Notes on Circuits and Probabilistic Algorithms

1 Circuits

Definition 1 *A language L is solved by a family of circuits $\{C_1, C_2, \ldots, C_n, \ldots\}$ if for every $n \geq 1$ and for every x such that $|x| = n$,*

$$x \in L \;\Leftrightarrow\; C_n(x) = 1.$$

Definition 2 *For a function $S : \mathbb{N} \to \mathbb{N}$ and a language L we say $L \in \mathbf{SIZE}(S(n))$ if L is solved by a family $\{C_1, C_2, \ldots, C_n, \ldots\}$ of circuits, where C_n has at most $S(n)$ gates.*

Recall the following two results from Handout 5.

Lemma 1 *For every language L, $L \in \mathbf{SIZE}(O(2^n))$.*

Lemma 2 *If $L \in \mathbf{TIME}(t(n))$, then $L \in \mathbf{SIZE}(O(t^2(n)))$.*

We can express such results in terms of complexity classes in the following way.

Corollary 3 $\mathbf{P} \subseteq \mathbf{SIZE}(n^{O(1)})$ *and* $\Sigma^* \subseteq \mathbf{SIZE}(O(2^n))$.

Lemma 4 *If $n \geq 3$ and $S \leq \frac{1}{4n} \cdot 2^n$, then there is a function $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed using a circuit of size S.*

Proof: This is a counting argument. There are 2^{2^n} functions $f : \{0,1\}^n \to \{0,1\}$, and we will show that the number of circuits of size S is smaller than 2^{2^n}.

To bound the number of circuits of size S we create a compact binary encoding of such circuits. Identify gates with numbers $1, \ldots, S$. For each gate, specify where the two inputs are coming from and the type of gate. The total number of bits required to represent the circuit is

$$S \cdot (2 + 2\log(n+S)) \leq S \cdot (2 + 2\log 2S) = S \cdot (4 + 2\log S).$$

So the number of circuits of size S is at most $2^{4S+2S\log S}$. We assumed $S \leq \frac{1}{4n} \cdot 2^n$ and so we have

$$\begin{aligned} 4S + 2S\log S &\leq \frac{1}{n}2^n + \frac{1}{2n} \cdot 2^n \cdot (n - \log n - 2) \\ &\leq \frac{1}{n} \cdot 2^n + \frac{1}{2} \cdot 2^n \\ &< 2^n \end{aligned}$$

and we conclude that the number of circuits of size S is strictly smaller than 2^{2^n}, and so some function cannot be computed by any circuit of size S. □

Corollary 5 *There is some language L such that $L \notin \mathbf{SIZE}\left(\frac{1}{4n} \cdot 2^n\right)$.*

It is widely believed that $\mathbf{NP} \not\subseteq \mathbf{SIZE}(n^{O(1)})$, and proving that this is the case is clearly only more difficult than proving $\mathbf{P} \neq \mathbf{NP}$. As of now, we don't even know how to prove $\mathbf{NP} \not\subseteq \mathbf{SIZE}(O(n))$.

2 Probabilistic Algorithms

Definition 3 *A language L is in* **BPP** *if there is a polynomial $p()$ and a polynomial time algorithm $A(\cdot,\cdot)$ such that for every string x of length L*

- *If $x \in L$ then $\mathbf{Pr}_{r\in\{0,1\}^{p(n)}}[A(x,r)$ accepts$] \geq 2/3$;*
- *If $x \notin L$ then $\mathbf{Pr}_{r\in\{0,1\}^{p(n)}}[A(x,r)$ rejects$] \geq 2/3$.*

In other words, a decision problem is in **BPP** if there is a probabilistic polynomial time algorithm that on every input gives a wrong output with probability at most $1/3$. The choice of the particular constant $1/3$ is quite arbitrary, and the probability of error can be reduced by using the following trick: run the algorithm $A()$ several times, using fresh randomness each time. If a majority of the runs accept then accept, otherwise reject. The idea is that if one run of the algorithm has a probability at most $1/3$ of giving an incorrect answer, then if we run the algorithm k times we expect to see less than $k/3$ errors. With high probability, the number of errors will be less than $k/2$ and so by taking the most frequent answer we solve the problem correctly. To make the last sentence formal, we need the following result from probability theory.

Lemma 6 (Chernoff Bound) *Let $X_1,\ldots,X_k$ be independent random variables that take only values zero or one and such that for each i we have $\mathbf{Pr}[X_i = 1] \leq p$ and let $\epsilon < 1/2$. Then*

$$\mathbf{Pr}[X_1 + \cdots + X_k \geq (p+\epsilon)k] \leq e^{-2\epsilon^2 k}$$

Lemma 7 (Error Reduction) *If $L \in$* **BPP** *then there is a probabilistic polynomial time algorithm A' for L whose error probability is at most $1/2^{n+1}$ for inputs of length n.*

Proof: Let $A(\cdot,\cdot)$ be a probabilistic algorithm for L with error probability at most $1/3$, and let $p(n)$ be the length of the random string used by algorithm $A(\cdot,\cdot)$ when the first input is of length n.

The algorithm $A'()$ is given an input x of length n and then random strings $r_1,\ldots,r_k$ where $k = 13n$ and each r_i is a string of length $p(n)$.

We compute $A(x,r_1)$, $\ldots$, $A(x,r_k)$, and if at least $k/2$ of the computations accept then A' accepts, otherwise it rejects.

This means that $A'(x,r_1,\ldots,r_k)$ is wrong only if more than $k/2$ of the computations $A(x,r_i)$ are wrong. Let us define the random variables $X_1,\ldots,X_k$ so that $X_i = 1$ if $A(x,r_i)$ is wrong, and $X_i = 0$ otherwise. The probability that $A'(x,r_1,\ldots,r_k)$ is wrong can be computed using the Chernoff bound with $p = 1/3$ and $\epsilon = 1/6$.

$$\mathbf{Pr}\left[\sum_i X_i > \frac{k}{2}\right] \leq e^{-2\cdot\left(\frac{1}{6}\right)^2\cdot k} = e^{-k/18} < 2^{-n-1}$$

if n is large enough. $\Box$

Note that, more generally, if we have an algorithm whose error probability is $1/2 - \epsilon(n)$ and we do the above error-reduction procedure with $k(n) = \frac{1}{2}\cdot\frac{1}{(\epsilon(n))^2}\cdot\ln\frac{1}{\delta(n)}$ then we get an algorithm whose error probability is at most $\delta(n)$. The new algorithm has polynomial running time as long as $1/\epsilon$ is at most polynomial and $1/\delta$ is at most exponential.

Theorem 8 (Adleman [Adl78]) **BPP** $\subseteq$ **SIZE**$(n^{O(1)})$.

PROOF: Let L be a problem in **BPP** and A' be an algorithm for L that on every input of length n the probability of error is at most 2^{-n-1}.

From A' we can get a family of polynomial size circuits $C_1, \ldots, C_n, \ldots$ such that for every input x of length n and random string r the output of $C_n(x,r)$ is the same as $A'(x,r)$. Now the idea is to find a string r that works correctly for all inputs x; we show that such a string exists by showing that a random string has such a property with probability greater than zero.

$$\mathbf{Pr}_r[\exists x \in \{0,1\}^n . C_n(x,r) \text{ is wrong }] \le \sum_{x\in\{0,1\}^n} \mathbf{Pr}_r[C_n(x,r) \text{ is wrong }] \le \frac{1}{2}$$

so that

$$\mathbf{Pr}_r[\forall x \in \{0,1\}^n . C_n(x,r) \text{ is right }] \ge \frac{1}{2}$$

Let r_{good} be a string r such that $C_n(x,r)$ is right for every x of length n, and define the circuit $C'_n(x) = C_n(x, r_{\text{good}})$.

This process defines a family of polynomial size circuits for L. $\Box$

It is now strongly believed that $\mathbf{P} = \mathbf{BPP}$. The main reason for such belief is the following result [NW94, IW97].

Theorem 9 (Nisan-Impagliazzo-Wigderson) *Suppose there is a constant $\epsilon > 0$ and a language $L \in \mathbf{TIME}(2^{O(n)})$ such that for every large enough n there is no circuit of size $\le 2^{\epsilon n}$ that solves L on inputs of length n. Then $\mathbf{P} = \mathbf{BPP}$.*

Even though the premise of the theorem is strongly believed to be true, we do not even know how to prove that $\mathbf{TIME}(2^{O(n)}) \not\subseteq \mathbf{SIZE}(O(n))$.

References

[Adl78] Leonard Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[IW97] R. Impagliazzo and A. Wigderson. $P = BPP$ unless E has sub-exponential circuits. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994. Preliminary version in *Proc. of FOCS'88*.