CS 2401 - Introduction to Complexity Theory

Lecture #11: Dec 8th, 2015

Lecturer: Toniann Pitassi

Scribe Notes by: Xu Zhao

1 Communication Complexity Applications

Communication Complexity (CC) has many applications in computer science. Specifically, CC lower bounds for Set Disjoint problem is useful to prove many results in a large variety of contexts.

- Set disjoint problem
- Streaming problem
- Monotone formula and circuit complexity
- Proof complexity

2 Set Disjoint Problem

Definition The set disjoint problem is defined as follows. Suppose Alice has a number $x \in \{0,1\}^n$ and Bob has a number $y \in \{0,1\}^n$. We denote the disjoint function $DISJ(x,y) = 1$ if and only if there exists an integer i, such that $x_i = y_i = 1$.

Theorem 1 *$DISJ$ function requires $\Omega(n)$ bits communication for deterministic protocols.*

Proof Consider all (x,y) pairs that $x \oplus y = 1$. Each pair of (x,y) is in its own rectangle in the communication matrix and there are 2^n such pairs. For any (x,y) that $x \oplus y = 1$, we have $DISJ(x,y) = 0$. Given arbitrary two such pairs, (x,y) and (x',y'), we have either $DISJ(x',y) = 1$ or $DISJ(x,y') = 1$. So $DISJ$ function has a fooling set of size 2^n, thus its deterministic communication complexity is $\Omega(n)$.

Note that the $DISJ$ function has a $O(1)$ non-deterministic communication protocol. This is because each of Alice and Bob can guess the value of i then send each other's bit x_i and y_i, then compare them and output the result.

Meanwhile, we claim that $DISJ$ function requires $\Omega(n)$ bits communication for all randomized protocols. The proof of this claim requires knowledge on Information Theory and is not covered in this lecture.

3 Streaming Problem Complexity

Suppose the input is a stream of data, the streaming complexity of a problem is how much space it would need to store in the worst case processing the stream. We use the following frequency moment problem as an example to show how to use communication complexity to prove streaming problem complexity lower bounds.

Definition Suppose $S \in [n]^m$ is a length-m stream and S_j is the j-th element in S. Let $M_i = |\{\sum j \in [m] | S_j = i\}|$, so M_i equals to number of times that i appear in the stream. We further define $F_k = \sum_{i=1}^{n} M_i^k$. Hence F_0 is the number of distinct elements in the stream, F_1 is the length of the stream, and F_∞ is the number of occurrences of the most frequent item in S.

Theorem 2 *If we have a small space algorithm for F_∞, we can solve $DISJ$ with small communication complexity.*

Proof Suppose Alice has a number $x \in \{0,1\}^n$, Bob has a number $y \in \{0,1\}^n$. For Alice, she can generate a stream of numbers $a_x = \{i | x_i = 1\}$, then run algorithm F_∞ on a_x. This computation will only take a small space based on our assumption. Suppose it takes s bits of space. Then Alice can send these s bit space after the compuation of $F_\infty(a_x)$ to Bob. Bob will continue to use these s bit space to simulate the computation on b_y, where $b_y = \{i | y_i = 1\}$. The output will be the number of occurrences of the most frequent item in $a_x \cup b_y$. If the output number is larger than one, we know there is some number appear more than once in $a_x \cup b_y$, which means $DISJ(x,y) = 0$. Otherwise $DISJ(x,y) = 1$.

4 Circuit Complexity

In this section we will discuss the relation between circuit depth lower bound and communication complexity.

Definition Clique problem. For $k = n/10$, given an n-node undirected graph $G = (V, E)$, does it contain a clique of size k? (A k-sized clique means there is a subset $S \subseteq V$, $|S| = k$, such that $\forall i, j \in S, < i, j > \in E$.)

The clique problem is a problem having the monotone property. It takes a n^2-sized vector as input (could be adjacent matrix or adjacent list representation of this graph). If the answer of the clique problem on this input graph is true, then it means adding new edge into this graph, the graph will keep the property that contain a k-clique. In other words, changing any 0 to 1 in the input will only make the output larger, which means larger input is more likely to have larger output.

Theorem 3 *Any monotone circuit for the clique problem requires depth $\Omega(\sqrt{n})$. This implies any monotone formula for clique requires size $2^{\Omega\sqrt{n}}$.*

Monotone circuits means circuits with AND and OR gates but without negation. This is because changing any input bit from 0 to 1 will make the output more likely to output 1. For any

circuit that requires depth $\Omega(\sqrt{n})$, we claim that it requires formula size $2^{\Omega\sqrt{n}}$. This is because any formula with size S can be balanced to depth $\log(S)$.

So we will show how to reduce the depth of circuit solving a function f to a communication complexity solving a Karchmer-Wigderson game $R(f)$.

Definition Suppose we have a boolean function solving a search problem: $f : \{0,1\}^n \to \{0,1\}$ where the output indicates if the search result is true for false. For example, for the searching clique problem the input of f could be a graph and f returns whether this graph has a clique of size $k = n/10$.

We define $R(f)$ as the following problem: at the beginning, Alice gets $x \in f^{-1}(1)$ and Bob gets $y \in f^{-1}(0)$. For the clique problem example, Alice gets a graph that contains a clique of k vertices. Meanwhile, Bob gets a maxterm $k-1$ non-clique graph, which means adding even one edge will make the graph contain a clique of size k. Output of $R(f)$ will be the first i such that $x_i \neq y_i$. If f is monotone, then $R(f)$ is to output the first i such that $x_i = 1$ and $y_i = 0$.

Theorem 4 *Let f by any function $\{0,1\}^n \to \{0,1\}$. $CC(R_f)$ equals to the minimal depth of the circuit that solves f. If f is monotone, then $CC(R_f)$ equals to the minimal depth of monotone circuit solving f.*

Proof We show that $CC(R_f) \leq depth(f)$ by reducing a circuit that computes f to a protocol that solves R_f. The circuit can be shown as a tree of gates. We can assume that the top of the circuit is an OR gate and next level is a AND gate, and so on. Suppose Alice has $x \in f^{-1}(1)$ and Bob has $y \in f^{-1}(0)$. Because $f(y) = 0$, for $f(y)$ there must be a gate on the next level of circuit whose output is different with $f(x)$. So Alice and Bob will communicate, telling each other which gate their output start to diverse. Then they recurse to a lower level in the circuit. At this time we meet the same problem as before, but on a smaller subcircuit. We recurse until we meet to the leaves of the circuit, where the input is simply one of bits on x and y, where we can identify the index of difference between x and y. We send one message for every level of circuit that we traverse down, so the total communication complexity of $R_f \leq depth(f)$. Here we give an example of how we build the protocol.

Example

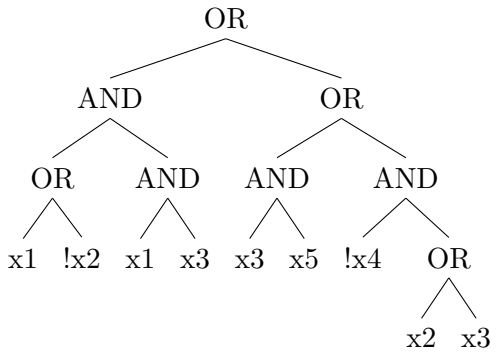

Figure 1: Convert a circuit into a communication protocol.

Suppose Alice has input 01101 and Bob has input 01010. Then on the circuit above the sequence of bits sent would be as follows.

1. Alice: 0 (go right)
2. Bob: 1 (go left)
3. Alice: 0 (go left)
4. Both Alice and Bob reach x_3 bit on which they differ.

Now we will show $depth(f) \leq CC(R_f)$. Given a protocol for R_f, we can also construct a circuit computing f with depth $CC(R_f)$. Suppose R_f takes x for Alice and y for Bob, it outputs z as the result. Consider a protocol tree T for R_f and we can convert T into a circuit as follows:

1. For each node where the message is sent by Alice, replace the node with an OR gate.
2. For each node where the message is sent by Bob, replace the node with an AND gate.
3. At each leaf of the protocol tree, with associated monochromatic rectangle AxB and input bit i, one of the following holds:
   (a) $\forall \alpha \in A, \alpha_i = 1$ and $\forall \beta \in B, \beta_i = 0$
   (b) $\forall \alpha \in A, \alpha_i = 0$ and $\forall \beta \in B, \beta_i = 1$

   If (a) holds, we assign the leaf node to z_i, otherwise we assign it to $\bar{z_i}$.

5 Proof Complexity

In this section we will show the relation between communication complexity and proof complexity. Proof complexity is about the length of proof that is needed for proving a theorem. For example, an open problem in proof complexity field is about how to prove or disprove there exists a family of 3DNF tautologies $\{f_n | n \text{ sufficient large}\}$ where each f_n is a 3DNF tautology and n variables, such that any f_n requires Frege proof of size superpolynomial in n.

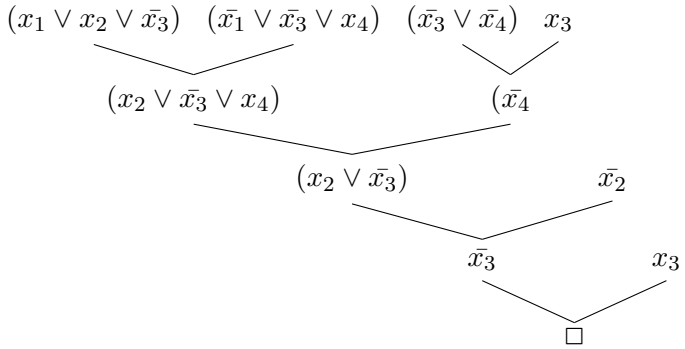

Figure 2: Resolution refutation proof tree of an unsatisfiable formula.

Definition Resolution proofs. Resolution refutation can show a CNF formula is unsatisfiable. Resolution rule is the only inference rule in this system. It produces a new clause implied by two clauses containing complementary elements. Two elements are complementary if one is the negation of the other. For example $\neg c$ is complement to c. The resulting clause contains all the elements that do not have the complements. If the proof finishes with an empty clause, then the original formula is unsatisfiable. Resolution rule is known to be both sound and complete. Figure 1 shows a resolution proof tree of an unsatisfiable CNF formula $(x_1 \vee x_2 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_3} \vee x_4) \wedge \bar{x_2} \wedge x_3 \wedge (\bar{x_3} \vee \bar{x_4})$. Size of a resolution refutation proof is the total number of clauses in the proof.

Lemma 5 *If f has a resolution refutation proof of depth d, then $search_f$ has a communication protocol of cost $O(d)$.*

Proof We can reduce depth of a resolution proof tree to communication complexity of a search problem. Let f be an unsatisfiable CNF formula. We define the problem $search_f$ where the variables in f are partitioned into two groups: x and y. Alice gets assignment α to x variables and Bob gets assignment β to y variables. The taget is to find out a clause C_i in f, such that $C_i(\alpha, \beta) = 0$. We can do it starting from the root of the proof tree towards leaves of the tree. For each child clause of root, Alice will evaluate all literals for the clauses using α and send the result to Bob and Bob will do the same thing. If either Alice or Bob get result 1 it means this clause is true under (α, β), otherwise the clause is false. Because the resolution rule is sound, the assignment (α, β) is guaranteed to falsify at least one branch clause, where Alice and Bob will start to go one step down until they reach the leaf node. In the end they will find a clause of f is falsified. It takes at most d steps so $CC(search_f) = O(d)$.

Note that this proof can be generalized to any sound proof system that can be represented by a 2-player communication protocol.

Theorem 6 *Any resolution proof of f_n requires a resolution tree of depth $\Omega(n)$ and tree size $2^{\Omega(n)}$.*