Lecture 22: Constant Depth Circuit

instructor: Jin-Yi Cai Scribe: Rakesh Kumar, Yunpeng Li

The main goal of complexity theory is to understand the relationships among resource bounded computations under various models. We can obtain various central complexity classes by combining the different models and resources, and the study of the relationships among these classes forms the core of complexity theory. For instance, comparing time bounded non-deterministic and deterministic computations leads to the NP vs. P question, the most important open question in computer science.

Many researchers believe that circuits might be a easier model than Turing machine and tried hard to prove lower bound for the circuits mode. It is easy to see that any language in P has a corresponding polynomial size circuit. And as lectured before: if NP has polynomial size circuit, the polynomial hierarchy will collapse, which is unlikely. People do hope to find a super-polynomial size lower bound for a NP-complete languages, which leads to the separation of P and NP. However, it seems to be the Waterloo for complexity theory since the best known result till now is only $O(n)$.

To make life relatively easier, we consider a special restricted circuit family, the constant depth circuit, which is successfully proved to be overwhelmed by some complexity classes.

1 Constant Depth Circuit

A circuit with n inputs is a directed acyclic graph with n leaves(called the *input wires*) and each internal nodes is labelled with $\vee, \wedge, \neg$. The $\vee$ and $\wedge$ nodes has fanin(the number of incoming edges) of t and the $\neg$ node has fanin 1. There is a designate output node. Thus the circuit implements a function from $\{0,1\}^n$ to $\{0,1\}$. The size of the circuit is the number of wires in the graph.

A *circuit family* $(W_n)_(n \geq 1)$ is a sequence of circuits where W_n is a circuit with n inputs. $(W_n)_(n \geq 1)$ decides a language L if for every input x

$$x \in L \Leftrightarrow W_{|x|}(x) = 1$$

Without lose of generality, we can view each circuit as a tree by allowing duplications. The height of the tree is defined as the depth of the circuit. *Constant depth circuit* is the set of circuit families with constant depth and polynomial size. In particular, for the d-depth circuit, if the output gate is $\vee$, we call it Σ_d-circuit, and Π_d circuit if $\wedge$ gate.

We consider the binary addition as an example. Given two binary number $a = a_{n-1} \cdots a_1 a_0$ and $b = b_{n-1} \cdots b_1 b_0$, we can use the "school method" to add these two–adding each column from the lease significant bit to the most significant bit, with carry bit c_i along the way. Let $r = a + b$:

$$\begin{aligned} r_k &= a_k \oplus b_k \oplus c_k \\ c_k &= (c_{k-1} \wedge (a_k \vee b_k)) \vee (a_k \wedge b_k) \end{aligned}$$

And we can do it faster in parallel. Note that the computation expressed above is sequential. we have to compute all the previous output bits before computing r_k. We can express bits r_k's in CNF or DNF form. Such a circuit will have dept just 2 but this will give us an exponential size circuit in general.

It turns out that we can do this in constant depth circuit using a method called *Carry-Look-Ahead*. In computing $r_k = a_k \oplus b_k \oplus c_k$, the main problem is computing c_k and what we want to do is to compute c_k in one step for any k. generally, for c_k to be 1, a carry bit has to be generated at some position $i < k$ and should be propagated all the way through position k, formally

$$
\begin{aligned}
c_k &= \exists(0 \le i \le k)\left[a_i \wedge b_i \bigwedge \forall j(i < j < k)(a_j \vee b_j)\right] \\
&= \bigvee_{0 \le i < k}\left[a_i \wedge b_i \wedge \bigwedge_{i<j<k}(a_j \vee b_j)\right]
\end{aligned}
$$

To do the above computation in constant depth, we have to have arbitrary fanin.

We will consider here only depth 2 circuit. We say a boolean function G on variable $\{x_1, \cdots, x_n\}$ is a t-And-Or if $G = G_1 \wedge G_2 \wedge \cdots \wedge G_w$, where each G_i is the OR of at most t literals, (a literal is a variable or its complement) i.e.,

$$G_i = \bigvee_{j \in S} x_{ij} \vee \bigvee_{k \in S'} \overline{x}_{ik} \qquad \text{with } |S \cup S'| \le t|$$

And we denote $G' = \bigwedge_{i>1} G_i$. Similarly, we define the t-Or-And.

2 Decision Tree Complexity

A *decision tree* is a model of computation that used to study the number of bits of an input that need to be examined in order to compute some function on these inputs.

Recall the use of decision tree in the proof of comparison-based sorting. The depth of the decision tree T, $d(T)$, is defined as the longest path from root to leaf. And we define the decision tree complexity as

Definition 2.1 (Decision Tree Complexity). *For boolean function $f : \{0,1\}^n \mapsto \{0,1\}$ the decision tree complexity for f, $DC(f)$ is defined by*

$$DC(f) = \min_T\{d(T) \mid T \text{ computes } f\}$$

We consider the graph connectivity as our example. Given a graph G as input in adjacency matrix, we would like to know how many bits a decision tree algorithm has to read in order to determine the connectivity of the graph. By the adversary argument, it is easy to see that DC(graph connectivity)$=\binom{n}{2}$, where n is the number of vertices of the graph.

3 Random Restriction

Let Σ be a finite set of boolean variables, A restriction on Σ is a partial assignment of Σ to boolean values. Formally, a restriction on Σ is a mapping ρ of Σ ro $\{0, 1, *\}$, where $\rho(x) = 0$(resp. 1) indicates that $x \in \Sigma$ is assigned the value 0(resp. 1) and $\rho(x) = *$ indicates that x is not assigned a value, i.e., is preserved as a variable.

Definition 3.1. *Let Σ be a set of variables and $p, 0 < p < 1$, be a real number. F is a boolean function over Σ.The random restriction is the distribution on the restrictions on Σ, for each variable $x \in \Sigma$,*

$$\rho(x) = \begin{cases} *, & \text{with probability } p; \\ 0, & \text{with probability } (1-p)/2; \\ 1, & \text{with probability } (1-p)/2. \end{cases}$$

We would like to know $DC(G\mid_\rho$, in particular, $\Pr[DC(G\mid_\rho > \Delta]$, the goal is to say that the probability decays exponentially in Δ.

4 Switching Lemma

LEMMA 4.1. *Let G be a t-And-Or formula $G_1 \wedge G_2 \wedge \cdots \wedge G_w$. Let ρ be a random p-restriction. Then for all $\Delta \geq 0$,*

$$\Pr[DC(G\mid_\rho > \Delta] \leq (5pt)^\Delta$$

Proof. The lemma is proved by an induction on w. Concerning G_1, immediately there are 2 cases, either $G_1\mid_{rho} \equiv 1$ or $G_1\mid_{rho} \not\equiv 1$. By renaming literals, we may assume $G_1 = \bigvee_{i\in T} x_i$. Then $G_1\mid_{rho} \equiv 1$ is equivalent to $\rho(i) = 1$ for some $i \in T$. If $G_1\mid_{rho} \equiv 1$, we want to prove that the conditional probability that the rest of G has $DC(G\mid_\rho > \Delta$ is no larger. If however $G_1\mid_{rho} \not\equiv 1$, we want to carefully analyze what happens to the variables in T. All of this will accumulate as some prior condition on ρ. It will be seen that the inductive step will carry a condition that refers to some prior collections of subsets of variables on each of which ρ has assigned some variables if it in some definite way. We will use Håstad's proof, which is more elegant.

First, we make a stronger claim, that for any boolean function F, we have

$$\Pr[DC(G\mid_\rho > \Delta \mid F|_\rho \equiv 1] \leq (5pt)^\Delta$$

The lemma follows from above equation by taking F to be the constant function.

Notice that the claim is true for $\Delta = 0$ since the RHS becomes 1 in this case. Similarly, if $5pt \geq 1$, then the statement is true. Thus, we may assume $\Delta > 0$ and $5pt < 1$.

We will finish the proof next time. □