm 5. If $x \notin L$, then $\mathbb{P}[\text{accept}] = 0$. If $x \in L$, then misclassifying happens if we never see r_i such that $\mathcal{A}(x, r_i) = \text{accept}$. For each $i \in [2^{k+2}]$, let

$$\sigma(r_i) = \begin{cases} 0, & \text{if } \mathcal{A}(x, r_i) = \text{reject} \quad \text{(incorrect)}, \\ 1, & \text{otherwise} \quad \text{(correct)}. \end{cases}$$

Then $\mathbb{E}[\sigma(r_i)] = \mathbb{P}[\sigma(r_i) = 1] = \mathbb{P}[\text{accept}] \geq 1/2$. Let $\ell = 2^{k+2}$. Let $Y = \sum_{i=1}^{\ell} \sigma(r_i)$ be the number of correct answers. Then $\mathbb{E}[Y] \geq 2^{k+2}/2$, so $\mathbb{E}[Y/\ell] \geq 1/2$.

Lemma 10 (Chebychev's inequality)**.** *Let X be a random variable. Let $\mu = \mathbb{E}[X]$. Then*

$$\mathbb{P}[|X - \mu| \geq \varepsilon] \leq \frac{\text{Var}[X]}{\varepsilon^2}.$$

Lemma 11 (pairwise independence tail inequality)**.** *Let $X_1, \ldots, X_t \in [0,1]$ be pairwise independent random variables. Let $X = \sum_{i=1}^{t} X_i/t$. Let $\mu = \mathbb{E}[X]$. Then*

$$\mathbb{P}[|X - \mu| \geq \varepsilon] \leq \frac{1}{t\varepsilon^2}.$$

Since $\mathbb{E}[Y/\ell] \geq 1/2$, then Lemma 11 implies that

$$\mathbb{P}[\text{error}] = \mathbb{P}[Y = 0] = \mathbb{P}\left[\frac{Y}{\ell} = 0\right] \leq \mathbb{P}\left[\left|\frac{Y}{\ell} - \mathbb{E}\left[\frac{Y}{\ell}\right]\right| \geq \mathbb{E}\left[\frac{Y}{\ell}\right]\right] \leq \frac{1}{\ell\left(\frac{1}{2}\right)^2} = \frac{4}{\ell} = \frac{4}{2^{-(k+2)}} = 2^{-k}.$$

Note that the only place where randomness is used is Line 1 in Algorithm 5, which uses $O(k+r)$ random bits by assumption. The running time of Algorithm 5 is $O(2^k T_{\mathcal{A}}(n))$, where $T_{\mathcal{A}}(n)$ is the running time of $\mathcal{A}$ on an input of length n.

3.5 Interactive Proofs

The model of interactive proofs is illustrated in Figure 1. Let V be a polynomial time verifer, and let P be an "all-power" prover, which has unbounded time and space but is recurisve. Both V and P have read access to the input, and read/write access to conversation tapes. Each of V and P has a private workspace. Moreover, V has random bits. We can show that P does not need random coins (i.e., anything it can do with coins can be done without coins).

Definition 12 (Goldwasser, Micali, Rackoff)**.** An *interactive proof system (IPS)* for a language L is a protocol such that

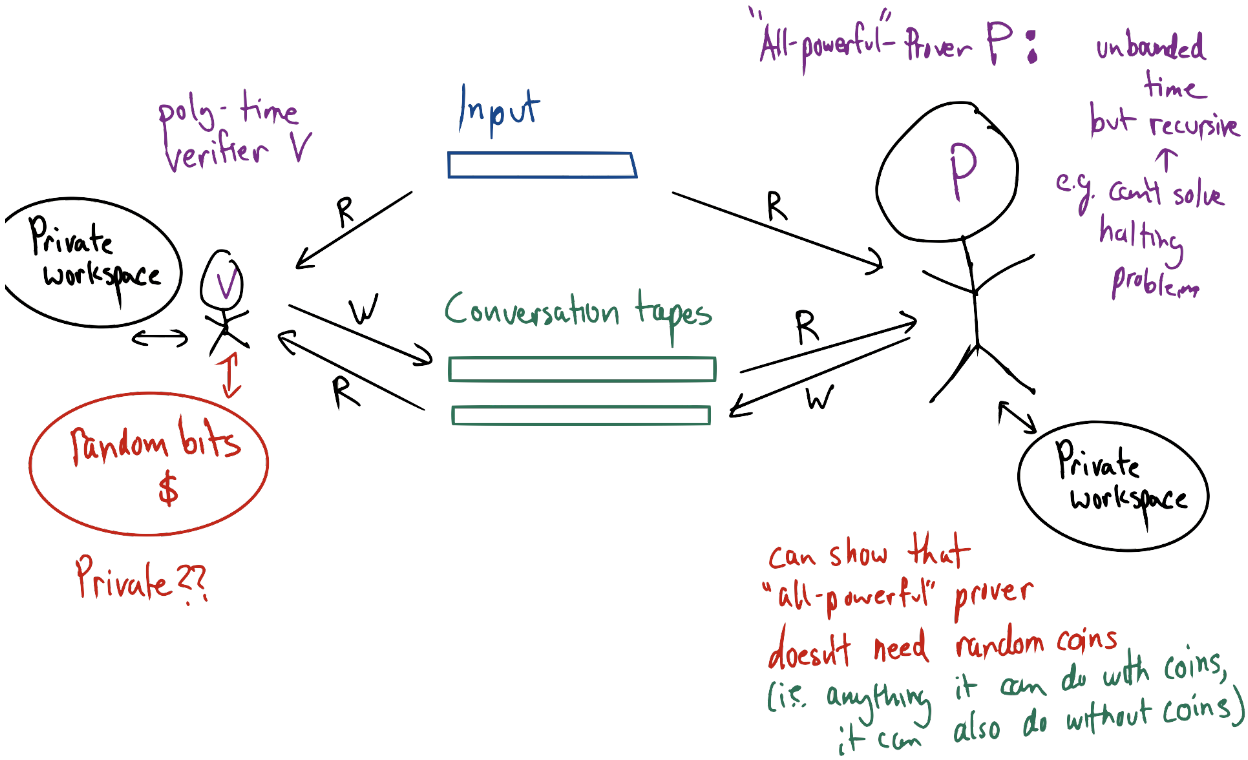

Figure 1: The model of interactive proofs.

- if $x \in L$ and both V and P follow the protocol, then $\mathbb{P}_{\text{coins of } V}[V \text{ accepts } x] \geq 2/3$;
- if $x \notin L$ and V follows the protocol, then $\mathbb{P}_{\text{coins of } V}[V \text{ rejects } x] \geq 2/3$.

Definition 13. $\mathrm{IP} = \{L \subset \{0,1\}^* : L \text{ has an IPS}\}$.

By definition, $\mathrm{NP} \subseteq \mathrm{IP}$.

Theorem 14. $\mathrm{IP} = \mathrm{PSPACE}$.

Problem 15 (graph isomorphism, GI)**.** *Given graphs G and H, are they isomorphic?*

Problem 16 (co-graph isomorphism, $\overline{\mathrm{GI}}$)**.** *Given graphs G and H, are they not isomorphic?*

We know $\mathrm{GI} \in \mathrm{NP}$. It is unknown that $\overline{\mathrm{GI}} \in \mathrm{NP}$. However, $\overline{\mathrm{GI}} \in \mathrm{IP}$.

Theorem 17. *$\overline{GI} \in \mathrm{IP}$.*

Proof. We give a protocol for $\overline{\mathrm{GI}}$, where G_1 and G_2 are graphs:

(i) V picks $c \in \{1, 2\}$.

(ii) V picks a random labeling of vertices in G_c and send the new adjacency matrix to P.

(iii) P guesses c.

(iv) Repeat the above process for k times.

If $G_1 \ncong G_2$, then P can guess correctly every time. If $G_1 \cong G_2$, then P needs to guess coin flips correctly each time, and P can do this with probability at most $1/2^k$. $\square$

Do V's coins need to be private? In the example of $\overline{\mathrm{GI}}$, it *seems* that if P saw V's choices, then it could cheat. However, Goldwasser and Sipser gives a surprising answer: anything that has a protocol with private coins also has a (possible different) protocol with public coins.

Theorem 18 (Goldwasser, Sipser)**.** $\mathrm{IP}_{\textit{private coins}} = \mathrm{IP}_{\textit{public coins}}$.