University of California, Los Angeles
CS 289 Communication Complexity

Instructor: Alexander Sherstov
Scribe: Pratik Chaudhari
Date: October 20, 2014

LECTURE
5

Log-rank conjecture and nondeterministic complexity

In this lecture, we first take a look at the log-rank conjecture. It has been conjectured that the deterministic communication complexity is essentially characterized by the log-rank of the characteristic matrix. In particular, it is at most the log-rank raised to a constant c. This is a major open problem and while the conjecture was first proposed with $c = 1$, we will show here that c in fact might be large.

We then introduce nondeterministic communication complexity where Alice and Bob are allowed to "guess" while executing the protocol. This is analogous the "proof system" approach, roughly speaking, Alice and Bob guess the smallest string that helps them to compute $f(x, y)$ without much communication. Just as we used partitions in the previous lectures to analyze deterministic complexity, we introduce covers of the characteristic matrix M_f; these effectively characterize the nondeterministic communication complexity of a function f.

Lastly, we compare and contrast the two notions of complexities. In particular, we show that the gap between deterministic and nondeterministic complexity can be exponentially large. This is surprising because we do not know whether such a result is true for the complexity classes P and NP. However, we also show some functions f where the nondeterministic complexity is almost as large as the deterministic complexity. We end the class after a brief discussion about the fact that nondeterministic and co-nondeterministic complexities cannot both be small.

5.1 Log-rank conjecture

We start the lecture with a discussion on the log-rank conjecture. We have already seen that log-rank of the characteristic matrix M_f of a function $f : X \times Y \to \{0, 1\}$ is a very good lower bound on the deterministic communication complexity $D(f)$ [4]. A prominent conjecture however is the following which states that $D(f)$ is essentially upper-bounded

by the log-rank of M_f. This was first proposed by Lovasz and Saks [3] in a slightly different form.

CONJECTURE 5.1. For some constant $c > 0$, for all functions f,

$$\log \mathrm{rk}(M_f) \leq D(f) \leq \left[\log \mathrm{rk}(M_f)\right]^c + c.$$

REMARK 5.2. The original conjecture was proposed with $c = 1$. However we will see today that c is essentially large. In particular we shall prove that $D(f) \geq \left[\log \mathrm{rk}(M_f)\right]^{1.58}$. Note that we have already proved the first inequality of the conjecture in previous classes, the second inequality on the other hand is a hard open problem in the field. To put things into perspective, we proved in the last lecture that

$$D(f) \leq \mathrm{rk}_{\mathbb{R}}(M_f) + 1$$

which is an exponentially-worse upper bound than what the conjecture predicts.

Let us introduce the following theorem which quantifies the above discussion.

THEOREM 5.3 (Nisan and Wigderson [5]). *There is a function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *such that*

$$D(f) = \Omega(n) \qquad \text{but,}$$
$$\log \mathrm{rk}(M_f) \leq O\left(n^{\frac{1}{1.58\ldots}}\right).$$

REMARK 5.4. First note that the above theorem implies $D(f) \geq \left[\log \mathrm{rk}(M_f)\right]^{1.58\ldots}$ which means that the constant c in Conj. 5.1 is greater than 1.58. To prove this theorem, we will construct a "gadget" and compose it with itself many times to construct a complicated function which will give us the above bounds.

Proof. The basic idea of the proof is to construct a Boolean function with low degree and high "sensitivity". Define a real-valued polynomial with three binary inputs $h : \{0,1\}^3 \to 0, 1$ as

$$h(z_1, z_2, z_3) = z_1 + z_2 + z_3 - z_1z_2 - z_2z_3 - z_1z_3.$$

Note that h is symmetric with respect to its arguments. We can now see that if exactly one of the zs is 1, $h = 1$, if exactly two zs are 1, we have $h = 1$. For all other cases, $h = 0$. In order words,

$$h(z_1, z_2, z_3) = \begin{cases} 1 & \text{if } z_1 + z_2 + z_3 = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Define a recursive function H_d as shown in Fig. 5.1. To construct H_d, compose h with itself d times. In more cumbersome notation, for $d > 1$

$$H_d = h(H_{d-1}, H_{d-1}, H_{d-1}); \qquad H_1 = h.$$

H_d thus has 3^d leaves.

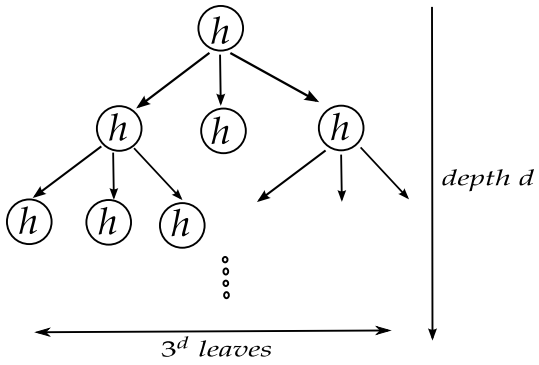

FIGURE 5.1: H_d is a function of 3^d inputs.

We can also compute H_d for a few special cases which we need, the others we do not care for.

$$H_d(z_1, z_2, \ldots, z_{3^d}) = \begin{cases} 1 & \text{if } \sum_i z_i = 1 \\ 0 & \text{if } \sum_i z_i = 0 \\ \text{don't care} & \text{otherwise.} \end{cases}$$

Let $f(x, y) = H_d(x \wedge y)$, i.e., the length of the inputs of Alice and Bob, $n = 3^d$. To prove this theorem we will use a result which we have not proved yet. The disjointness problem DISJ_n is really one of the cornerstones in the theory communication complexity and plays a large role in deriving important results. Well, in any case, we use the fact that

$$D(\text{DISJ}_n) \geq \Omega(n) = \Omega(3^d),$$

which can be proved using the fooling-set method. See [1] for a wonderful introduction to the disjointness problem.

Why should we expect $\text{rk}(M_f)$ to be low? The answer is that because the degree of h is just 2 instead of the full-degree, which'd be 3. Thereby, the degree of H_d which is just h composed d times with itself, is at most 2^d.

PROPOSITION 5.5. *If $H_d = \sum_S c_S z_S$ where z_S are monomials and c_S are their coefficients (S runs over all subsets of $\{1, \ldots, n\}$), the rank $\text{rk}(M_f)$ is at most the number of monomials z_S that make up H_d, i.e.,*

$$\text{rk}(M_f) \leq \text{number of monomials in } H_d$$

Proof. Note that

$$\begin{aligned} M_f &= [H_d((x_1, y_1), \ldots, (x_n, y_n))] \\ &= \sum_S c_S M_S, \end{aligned}$$

where M_S is the matrix defined by $M_S(x, y) = \prod_{i \in S} x_i \cdot y_i$. The entries of M_S where either of the row or column do not lie in S are zero. Now for every set S, the rank of M_S is just

1. Using the rank inequalities, we can see that the rank $\text{rk}(M_f)$ is bounded above by the number of non-zero monomials of f. □

PROPOSITION 5.6. *We also have*

$$\text{number of monomials in } H_d \leq 6^{2^d-1}.$$

Proof. The proof of this proposition is a simple induction argument. Let m_d be the number of monomials in H_d, note that $m_0 = 6$. In general,

$$m_d \leq 3m_{d-1} + 3m_{d-1}^2; \qquad d = 1, 2, 3, \ldots$$

Thus $m_d \leq 6m_{d-1}^2$ which gives $m_d \leq 6^{2^d-1}$. □

Together, the two propositions imply that

$$\begin{aligned} D(f) &\geq \Omega(3^d) \\ &\geq \Omega\left(\left[\log \text{rk}(M_f)\right]^{\log 3}\right). \end{aligned}$$

□

REMARK 5.7. To repeat, the above theorem implies that $c > 1.58$. Let us note that c can be improved slightly, using a gadget with 7 variables instead of 3. These gadgets are typically discovered using computer search and the one with 7 variables is the best result known to us yet.

A key takeaway from the development of this theorem is that rank bound is a very powerful method for computing the deterministic communication complexity. It works for all the problems that we (and other researchers elsewhere) have tried so far. The essential part of this program is however finding the right field to work in while computing the rank, most problems often become surprisingly easy when we use the right finite field.

5.2 Non-deterministic communication complexity

Let us now introduce nondeterministic communication complexity. This was first discovered while analyzing the complexity of fabrication of large-scale digital circuits for various arithmetic tasks [2]. In the next few classes, we will explore the power of nondeterminism and contrast it with deterministic communication complexity. This model is analogous to nondeterministic *computational* complexity and can be thought of as a "proof system". Roughly, nondeterministic communication complexity of a function f, denoted by $N(f)$, is the smallest number of bits that can be used to convince both Alice and Bob that $f(x, y) = 1$. This is conceptually similar to NP languages, given the polynomial-length certificate, we can efficiently check if some input $x \in L \in \text{NP}$.

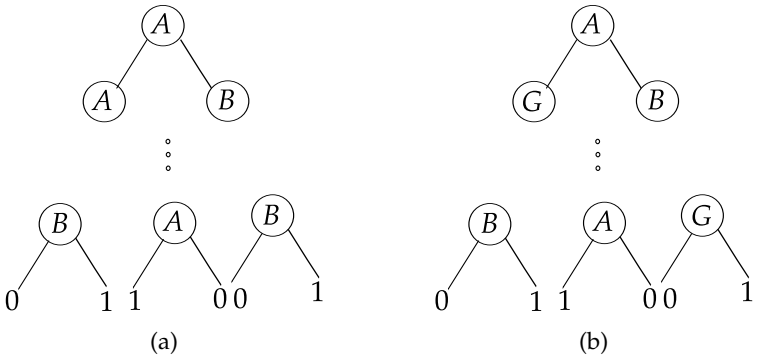

FIGURE 5.2: Fig. 5.2a shows a deterministic protocol. On the other hand, Fig. 5.2b depicts the protocol tree for a nondeterministic protocol with guess nodes marked by G.

5.2.1 The model

We know that deterministic protocols are binary trees. Each node of the tree is labeled either by a function $a_v : X \to \{0,1\}$ which corresponds to the bit sent by Alice or by a function $b_v : Y \to \{0,1\}$ which corresponds to Bob's bit. In addition to this, nondeterministic protocols have a special kind of node, a "guess node" which we will denote by $g_v \in \{0,1\}$. At any point in the protocol execution, Alice and Bob can get together and "publicly" guess a value of $g_v \in \{0,1\}$ upon which the protocol either moves left or right respectively in the protocol tree. An example nondeterministic protocol tree thus looks like the one shown in Fig. 5.2. Note that while deterministic protocol trees have a unique path in the binary tree for a given pair of inputs x, y, the same is not true for nondeterministic protocols. Indeed, the power of nondeterminism lies in the fact that Alice and Bob can periodically guess together and branch.

We have thus defined the syntax of a nondeterministic protocol tree. The semantics of this are as follows:

1. at any point of time Alice or Bob can toss a coin to "guess" and outcome of this guess is public;
2. coin tosses count towards communication cost;
3. future messages depend upon previous messages as well as past coin-tosses;
4. if $f(x,y) = 0$, the protocol must output a zero on *all executions*, in other words, we do not tolerate any false positives;
5. on the other hand, if $f(x,y) = 1$, the protocol only has to output one on at least one of the executions.

DEFINITION 5.8 (Nondeterministic cost). The cost of a nondeterministic protocol is the worst case cost on any input and is equal to the sum of the number of bits exchanged and

the number of coin tosses, i.e.,

$$N(f) = (\#\text{ of bits exchanged}) + (\#\text{ of coin tosses}).$$

REMARK 5.9. Note that we are still following the usual definition for *cost of a protocol*, it is simply the depth of the nondeterministic protocol tree. Also, note that coin tosses are costly, if we allow zero-cost coin tosses, any function f will have a nondeterministic communication complexity of exactly 2 bits. The following protocol achieves this — Upon input x, y, Alice and Bob together guess some $x' \in X$. Output 1 if $x' = x$ and $f(x', y) = 1$ and zero otherwise. Note that Alice can check if $x' = x$, send the result to Bob who in turn computes the value of $f(x', y)$. This protocol has a cost of exactly 2 bits if Alice and Bob can guess x' without any cost! Note that it however takes n bits to guess a random x' and we get the trivial bound $N(f) \leq n + 2$ (which is true for any function f) using this protocol.

PROPOSITION 5.10. *The nondeterministic communication complexity of any function $f : X \times Y \to \{0, 1\}$ is at most the deterministic communication complexity of f, i.e.,*

$$N(f) \leq D(f).$$

Proof. Indeed, any deterministic protocol is also a nondeterministic protocol where Alice and Bob are happy with the general scheme of things and choose not to guess at all. □

Admittedly, the above discussion on nondeterministic protocols is rather abstract, let us discuss a few examples which will make the notion much more natural.

EXAMPLE 5.11. The first example is the inequality problem, denoted by $\neg\text{EQ}_n$. Given two strings $x, y \in \{0, 1\}^n$, we need to check if $x \neq y$. The deterministic complexity of doing so is $n + 1$ bits, Alice simply sends her input to Bob. Here is a nondeterministic protocol for the same problem.

1. Guess an index $i \in \{1, \ldots, n\}$;
2. Output
$$\begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{else.} \end{cases}$$

First, let us compute the cost of this protocol. The first step costs $\log n$ bits while the second is just 2 bits. We thus have $N(\neg\text{EQ}_n) = \log n + 2$. Why is this protocol a valid nondeterministic protocol? If $x \neq y$, there is at least one index on which they differ. Thus the protocol returns a one on at least one of the execution traces (precisely, the one where it guesses this index). If the two strings are equal, there does not exist any such index and the protocol returns zero. According to the "proof system" analogy, the certificate for two strings being unequal is simply the index on which they differ. Note that $D(\text{EQ}_n) = D(\neg\text{EQ}_n) = n + 1$. How much is $N(\text{EQ}_n)$? Stay tuned ...

EXAMPLE 5.12. Let us discuss the problem of set intersection, denoted as $\neg\text{DISJ}_n$. Given two subsets $A, B \subseteq \{1, \ldots, n\}$, check if $A \cap B \neq \emptyset$. The nondeterministic protocol is as follows.

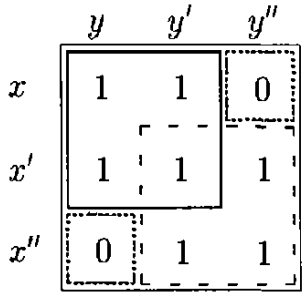

FIGURE 5.3: An example cover for the characteristic matrix. The cover number $C(f)$ is therefore 2.

1. Guess an element i;
2. Output:
$$\begin{cases} 1 & \text{if } i \in A \cap B \\ 0 & \text{else.} \end{cases}$$

Again, let us compute the cost of the protocol; it takes $\log n$ bits to guess an element i and 2 bits for the second step of the protocol. Thus $N(\neg \text{DISJ}_n) = \log n + 2$. It is instructive to verify if this is a valid nondeterministic protocol. It never gives false positives, if $A \cap B = \emptyset$, there does not exist any element i that results in an output of 1 in the second step. The protocol always returns zero. Also, if $A \cap B \neq \emptyset$, a good certificate for this is simply the element that lies in both A and B, our nondeterministic protocol guesses this element in $\log n$ bits.

5.2.2 Nondeterminism and covers

In the deterministic case, we saw that the log-rank effectively characterizes the communication complexity. In the nondeterministic case, we will see that the analogous concept is a "cover". Covers are more convenient combinatorial objects than partitions and the relation between covers and nondeterminism is very natural. Before we introduce notation, Fig. 5.3 shows an example cover. All the 1s in the matrix shown in Fig. 5.3 can be covered by 2 monochromatic rectangles.

DEFINITION 5.13 (Cover number). The cover number of a function $f : X \times Y \to \{0,1\}$, denoted by $C(f)$ is the minimum number of f-monochromatic rectangles $R_1, R_2, \ldots$ required to cover all the 1s in the characteristic matrix M_f, i.e.,

$$\cup R_i = f^{-1}(1).$$

Note that R_is in the above definition may overlap, we simply require that their union be $f^{-1}(1)$. The following theorem now effectively characterizes the nondeterministic communication complexity in terms of the cover number.

Note that we have not explicitly stated the notion of a rectangle for nondeterministic protocols. There are two ways to see this and they are equivalent. One can simply consider the characteristic matrix M_f and an f-monochromatic rectangle R is a set which can be written as a product of two sets $R_X \times R_Y$ where $R_X = \{x \mid \exists\, y \text{ s.t. } (x,y) \in R\}$ and $R_Y = \{y \mid \exists\, x \text{ s.t. } (x,y) \in R\}$. Alternatively, consider two input pairs (x,y) and (x',y'). Let the guesses for Alice and Bob for these two pairs be (z_1,z_2) for input (x,y) and (z_1',z_2') for input (x',y') respectively. If $(x,y),(x',y') \in f^{-1}(1)$, the same way that deterministic protocols partition the leaves of the protocol tree into rectangles, the messages that are sent for inputs (x',y) with guesses (z_1',z_2) and (x,y') and guesses (z_1,z_2') are the same. In other words, the path in the protocol tree is the same. Thus, all the inputs $(x,y),(x',y'),(x',y),(x,y')$ reach the same leaf and lie in the same 1-rectangle.

THEOREM 5.14. *For all functions f,*

$$\log C(f) \leq N(f) \leq \log C(f) + 3.$$

Proof. Let us prove the second inequality first. If $c = C(f)$, let $R_1, \ldots, R_c$ be a cover for 1s in M_f. Here is a protocol for computing f.

1. Guess the index of rectangle i;
2. Output
$$\begin{cases} 1 & \text{if } (x,y) \in R_i; \\ 0 & \text{else.} \end{cases}$$

Note that it takes $\lceil \log c \rceil$ bits to guess an index and 2 bits for the second step. In total, $N(f) \leq \log c + 3$.

The first inequality can be proved in a similar way as the deterministic communication complexity version. Fix a nondeterministic protocol for f with cost $N(f)$. Let

$$R_l = \{(x,y) \text{ such that leaf } l \text{ is reachable on input } (x,y)\}.$$

□

5.2.3 Deterministic vs. Nondeterministic complexity

Let us introduce some relationships between deterministic and nondeterministic complexities. The results here are quite surprising when contrasted with similar conjectures in computational complexity. Indeed, as far as communication complexity is concerned, the gap between deterministic and nondeterministic complexities is exponentially large. Moreover, we also know examples where the gap is precisely that big.

THEOREM 5.15. *For all functions f,*

$$N(f) \leq D(f) \leq 2^{N(f)} + 1.$$

Proof. The first inequality is trivial and we have seen it before. The second inequality is striking. Let us prove it below. As usual, let $c = C(f)$ and let $R_1, \ldots, R_c$ be rectangles that form a 1-cover. Here is a deterministic protocol for computing $f(x, y)$.

1. Alice sends Bob, $b_1, \ldots, b_c$ where each b_i is 1 if $x \in R_i$ and zero otherwise;
2. Bob outputs
$$\begin{cases} 1 & \text{if } y \in R_l \text{ and } b_l = 1; \\ 0 & \text{else.} \end{cases}$$

It is clear that the cost of this protocol is c bits for step 1 and 1 bit for step 2, i.e.,

$$D(f) \leq c + 1 \leq 2^{N(f)} + 1.$$

□

REMARK 5.16. Thm. 5.15 is tight on both inequalities. We have shown before that $D(\neg \mathrm{EQ}_n) = n + 1$, Alice simply sends her entire input. On the other hand, the nondeterministic complexity of this is $C(\neg \mathrm{EQ}_n) = \lceil \log n \rceil + 2$, we guess the index of the errant element in the two strings. This shows that

$$D(f) \geq \Omega\left(2^N(f)\right); \qquad f = \neg \mathrm{EQ}_n.$$

On the other hand, consider $f = \mathrm{EQ}_n$, for which we know that $D(f) = n + 1$. It can be shown using the fooling set method that certificate for EQ_n is at least 2^n bits long, which implies $N(f) \geq n$.

Observe that $D(f) = D(\neg f)$, Alice and Bob simply output the opposite of what they would have given for (x, y) after executing the protocol. However, $N(f)$ and $N(\neg f)$ are vastly different. While the gap between $D(f)$ and $N(f)$ can be exponential as shown in Thm. 5.15, we have the following remarkable result that

$$D(f) \leq 2N(f)N(\neg f).$$

We will prove this in the next lecture. $N(\neg f)$ is the co-nondeterministic communication complexity. This result thus says that either the nondeterministic or co-nondeterministic complexity of a function f can be small, but not both. (Stretching the analogy with complexity theory even further) contrast this with the open problem that asks whether $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ [6]. In the other direction, if f has nondeterministic and co-nondeterministic complexity of t, the deterministic complexity can be at most $O(t^2)$.

References

[1] A. Chattopadhyay and T. Pitassi. The story of set disjointness. *ACM SIGACT News*, 41(3):59–85, 2010.

[2] R. J. Lipton and R. Sedgewick. Lower bounds for VLSI. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 300–307. ACM, 1981.

[3] L. Lovász and M. Saks. Lattices, mobius functions and communications complexity. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 81–90. IEEE, 1988.

[4] K. Mehlhorn and E. M. Schmidt. Las Vegas is better than determinism in VLSI and distributed computing. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 330–337. ACM, 1982.

[5] N. Nisan and A. Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995.

[6] C. H. Papadimitriou and M. Sipser. Communication complexity. *Journal of Computer and System Sciences*, 28(2):260–269, 1984.