Recent advances in Complexity CIS 6930/CIS 4930	October 1, 2002
Lecture 11	
Lecturer: Dr. Meera Sitharam	Scribe: Erwin Jansen

1 Introduction

This lecture was split up in two parts. During the first an overview was given how to prove a circuit size lower bound *independent* of depth. The second part was a talk given by Zia on communication complexity and circuit depth.

2 Monotone Circuits

In order to prove a lower bound on circuit size independent of depth we will restrict ourselves to monotone circuits. A monotone circuit by definition computes a monotone boolean function. Before we explain what a monotone function is we will first describe what a dominating vector is:

Dominating Vector

Definition 1 (Dominating Vector). *Given two vectors $\vec{v}$ and $\vec{w}$ we say that $\vec{v}$ dominates $\vec{w}$ written as $\vec{v} \succeq \vec{w}$ if:*

$$\forall_i \langle v_i \geq w_i \rangle$$

Using this definition we can define what a monotone boolean function is. We call a boolean function monotone whenever it gives a one for an input X it gives a one for every input dominating X. Formally:

Monotone Function

Definition 2 (Monotone Function). *Let $X, Y \in \{0,1\}^*$ and $|X| = |Y|$. Now a function $f : \{0,1\}^* \to \{0,1\}$ is called a monotone function if:*

$$\forall_{Y \succeq X} \langle f(X) = 1 \to f(Y) = 1 \rangle$$

There are many non-trivial monotone functions that are interesting for a complexity theorist to investigate:

Example 1 (Monotone Functions). *Here we give two examples of monotone boolean functions, one in P and one in NP. For both examples we assume that the input variables are the possible edges in a graph. A one in the input represents the existence of an edge.*

- *Bipartite matching function: Consider a graph with a bipartite matching. The addition of another edge will leave the matching intact. This problem can be transformed into finding a maximum flow in a graph and hence the problem is in P.*

- *The clique function: Consider a graph with a clique of size k. Now adding an edge to this graph will still contain a clique of size k. This problem is in NP.*

A monotone circuit is a circuit that only consists of *or* and *and* gates. A monotone circuit computes a monotone function, since both $a \wedge b$ and $a \vee b$ are monotone functions. (Notice that only $\neg a$ is a non-monotone operation!). Vice versa any monotone function can be computed by a monotone circuit. This does not imply that we cannot compute the same function with a circuit that is not monotone, i.e. contains an $\neg a$ gate somewhere.

Excercise

Exercise 1. *Show that any monotone function has at least one monotone circuit.*

Now we are going to look at the lower bound of the size of a monotone circuit for a monotone function without any restrictions of depth. So the first problem we will look at is this:

Problem 1 (Bound for monotone circuit). *What is the circuit size lower bound on a monotone circuit for a monotone function without any restrictions on depth.*

If we could show an exponential lower bound and if we can show that non monotone circuits for monotone functions can be converted to monotone circuits with only a polynomial blow up in size then we would have proven that $P \neq NP$!

Obviously there is a catch somewhere. It has been shown that a simulation like this does not exist. In other words, there are monotone functions which have a small non-monotone circuit but require a big monotone circuit. It turns out that for the bipartite matching function that is in P it holds that:

Problem 2 (Matching has huge circuit). *A monotone circuit for bipartite matching is exponential in size.*

Showing that there is a lower bound for arbitrary monotone functions does not really help us in proving whether $P = NP$ or not.

2.1 Slice functions

Despite the huge gap between monotone and general circuit complexity there is a special class of functions for which the two complexities are polynomially related. The class of functions are called *slice* functions. A slice function is a function that gives a specific output when the number of ones is above or below a certain threshold. However if the amount of ones in the input equals the threshold then the function is arbitrary. Formally:

Slice Function

Definition 3 (Slice Function). *A slice function is a function that has the following characteristic:*

$$f(x) = \begin{cases} 0 & \text{if } \sum_{i=0}^{N} x_i < k, \\ \textit{arbitrary} & \text{if } \sum_{i=0}^{N} x_i = k, \\ 1 & \text{if} \sum_{i=0}^{N} x_i > k. \end{cases}$$

It sounds as though the use of slice functions are limited, but it turns out there are some slice functions that are in NP.Circuits for monotone slice functions can be simulated with polynomial blowup by monotone circuits[**?**]. You can prove $P \neq NP$ if you started out with a monotone slice function and proved an exponential monotone circuit size lower bound.

2.2 Proving the problems

Up until now we have given proofs that are called natural. Natural proofs follow a certain structure that has been formalized in [**?**]. If a proof has this structure then it cannot be used to prove $P \neq NP$. Unfortunately this means that we cannot build upon the techniques used so far.

The proof we will present is not going to be a natural proof technique. The reason for this is that we are using a special property of the function, namely its monotonicity Here we give the global outline of what we are going to prove in the upcoming lecture:

Theorem 4. *For $k \leq n^{\frac{1}{4}}$ the monotone circuit size required to compute the $clique_{n,k}$ function is at least $n^{\Omega(\sqrt{k})}$*

Proof-idea

1. Avoid dealing with all monotone circuits, but only those "nice" ones that are likely to compute $clique_{n,k}$.
2. Show that any monotone circuit computing $clique_{n,k}$ can be approximated by another nice monotone *approximator* circuit without a blow up in size.
3. Show that the nice monotone approximator circuit for $clique_{n,k}$ has to be big.

We will examine the performance of our approximator circuits over a subdomain of inputs that capture the essence of what is hard to distinguish between a clique or no clique: the sub-domain of our approximation circuit is going to consist of the positive and the negative test graphs.

Positive test

Definition 5 (Positive test graph). *A positive test graph is a graph that barely has clique's: more precisely it has only one k clique and no other edges.*

Negative test

Definition 6 (Negative test graph). *A negative test graph is a graph that has many $(k-1)$ clique's but no k clique. In fact, it is obtained by coloring a graph of n vertices randomly with $k-1$ colors so that exactly $n/k-1$ vertices has the same color, and putting edges between all pairs of vertices that have different colors.*

The first goal we set for ourselves will be to proof the following claim:

Claim 7. *Let C be a monotone circuit for $clique_{n,k}$ we can find an approximation $\hat{C}$ such that $C \leq \hat{C}$ holds on most positive test graphs ($\hat{C}$) and vice versa on negative test graphs.*

After we have proven this claim we will prove:

Claim 8. *Show that every approximator either outputs 0 on most positive test graphs or outputs 1 on most negative test graphs*

3 Part 2 – Zia's talk on communication complexity technique for proving circuit depth lower bounds

The talk is divided into three parts. In this lecture only the first two parts where discussed:

Part 1: Introduction and definition of the communication complexity of a function. An upper and lower bound for communication complexity will be established.

Part 2: An algebraic way of arriving at a lower bound for communication complexity.

Part 3: Define the relationship between the communication complexity of f and the minimum depth of a circuit computing f

The last part is discussed in the next lecture.

3.1 Introduction

Suppose X, Y, Z are arbitrary finite sets and $f : X \times Y \to Z$ a function. Suppose there are two players: Alice and Bob, who wish to evaluate $f(x, y)$ for some input $x \in X$ and $y \in Y$. We will assume that Alice and Bob both have unlimited computational power, since we are only interested in communication complexity.

Now the problem is that Alice only knows x and Bob only knows y. Hence to evaluate $f(x, y)$ they must communicate with each other. In other words they have to send bits to each other until the value of $f(x, y)$ can be determined. This communication will be carried out according to some fixed protocol $\mathcal{P}$ which is defined as follows:

Protocol

Definition 9 (Protocol). *A protocol $\mathcal{P}$ over domain $X \times Y$ with range Z is a binary tree with the following properties:*

1. *$\mathcal{P}$ depends only on f*
2. *$\mathcal{P}$ must determine at each stage whether the communication may stop, and if so, what the value $f(x, y)$ is*
3. *If the communication is not to stop, then $\mathcal{P}$ must specify who speaks next, i.e. which player sends a bit of communication next, and what the player says. This information must depend solely on the bits communicated so far during the run of the protocols*

Based upon the protocol we can define what the communication complexity is:

Communication Complexity

Definition 10 (Communication Complexity). *For a function $f : X \times Y \to Z$, the (deterministic) communication complexity of f, denoted by $D(f)$, is the minimum cost of $\mathcal{P}$, over all protocols $\mathcal{P}$ that compute f*

Lemma 11. *For every function $f : X \times Y \to Z$, we have $D(f) \leq \log_2 |X| + \log_2 |Z|$*

Proof. Alice could send all her input to Bob. This requires $\log_2 |X|$ bits, using an appropriate encoding. Then Bob can compute $f(x, y)$ using his unlimited computational power. Then Bob can send the answer back to Alice, which requires $\log_2 |Z|$ more bits. □

Lemma 12. *For every function $f : X \times Y \to Z$, we have $D(f) \geq \log_2 |Range(f)|$.*

Proof. Any protocol $\mathcal{P}$ computing f must have at least $|Range(f)|$ leaves. Hence the depth of $\mathcal{P}$ must be at least $\log_2 |Range(f)|$. □

The careful reader must have noticed by now that this lower bound is rather useless for boolean functions $f : X \times Y \to \{0, 1\}$ since by lemma 12 we have have $D(f) \geq 1$. We will now try to raise this lowerbound

3.2 Towards a higher lowerbound

Reachability

Definition 13 (Reachability). *Let $\mathcal{P}$ be a protocol and let v be a node of $\mathcal{P}$. The set R_v is the set of inputs (x, y) that reach node v.*

As one can see $R_{root} = X \times Y$.

Lemma 14. *If L is the set of leaves of a protocol $\mathcal{P}$, then $\{R_l\}_{l \in L}$ is a partition of $X \times Y$.*

Proof. No input (x, y) can reach two different leaves of $\mathcal{P}$ since from any internal node, (x, y) must take the left or the right subtree, not both. Thus the R_l's are mutally disjoint. Evidently $\bigcup_{l \in L} R_l \subseteq X \times Y$ and $X \times Y \subseteq \bigcup_{l \in L} R_l$ since any input will traverse $\mathcal{P}$ to reach some leaf l and therefor will be in R_l. □

Rectangle

Definition 15 (Rectangle). *A (combinatoral) rectangle in $X \times Y$ is a subset $R \subseteq X \times Y$ such that $R = A \times B$ for some $A \subseteq X$ and $B \subseteq Y$*

Lemma 16. *For every protocol $\mathcal{P}$ and node v of $\mathcal{P}$, the set R_v is a rectangle.*

Proof. We will prove by induction on the depth of v that R_v is a rectangle. If v is the root then we have already seen that $R_v = X \times Y$, which is a rectangle. Otherwise, let w be the parent of v. Assume without los of generality that v is the left child of w, and that in w Alice speaks, i.e. w is labeled with a function $a_w : X \to \{0.1\}$. The $R_v = R_w \cap \{(x, y) \mid a_w(x) = 0\}$. By the induction hypothesis, $R_w = A_w \times B_w$ for some $A_w \subseteq X$ and $B_w \subseteq Y$. Thus $R_v = (A_w \times B_w) \cup (\{x\}_{a_w(x)=0}) \times B_w$, which is a rectangle since $A_w \cap \{x\}_{a_w(x)=0} \subseteq X$. □

Monochromatic set

Definition 17 (Monochromatic set). *A subset $R \subseteq X \times Y$, is called f-monochromatic if the function f is fixed on R.*

Lemma 18. *Any protocol $\mathcal{P}$ for a function f induces a partition of $X \times Y$ into f-monochromatic rectangles. The number of these rectangles is the number of leaves of f*

Proof. By lemma 14, $\{R_l\}_{l \in L}$ is a partition of $X \times Y$, where L is the set of leaves of $\mathcal{P}$. By lemma 16, each $\{R_l\}_{l \in L}$ is a rectangle. Each R_l is monochromatic since if $(x, y) \in R_l$ then $f(x, y)$ is the element of Z that labels the leaf l, of course the number of these rectangles $\{R_l\}_{l \in L}$ is clearly L. □

Corollary 19. *If any partion of $X \times Y$ into f-monochromatic rectangles requires at least t rectangles, then $D(f) \geq \log_2 t$*

Proof. By lemma 18, any protocol for f must have at least t leaves. Hence the depth of such a protocol is at least $\log_2 t$ □

This corollary gives a strategy for proving lower bounds on the communication complexity of a function f. We simply prove lower bounds on the number of rectangles in any partition of $X \times Y$ into f-monocromatic rectangles.

Associated Matrix

Definition 20 (Associated Matrix). *For every $f : X \times Y \to Z$, the associated matrix M_f is the matrix of dimensions $|X| \times |Y|$, where the rows of M_f are indexed by the elements of X, the columns by the elements of Y, and the (x, y) entry of M_f is $f(x, y)$. By $rank(f)$ we mean the linear rank of M_f over $\mathbb{R}$, i.e., the number of linear independent rows of M_f over $\mathbb{R}$.*

Lemma 21. *For any function $f : X \times Y \to \{0, 1\}$, we have:*

$$D(f) \geq \log_2 rank(f)$$

.

Proof. Let $\mathcal{P}$ be a protocol for f and let L_1 be the subset of the set of leaves L of $\mathcal{P}$ in which the leaves are labeled 1. For each leaf $l \in L_1$, define a matrix M_L by $M_L(x, y) = 1$ for $(x, y) \in R_l$ and $M_L(x, y) = 0$ for $(x, y) \notin R_l$. Observe that for every (x, y) for which $f(x, y) = 0$, the (x, y) entry is 0 in a single matrix M_L. For every (x, y) for which $f(x, y) = 1$, the (x, y) entry is 1 in a single matrix M_L. Hence $M_f = \sum_{l \in L_1} M_l$. From linear algebra, we know that $rank(M_f) \leq \sum_{l \in L_1} rank(M_l)$. For each $l \in L_1$, we have $rank(M_l) = 1$. The is because the 1's of M_l are "markers" for those entries (x, y) of M_l that end up in R_l, which is a 1-rectangle, i.e., a monochromatic rectangle whose elements have 1 as their f-value. It follws that the nonzero rows of M_l are identical, and so M_l has exactly one linearly independent row. We now have:

$$rank(f) = rank(M_f \leq \sum_{l \in L_1} rank(M_l) = \sum_{l \in L_1} 1 = |L_1| < |L|$$

Hence by lemma 18 and corollary 19, $D(f) \geq \log_2 |L| \geq \log_2 rank(f)$. □

Corollary 22. *For any function $f : X \times Y \to \{0, 1\}$, we have:*

$$D(f) \geq \log_2(2rank(f) - 1)$$

Proof. Note that the proof of lemma 21 actually gives a lower bound on $|L_1|$, the number of 1-rectangles, rather than on $|L|$. We can get a lower bound on $|L_0|$, the number of 0-rectangles as follows: By switching the role of 1 and 0, we can look at $M_{\neg f}$, and the proof of lemma 21 gives a lower bound on $|L_1|$ for $M_{\neg f}$, which is a lower bound on $|L_0|$ for M_f. Let J be the all-1 matrix. Then $M_f = J - M_{\neg f}$ we have $rank(J) = 1$ and $rank(-M_{\neg f}) = rank(M_{\neg f})$, again from linear algebra. So $rank(M_f) \leq rank(J) + rank(-M_{\neg f})$. We now have

$$|L| = |L_1| + |L_0| = \tag{1}$$

$$\sum_{l \in L_1} 1 + \sum_{l \in L_0} 1 = \tag{2}$$

$$\sum_{l \in L_1} rank(M_l) + \sum_{l \in L_0} rank(M_l^{\neg f}) \geq \tag{3}$$

$$rank(M_f) + rank(M_{\neg f}) \geq \tag{4}$$

$$rank(M_f) + rank(M_f) - 1 \tag{5}$$

Hence we have $\log_2 |L| \geq \log_2(2rank(M_f) - 1)$. □

Exercise 2. *Alice and Bob each hold an n-bit string, $x, y \in \{0,1\}^n$. The equality function $EQ(x,y)$, is defined to be 1. if $x = y$ and 0 otherwise. Prove that $D(EQ) = N + 1$.* Exercise