Derandomization

1 Derandomizing Polynomial Circuits

1.1 Adleman's Proof

We continue the proof of Adleman's Theorem. Recall we were studying the matrix $\mathbf{A} \in \{0,1\}^{2n \times 2m}$ which lists out the function value $f(\boldsymbol{x}, \boldsymbol{r})$ according to the input bitstring $\boldsymbol{x}$ and randomness bitstring $\boldsymbol{r}$. Also recall that by the definition of $\boldsymbol{RP}$, the matrix entry is correct whenever $f(\boldsymbol{x}) = 0$ and is correct when $f(\boldsymbol{x}) = 1$ with probability at least $1/2$.

Consider the submatrix of $\mathbf{A}$ where $f(\boldsymbol{x}) = 1$. Then each row (representing input strings) has at least half 1s, so the whole submatrix is at least half 1s. Call this property $(*)$.

Our goal is to select a polynomial-sized set of columns (representing randomness strings) such that, in each row, we have at least one 1 in a selected column. Then by taking a big OR over all our chosen columns, we would obtain a correct circuit for any input string.

Note that this is a set cover problem. A good column choosing heuristic is to select columns with many 1s – in particular, many "useful" 1s that are not covered by other selected columns. For our first column, we select column $\boldsymbol{r}$ that has the largest number of 1s. This takes care of at least half of all rows by $(*)$. Then, we remove the covered rows and the selection column from consideration and repeat. From all remaining rows, a 0 was removed, so they are still at least half 1s; thus $(*)$ still holds.

After at most $\log_2(2^n) = n$ rounds, all rows are covered. So we have a polynomial-sized derandomized circuit. Note that this is an existential proof; it would take exponential time to construct the matrix $\mathbf{A}$ and a derandomized circuit. Also, this does not prove $\boldsymbol{RP} \subset \boldsymbol{P}$ because we are running a completely different algorithm for every n – we would need an infinite case distinction over all possible values of n. A "solution" is if each input of size n comes with a polynomial-sized "advice sheet" (which depends on the input size, but not the actual input) for the algorithm to use. In our case the "advice sheet" could be a description of the deterministic circuit. These complexity classes are called $\boldsymbol{RP/poly}$ and $\boldsymbol{P/poly}$.

Theorem 1. *[Adleman]. This derandomization theorem implies $\boldsymbol{RP/poly} \subseteq \boldsymbol{P/poly}$. It is a corollary that $\boldsymbol{RP} \subseteq \boldsymbol{RP/poly}$, but it is still open if $\boldsymbol{P} \subseteq \boldsymbol{P/poly}$.*

1.2 Probabilistic Method Proof

Choose $k = n + 1$ columns i.i.d. uniformly. Then,

$$\Pr[\text{row } i \text{ not covered}] \leq 2^{-k}$$

By Union Bound,

$$\begin{aligned} \Pr[\text{exists an uncovered row}] &\leq 2^n \cdot 2^{-k} \\ &= \frac{1}{2}. \end{aligned}$$

By Probabilitic Method, there exists a set of $n + 1$ columns that cover all rows (because there is a nonzero probability of finding it).

2 Method of Conditional Expectations

2.1 Framework

This method can often be applied for "practical" algorithms that minimize/maximize some quantity C (*e.g.,* an approximation guarantee). Suppose that the algorithm $\mathscr{A}$ ran up to some point, having flipped $i-1$ random bits, and is about to flip random bit i out of m. We compare the possible future outcomes (or expectations thereof) for each of the outcomes of this coin flip. The conditional expectation of C, conditioned on the past bits, over the future bits is

$$\begin{aligned}\mathbb{E}_{r_i,\ldots,r_m}[C \mid r_1,\ldots,r_{i-1}] &= \Pr[r_i=1]\cdot\mathbb{E}_{r_{i+1},\ldots,r_m}[C \mid r_1,\ldots,r_{i-1};r_i=1] \ (*)\\ &+ \Pr[r_i=0]\cdot\mathbb{E}_{r_{i+1},\ldots,r_m}[C \mid r_1,\ldots,r_{i-1};r_i=0] \ (\dagger)\\ &\leq \max(*,\dagger)\\ &\geq \min(*,\dagger).\end{aligned}$$

So if we want to minimize/maximize C, we pick the coin flip that gives us the smaller/larger conditional expectation. We can do this starting from bit $i=1$ iteratively. Note that in order to apply this technique, we must be able to evaluate the conditional expectations. This fully derandomizes the algorithm while preserving the guarantees.

2.2 Application to Maximum Cut

Given an undirected graph $G=(V,E)$, we'd like to partition $V=S\cup\bar{S}$ to (approximately) maximize the number of edges between S and $\bar{S}$. The best approximation algorithm for this problem is the Goemans-Williamson 0.878-approximation, but here we will choose a 1/2 approximation and derandomize it.

Johnson's Algorithm: for each node v independently, put it in S with probability 1/2 and $\bar{S}$ otherwise.

Our goal is to maximize $\mathbb{E}$[cut edges]. Let X be the random variable representing the number of edges cut and X_e be an indicator random variable representing whether $e\in E$ crosses the cut. Thus

$$\begin{aligned}X &= \sum_e X_e,\\ \mathbb{E}[X] &= \sum_e \mathbb{E}[X_e]\\ &= \sum_e \Pr[e \text{ is cut}]\\ &= \sum_e \frac{1}{2}\\ &= \frac{m}{2}\end{aligned}$$

Since $OPT\leq m$, we have a 1/2-approximation. Let's now derandomize using conditional expectations. The i^{th} bit of the algorithm determines if $v_i\in S$ or $v_i\in\bar{S}$. Consider iteration i and vertex v_i. Let $V_{i-i}=\{v_1,\ldots,v_{i-1}\}$ and $S_i=\{v_j : j\leq i \text{ placed in } S\}$. Also, $e(S_1,S_2)$ is the number of edges between S_1 and S_2.

$$\begin{aligned}\mathbb{E}[C \mid S_{i-1}, v_i\in S_i] &= \frac{1}{2}e(\bar{V}_i,V)+e(S_{i-1},V_{i-1}\setminus S_{i-1})+e(v_i,V_{i-1}\setminus S_{i-1})\\ \mathbb{E}[C \mid S_{i-1}, v_i\notin S_i] &= \frac{1}{2}e(\bar{V}_i,V)+e(S_{i-1},V_{i-1}\setminus S_{i-1})+e(v_i,S_{i-1})\end{aligned}$$

To apply the method of conditional expectation, we want to choose the larger of $\mathbb{E}[C \mid S_{i-1}, v_i\in S_i]$ and $\mathbb{E}[C \mid S_{i-1}, v_i\notin S_i]$. Canceling common terms, we only need to compare $e(v_i,V_{i-1}\setminus S_{i-1})$ and $e(v_i,S_{i-1})$.

Greedy algorithm: Place the first node arbitrarily, then always place the next node to maximize the number of cut edges to previously processed nodes.

3 Tail Bounds

So far, we studied expectation and basic probability calculations. With tail bounds, we'd like to show that a random variable is close to its expectation with "high enough" probability. Some key applications of this idea include:

1. Show that an algorithm works "well" (*e.g.,* runtime, approximation guarantee) not just in expectation but with high probability
2. Further "process" random variables (*e.g.,* analyze a minimum/maximum of random variables). The typical approach is to show that each random variable is small/large with high probability, then take a union bound to show that the minimum/maximum cannot be too small/large.

Theorem 2. *[Markov's Inequality]. If X is a non-negative random variable, then*

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$

Proof. In the worst case, X takes on values 0 and t only. Then $\mathbb{E}[X] \geq \Pr[X \geq t] \cdot t$. □

A frequentuse case is when $\mathbb{E}[X_t] \leq c^{-t}$ (exponentially small in the number of rounds of some algorithm). Then $\Pr[X_t \geq 1] \leq c^{-t}$.