6.842 Randomness and Computation February 22, 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 $2^{r(n)} \cdot t(n)$.

```
1 run A on every possible random string of length r(n)
2 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 & Maximum Cut

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.

```
1 flip coins r_1, ..., r_n ∈ {0,1} (n = |V|)
2 put vertex i on side r_i (i.e., if r_i = 0 put in S, else in T) for each i ∈ [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 ∈ Z_q
foreach i ∈ {0, ..., q − 1} do
    r_i ← a · i + b  mod q
output r_0, ..., 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 $c, d \in \mathbb{Z}_q$,

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

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.