

Unless stated otherwise, let $\Sigma = \{0, 1\}$ be the binary alphabet.

1 Circuits

Example 1.1 (Hardwiring). Let $L \subseteq \Sigma^*$ be such that $L \subseteq 0^*$ and L is undecidable. Then $L \in P/poly$ because, on input $x \in \Sigma^n$, we hardwire the value 0^n into a circuit that outputs 1 if $x = 0^n$ and 0 otherwise.

Example 1.2 (Hardwiring). Let $L \subseteq \Sigma^*$ be such that, for all $n \in \mathbb{N}$, $|L \cap \Sigma^n| \leq n^2$. Then $L \in P/poly$ because we can hardwire all strings in $L \cap \Sigma^n$ into the circuit using only about $O(n \cdot n^2) = O(n^3)$ gates.

So far, we have discussed four notions of “efficient computation”:

- Deterministic Uniform: P
- Probabilistic Uniform: BPP
- Deterministic Nonuniform: P/poly
- Probabilistic Nonuniform: BPSIZE(poly)

The last is defined by *probabilistic circuits*, each of which can be thought of as a circuit C taking two strings of input bits, the *input* $x \in \Sigma^*$, and the *random bits* $r \in \Sigma^*$. If $L \in BPSIZE(poly)$, then there exists a family of probabilistic circuits $C = [C_0, C_1, \dots]$ and polynomials p, s such that, for all $n \in \mathbb{N}$, $|C_n| \leq s(n)$, and, for all $x \in \Sigma^n$,

$$x \in L \implies \Pr_{r \in \Sigma^{p(n)}} [C_n(x, r) = 1] \geq 2/3,$$

and

$$x \notin L \implies \Pr_{r \in \Sigma^{p(n)}} [C_n(x, r) = 1] \leq 1/3.$$

In this case, we say C *decides* (*accepts*, *computes*) L .

Theorem 1.3. *Let $L \subseteq \Sigma^*$ be accepted by a program that runs in time $t(n)$. Then there is a circuit family $C = [C_0, C_1, \dots]$ that decides L and, for all $n \in \mathbb{N}$, $|C_n| \leq t^2(n)$. (Actually, this bound can be improved to $O(t(n) \log t(n))$.)*

In other words, any deterministic computation can be converted into a circuit. This implies the following.

Proposition 1.4. $P \not\subseteq P/poly$.

The subset can be seen to be proper by Example 1.1.

We also have the following proposition, which follows from the fact that a BPP machine that ignores its random bits is simply a P machine.

Proposition 1.5. $P \subseteq BPP$.

Whether this subset is proper remains open.

Question 1.6. $P = BPP$?

The following theorem states that nonuniform polynomial-size circuits can simulate uniform randomness.

Theorem 1.7. $BPP \subseteq P/poly$.

Proof. Let $L \in BPP$. It suffices to show that there exists a polynomial-size deterministic circuit family $C = [C_0, C_1, \dots]$ that accepts L . Since $L \in BPP$, there is a polynomial p and a deterministic polynomial-time machine M such that, for all $x \in \Sigma^n$,

$$x \in L \implies \Pr_{r \in \Sigma^{p(n)}} [M(x, r) \text{ accepts}] \geq 1 - \frac{1}{2^{2n}},$$

$$x \notin L \implies \Pr_{r \in \Sigma^{p(n)}} [M(x, r) \text{ accepts}] < \frac{1}{2^{2n}}.$$

Definition. For each $x \in \Sigma^n$, we say $r \in \Sigma^{p(n)}$ is *wrong for x* if $M(x, r) \neq L(x)$.

For all $x \in \Sigma^n$, number of $r \in \Sigma^{p(n)}$ that are wrong for x is at most $2^{p(n)}/2^{2n}$.

Definition. $r \in \Sigma^{p(n)}$ is *bad* if there is an $x \in \Sigma^n$ such that r is wrong for x .

For all $n \in \mathbb{N}$, the number of bad r 's is at most

$$\frac{2^{p(n)}}{2^{2n}} 2^n \leq \frac{2^{p(n)}}{2^n}.$$

by the union bound (take the union over all x of those r 's that are wrong for x) and the previously stated bound on the number of wrong r 's for each x .

Thus

$$\Pr_{r \in \Sigma^{p(n)}} [(\exists x \in \Sigma^n) M(x, r) \neq L(x)] \leq \frac{1}{2^n}.$$

Therefore, there exists an r that is not bad. (In fact, this inequality implies that most r 's in $\Sigma^{p(n)}$ are not bad.)

This r , because it is not bad, works for *all* strings of length n . Therefore, we hardwire this r into the circuit C_n , which, on input $x \in \Sigma^n$, simulates $M(x, r)$. Because r is not bad, $M(x, r)$ outputs the correct answer, and C_n correctly decides x . $|C_n|$ is bounded by a polynomial in n because $|r| = p(n)$ and M runs in polynomial time. This shows that $L \in P/poly$. \square

The main idea behind this proof is, "For every x , almost all r 's are correct for x , so there is one r correct for *every* x ."

We can use the same idea to convert any probabilistic *circuit* (rather than probabilistic algorithm, as in Theorem 1.7) into a deterministic circuit. Therefore, *randomness does not help in nonuniform computation*, whence the following equality holds.

Theorem 1.8. $BPSIZE(poly) = P/poly$.

2 Max-Cut

Given a graph $G = (V, E)$, with $A, B \subseteq V$, $A \cap B = \emptyset$, $A \cup B = V$ (i.e., (A, B) is a partition of V), define $\text{Cut}(A, B) =$ the number of edges from A to B . Max-Cut is the problem of partitioning V into A and B such that the size of $\text{Cut}(A, B)$ is maximized.

Max-Cut is NP-hard; i.e., a polynomial-time algorithm for Max-Cut implies $P = NP$. What about approximation algorithms?

Max-Cut $_c$ is the problem, given a graph G find a cut whose size is at least c times the value of the maximum cut in G , where $c < 1$. There is a randomized algorithm that solves Max-Cut $_{1/2}$:

```
MAX-CUT $_{1/2}(G = (V, E))$ 
1   $n \leftarrow |V|$ 
2   $m \leftarrow |E|$ .
3   $A \leftarrow \emptyset$ 
4   $B \leftarrow \emptyset$ 
5  for  $i \leftarrow 1$  to  $n$ 
6      do randomly pick  $b \in \{0, 1\}$ 
7      if  $b = 0$ 
8          then  $A \leftarrow A \cup \{v_i\}$ 
9          else  $B \leftarrow B \cup \{v_i\}$ 
```

Theorem 2.1. *The above algorithm outputs a cut whose size is at least $m/2$ with probability at least $\frac{1}{m+1}$.*

Proof. For each edge, what is the probability that it belongs to the cut the algorithm selects? It is $1/2$ since the vertices in each edge are on different sides of the cut with probability $1/2$. Therefore, the expected number of edges that belong to the cut is $m/2$.

Define for all $1 \leq i \leq m$

$$X_i = \begin{cases} 1, & \text{if } i^{\text{th}} \text{ edge is in the cut;} \\ 0, & \text{otherwise.} \end{cases}$$

and

$$X = \sum_{i=1}^m X_i.$$

X is the random variable that corresponds to the output of the algorithm; i.e., the size of the cut output. Then $E[X_i] = 1/2$ and $E[X] = m/2$.

Note that for all random variables Y ,

$$\Pr[Y \geq E[Y]] > 0.$$

Thus

$$\Pr[X \geq m/2] > 0.$$

We want $\Pr[X \geq m/2]$ to be large. Because it is not zero, we can repeat the algorithm to amplify it. But we need it bounded away from 0 so it can be amplified in polynomial time; i.e., we need $\Pr[X \geq m/2]$ to be at least $1/\text{poly}(m)$.

The crucial observation is to notice that X cannot take on any value, but only the integer values $0, 1, \dots, m$. The average is $m/2$. Since X is an integer, to be *strictly less* than $m/2$, X must be less than or equal to $m/2 - 1$. We first consider the case m is odd, so $m = 2k + 1$. Thus $E(X) = k + 1/2$. Since X can take only integer values, $\Pr[X \geq E(X)] = \Pr[X \geq k + 1]$, and $\Pr[X < E(X)] = \Pr[X \leq k]$.

Define

$$p = \Pr[X \geq m/2].$$

Then

$$\begin{aligned} m/2 &= E[X] \\ &= \sum_{a=0}^m \Pr[X = a] \cdot a \\ &= \sum_{a < E[X]} \Pr[X = a] \cdot a + \sum_{a \geq E[X]} \Pr[X = a] \cdot a \\ &\leq \sum_{a=0}^k \Pr[X = a] \cdot a + \sum_{a=k+1}^m \Pr[X = a] \cdot a \\ &\leq k \sum_{a=0}^k \Pr[X = a] + (2k + 1) \sum_{a=k+1}^m \Pr[X = a] \\ &\leq k(1 - p) + (2k + 1)p \\ &\leq k - kp + 2kp + p. \end{aligned}$$

Since $m/2 = k + 1/2$,

$$\begin{aligned} k + 1/2 \leq k - kp + 2kp + p &\implies 1/2 \leq p(1 + k) \\ &\implies p \geq \frac{1}{2(1 + k)} \geq \frac{1}{m + 1}. \end{aligned}$$

Therefore, $\Pr[X \geq m/2] \geq \frac{1}{m+1}$, which is bounded far enough away from 0 to amplify in polynomial time. \square

Goemans and Williamson showed that this can be improved to a polynomial-time randomized algorithm for $\text{Max-Cut}_{0.87}$. On the other hand it is known that there exists a constant $c < 1$ such that Max-Cut_c is NP-hard. Best known value of c is $83/84$.

Given a boolean formula in 3-CNF form

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_3} \vee x_1 \vee x_4) \wedge \dots,$$

computing a satisfying assignment to ϕ is NP-hard. Using similar ideas as before, the following can be shown.

Theorem 2.2. *There is a probabilistic polynomial-time algorithm that, given a 3-CNF formula ϕ , computes an assignment that satisfies at least a fraction $7/8$ of the clauses of ϕ .*

This cannot be improved to $7/8 + \epsilon$ for any $\epsilon > 0$ unless $P = NP$ (this is due to Håstad, using the PCP theorem)