CMPUT 675: Computational Complexity Theory **Winter 2019**

Lecture 9 (Febuary 5, 2019): Circuits and Randomized Complexity

Lecturer: Zachary Friggstad *Scribe: Joseph Meleshko*

9.1 Circuits

9.1.1 Meyer's Theorem

Theorem 1 (Meyer's Theorem [KL80]) $\mathbf{EXP} \subseteq \mathbf{P}_{\text{poly}} \implies \mathbf{EXP} = \Sigma_2^{\mathbf{P}}$

Proof. Let $L \in \mathbf{EXP}$ be decided by Turning machine M in time 2^{n^c}. On an input x, let $z_0^x, z_1^x, ..., z_{2^{n^c}}^x$ be binary encodings of the snapshots of the computation $M(x)$ where z_0^x is the initial snapshot and $z_{2^{n^c}}^x$ is a snapshot in a halting state. Consider $L' = \{(x, i, j) \mid \text{bit } j \text{ of } z_i^x \text{ is } 1\}$. $L' \in \mathbf{EXP}$ as we can just simulate M on x for i steps and determine the j'th bit of this snapshot, so by assumption $\exists$ poly-size circuit family $\{C_n\}$ for L'. Let $L'' = \{(C, x) \mid \forall i, j,\ C(x, i, j) = \text{bit } j \text{ of } z_i^x\}$.

$L'' \in$ **co-NP**: Let V be a Turing machine that will "locally" check the statement $C(x, i, j) =$ bit j of z_i^x in the following way. That is, V will take (C, x, i, j) as input. If $i = 0$, it checks if the j'th bit of z_0^x is $C(x, i, j)$ by evaluating $C(x, i, j)$ in polynomial time and also by directly computing the j'th bit of z_0^x.

For $i > 0$, V computes $C(x, i, j)$ by evaluating the circuit and also determines if j'th bit of z_i^x by querying $C(x, i-1, j')$ for a polynomial number of bit positions j', in the same way that the proof of the Cook/Levin theorem produced implications that showed how to determine the value of each bit of a configuration by examining a polynomial (indeed, constant) number of bits of the previous configuration. Then a **co-NP** witness is the minimum i and some associated j such that $C(x, i, j)$ is not equal to the j'th bit of z_i^x that verifier V computes by interacting with C at index $i - 1$ as described above. One can think of this as being locally verifiable. We only need to look at part of the last configuration to verify a bit of the current one.

Therefore $x \in L$ if and only if $\exists C (C, x) \in L''$ where $L'' \in$ **co-NP**. In conclusion, our **EXP** language is reduced to a language $\Sigma_2^{\mathbf{P}}$, thus $\mathbf{EXP} \subseteq \Sigma_2^{\mathbf{P}}$. Equality then holds as it is simple to see $\Sigma_2^{\mathbf{P}} \subseteq \mathbf{EXP}$. ■

A neat consequence is that a circuit "upper bound" implies a certain lower bound. Alternatively, if $\mathbf{P} = \mathbf{NP}$ then we also have circuit lower bounds for **EXP**.

Corollary 1 $\mathbf{EXP} \subseteq \mathbf{P}_{\text{poly}} \implies \mathbf{P} \neq \mathbf{NP}$

Proof. If $\mathbf{P} = \mathbf{NP}$ then $\mathbf{P} = \Sigma_2^{\mathbf{P}}$ but by assumption and above theorem, $\mathbf{EXP} = \Sigma_2^{\mathbf{P}}$ and therefore $\mathbf{P} = \mathbf{EXP}$. This is a contradiction with the time hierarchy theorem. ■

9.1.2 Circuit Lower Bounds

Claim 1 $\exists f : \{0,1\}^n \to \{0,1\}$ *requiring circuits of size* $> \frac{2^n}{10n}$

Proof. $\#(f : \{0,1\}^n \to \{0,1\}) = 2^{2^n}$ (Possible functions over n bits)
Our circuits are over 4 node types:

- Variable nodes: outputs the value of the variable $(x_1, x_2, ...)$
- Constant nodes: outputs a constant value (1, 0)
- Binary operators: takes two inputs and outputs a value based on inputs ($\vee$, $\wedge$)
- Unary operators: takes one input and outputs a value based on input ($\neg$)

There are a total of 5 non-variable labels and each depends on at most two previous nodes, each of which can be described by giving their index. Therefore, we can describe each node in a circuit of size s with $\leq 9 \cdot \log_2(s)$ bits. If we fix the convention that the variable nodes are the first n nodes, then we really only need to specify the non-variable nodes.

$$(\#\text{ of circuts with size } \leq s) \leq 2^{9 \cdot \log_2(s) \cdot s}$$

$$(\#\text{ of circuts with size } \leq \frac{2^n}{10n}) \leq 2^{9 \cdot \log_2(\frac{2^n}{10n}) \cdot \frac{2^n}{10n}} \leq 2^{\frac{9}{10} \cdot \frac{2^n}{n} \cdot \log_2(2^n)} = 2^{\frac{9}{10} \cdot 2^n} < 2^{2^n}$$

Comparing the values, we don't have enough circuits to describe every possible function, so there must be a function that requires circuits larger than $\frac{2^n}{10n}$. ■

9.2 Randomized Algorithms

There are several algorithms that use randomization for greater efficiency or effectiveness. Examples include:

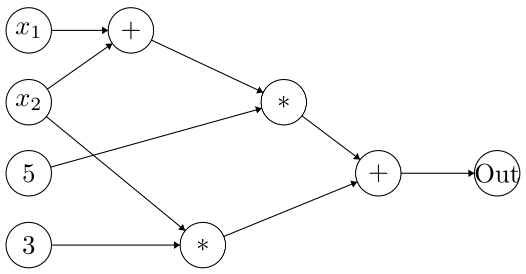
- s-t connectivity (undirected graphs): A random walk starting at s of polynomial length will probably visit t if s and t lie in the same component.
- Min-Cut: Pick an edge uniformly at random, contract it while discarding loops but keeping parallel edges. Stop when you reach a graph with 2 nodes. One can show that if this is repeated a polynomial number of times then with high probability one of the final 2-node graphs represents a minimum cut of the graph.
- 2SAT: While there is an unsatisfied clause, pick a random variable from that clause and flip its bit. If the 2SAT instance is satisfiable, then after repeating this step polynomially-many times you are likely to see a satisfying assignment.
- Median: To find the k'th largest value in an unsorted array, pick a random pivot, divide the array into the two halves around this pivot, and recurse on the side with the k'th largest element (adjusting k if necessary). The expected running time can be shown to be linear.
- Primality testing: There are a couple of fairly simple randomized algorithms that test if a number n is prime in $O(\log^c n)$-time for some constant c. These are even very practical algorithms. In contrast, the known deterministic algorithm is not very practical.

Before discussion randomized complexity, we highlight some key examples.

9.2.1 Polynomial Identity Testing

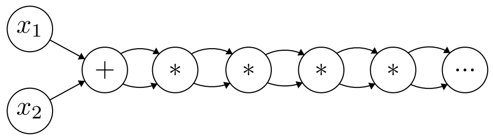
Given an algebraic circuit, does it describe the zero polynomial?

Example of an algebraic circuit for the following expression: $(5 \cdot (x_1 + x_2)) + (-3 \cdot x_2)$:



One could always expand the expression describe by a circuit by totally distributing all multiplication and then cancel similar terms. But the number of terms in the fully expanded form could be exponential in the size of the circuit, so this is not efficient. Example:

$(x_1 + x_2)^{2^n}$ (n is number of $*$ nodes):



Precisely, we consider this problem.

Input: Algebraic circuit over $\mathbb{Q}$.

Output: Accept if the polynomial $p(x_1, x_2, ..., x_n)$ given by the circuit has $p(z_1, z_2, ..., z_n) \equiv 0$. i.e. p is the zero polynomial.

This has a very simple randomized algorithm that is driven by the following lemma.

Lemma 1 (Schwartz-Zippel) *Let S be a finite subset of $\mathbb{Q}$. For any nonzero polynomial $p(x_1, x_2, ..., x_n)$ of degree $d \geq 1$,*

$$\Pr_{z_1, z_2, ..., z_n \sim S} [p(z_1, z_2, ..., z_n) = 0)] \leq \frac{d}{|S|}$$

Where $z_1, z_2, ..., z_n \sim S$ denotes a random, uniform, and independent sampling from S.

Proof. By induction on n, (0_p denotes the zero polynomial)
$n = 1$, any degree d polynomial with 1 variable has at most d roots.
$n \geq 2$, write

$$p(x_1, x_2, ..., x_n) = \sum_{i=0}^{k} x_n^i \cdot q_i(x_1, x_2, ..., x_{n-1})$$

where k is the maximum degree of x_n. (q_k has degree $d - k$.) So by induction,

$$\Pr_{z_1, z_2, ..., z_n \sim S} [q_k(z_1, z_2, ..., z_{n-1}) = 0] \leq \frac{d-k}{|s|}.$$

Now $\forall z_1, z_2, ..., z_{n-1}$ such that $q_k(z_1, z_2, ..., z_{n-1}) \neq 0$,

$$\Pr_{z_n}[p(z_1, z_2, ..., z_{n-1}, z_n) = 0 \mid \text{ given } z_1, z_2, ..., z_{n-1}] \leq \frac{k}{|S|}$$

because any degree k-polynomial has at most k roots. Therefore:

$$\begin{aligned}\Pr_{z_1,z_2,...,z_n\sim S}[p(z_1,z_2,...,z_n)=0)] &\leq \Pr[q_k=0_p]+\Pr[p=0_p \mid q_k\neq 0_p]\cdot\Pr[q_n\neq 0_p] \\ &\leq \frac{d-k}{|S|}+\frac{k}{|S|}=\frac{d}{|S|}\end{aligned}$$

■

This yields a simple randomized algorithm: pick random $z_1,\ldots,z_n\in\{1,2,3,\ldots,2^{d+1}\}$ where n is the number of variables and d is the depth of the circuit. One can show the degree of the underlying polynomial is at most 2^d. The Schwartz-Zippel lemma shows that the probability a nonzero polynomial vanishes is at most $1/2$ whereas, clearly, a zero polynomial will vanish on any input.

Of course, one has to worry about the complexity of evaluating all arithmetic steps and this turns out to be an issue (see the $(x_1+x_2)^{2^n}$ example above), but it can be circumvented with further tricks by reducing all calculations modulo a random large prime p with polynomial bit complexity. This concern will not be an issue with the next example.

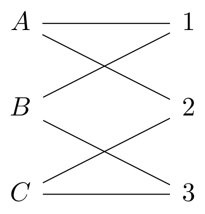
Further Comments: Polynomial identity testing is, perhaps, the flagship problem for the question of whether efficient randomized algorithms can be derandomized.

The question of polynomial identity testing is more subtle over finite fields: for example the polynomial x^2+x vanishes at each point in $\mathbb{Z}/2\mathbb{Z}$ but itself is not the zero polynomial so plugging in random values from $\mathbb{Z}/2\mathbb{Z}$ will not work. Observe the Schwartz-Zippel lemma can be stated and proven (with the same proof) for any field, not just $\mathbb{Q}$. To make it effective for circuits over finite fields, we sample random points from a large enough extension field and do all calculations in this extension field. There are relatively simple algorithms for efficiently constructing and working over extension fields of finite fields like $\mathbb{Z}/p\mathbb{Z}$ for primes p.

9.2.2 Randomized algorithms for bipartite graph problems

9.2.2.1 Bipartite Matching

Example bipartite graph and incidence matrix:



Bipartite incidence matrix for above:

$$M=\begin{array}{c c}
 & \begin{array}{ccc}1 & 2 & 3\end{array} \\
\begin{array}{c}A\\B\\C\end{array} & \begin{pmatrix}x_1 & x_2 & 0\\ x_3 & 0 & x_4\\ 0 & x_5 & x_6\end{pmatrix}
\end{array}$$

Input: Bipartite Graph $G=(V,E)$ with both sides having the same size n.

Output: "Accept" if there exists a perfect matching

Let M be the bipartite incidence matrix where $M_{u,v} = 0$ if $uv \notin E$ and $M_{u,v} = x_e$ if $e = uv \in E$. We first recall one definition of determinants.

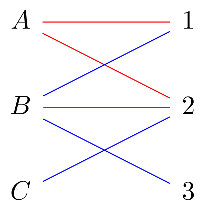
$$\det(M) = \sum_{\sigma:\{1,2,\dots,n\}\to\{1,2,\dots,n\}} \text{sign}(\sigma) \cdot \prod_{i=1}^{n} M_{i,\sigma(i)}$$

Notice this is a polynomial in the entries of M. As a polynomial, $\det(M) \equiv 0$ if and only if there is no perfect matching since $\text{sign}(\sigma) \cdot \prod_{i=1}^{n} M_{i,\sigma(i)} \neq 0$ if and only if σ is a perfect matching in G (meaning $i, \sigma(i)$ is an edge for each i) and two polynomial terms arising from different perfect matchings σ cannot cancel because they depend on different sets of variables. Also, note $\det(M)$ has degree n.

So a simple randomized algorithm to detect if G has a perfect matching would be to sample each x_e independently from $\{1, \dots, 2n\}$. By the Schwartz-Zippel lemma, the determinant of the resulting matrix $\overline{M}$ does not vanish with probability at least $1/2$ if G contains a perfect matching (i.e. $\det(M) \not\equiv 0$). There are well-known algorithms that evaluate the determinant of a matrix over $\mathbb{Q}$ in polynomial time in the total size of the matrix (including the size of the entries), so this is a randomized, polynomial-time algorithm to detect if a bipartite graph has a perfect matching.

9.2.2.2 Red/Blue Matching

Example bipartite graph with colouring and modified incidence matrix:



$$M = \begin{array}{c} \\ \text{A} \\ \text{B} \\ \text{C} \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} y \cdot x_1 & y \cdot x_2 & 0 \\ x_3 & y \cdot x_4 & x_5 \\ 0 & x_6 & 0 \end{pmatrix} \end{array}$$

Input: Bipartite Graph $G = (V, E)$ with edge colourings, (blue or red) non-negative integer k

Output: "Accept" if there exists a perfect matching with exactly k red edges.

In this case, the bipartite adjacency matrix M is formed by $M_{u,v} = 0$ if $uv \notin E$, $M_{u,v} = x_e$ if $e = uv \in E$ is blue, and $M_{u,v} = y \cdot x_e$ if $e = uv \in E$ is red. Here, y is a variable that is used for all red edges.

Write the determinant of M as:

$$\det(M) = \sum_{i=0}^{n} y^i \cdot f_i(x_1, x_2, ..., x_{|E|})$$

Then $f_i(x_1, x_2, ..., x_{|E|}) = 0$ if and only if there is no perfect matching with exactly i red edges.

Sketch of algorithm:

1. Sample $x_1, x_2, ..., x_{|E|} \sim \{1, 2, ..., 2n\}$ uniformly at random.
2. Let M' be the resulting matrix with only y as a variable. Note $\det(M')$ has degree at most n.
3. Interpolate $\det(M')$ by trying all $y \in \{0, 1, ..., n\}$ and evaluating the corresponding determinant.
4. Return "Accept" if coefficient of y^k in $\det(M') \neq 0$.

Note that each f_i polynomial has degree at most n, so by the Schwartz-Zippel lemma we have $f(x_1, \ldots, x_{|E|}) \neq 0$ with probability at least 1/2 if f_i is not the zero polynomial to begin with. The rest of the algorithm is deterministic, so with probability at least 1/2 we will correctly determine G has a perfect matching with k red edges if it does so, and with probability 1 we will determine G has no perfect matching with k red edges if it does not.

Unlike the case of classic bipartite matching, no deterministic, polynomial-time algorithm is known for Red/Blue Matching.

References

KL80 R. Karp and R. Lipton, Some connections between nonuniform and uniform complexity classes, *Proceedings of the 12th annual ACM symposium on Theory of computing*, 1980, p.302-309