Communication Complexity and Applications

Lecturer: Toniann Pitassi

Scribe Notes by: Yuval Efron

1 Applications

In the coming lectures lecture, we go over several applications of communication complexity in other areas of theoretical computer science such as streaming, property testing, game theory, time/space trade-offs, circuit complexity, proof complexity, extension complexity, graph theory.

Throughout the lecture, we frequently make use of the following results.

$$\mathsf{BPP}^{\mathsf{CC}}(\mathsf{DISJ}_n) = \Omega(n)$$
$$\mathsf{BPP}^{\mathsf{CC}}(\mathsf{UDISJ}_n) = \Omega(n)$$
$$\mathsf{coNP}^{\mathsf{CC}}(\mathsf{UDISJ}_n) = \Omega(n)$$

Theorem 1. *The k-player Number on Forehead(NOF) communication complexity of DISJ_n, UDISJ_n is $\Omega(\frac{n}{2^k})$.*

We will see the proofs of these result in future lectures.

1.1 Streaming

The material in this section is based on the lecture notes of Tim Roughgarden [10]. In the streaming model of computation, we make a single pass over a given input while maintaining some local memory M of size s, the goal is to compute some property of the input while minimizing s. Specifically, we focus on computing frequency moments. A bit more formally, in the frequency moments problem we have the following.

- The stream consists of a string $S \in [n]^m$, which we observe one element at a time.
- Let $M_i = |\{j \in [m] \mid S_j = i\}|$. The k-th frequency moment is defined to be $F_k = (\sum_{i=1}^{n} M_i^k)^{1/k}$. For example F_0 is the number of distinct elements in the stream. $F_1 = m$, i.e. the length of the stream. F_∞ is the number of occurrences of the most frequent element.
- The goal: Approximate F_k for various values of k.

For some values of k, F_k can be efficiently approximated in the streaming setting. Specifically, the following by Alon, Matias and Szegedy [1] holds.

Theorem 2. *There is a streaming protocol $\mathcal{A}$ that on a given stream S computes w.p. at least $1-\delta$ a multiplicative $(1+\epsilon)$- approximation of F_0, F_2 using a memory of size $s = O(\frac{(\log n + \log m)\log\frac{1}{\delta}}{\epsilon^2})$.*

Alas, in this course we are mainly interested in lower bounds, and we state the above result mainly to emphasize that proving lower bounds for computing frequency moments is not a trivial task. Specifically, we show that any streaming algorithm approximating F_∞ with $\epsilon = 0.99$ and $\delta = \frac{1}{3}$ must have space $s = \Omega(\min\{n, m\})$. Note that such a lower bound is essentially tight due to the following naive algorithms that compute moments exactly: Use m memory to store the entire stream and compute F_∞, or maintain a histogram of occurrences of size $O(n \log m)$ and compute deduce the most frequent elements and its number of occurrences. For simplicity, we assume that $m = n$.

Theorem 3. *Approximating F_∞ within a factor of 1.99 w.p. at least $\frac{2}{3}$ requires $s = \Omega(n)$.*

Proof. We prove the theorem by reduction from DISJ. Let $\mathcal{A}$ be some streaming algorithm approximating F_∞ of a given stream $S \in [n]^m$ with space s. Given $\mathcal{A}$, consider the following protocol for solving DISJ_n with Alice and Bob receiving inputs x, y respectively.

From x, Alice constructs the following set $a_x = \{i \mid a_i = 1\}$, and similarly Bob constructs b_y. Then Alice simulates $\mathcal{A}$ on a_x by inserting into the stream the elements of a_x in some arbitrary order. Denote by M the memory state of $\mathcal{A}$ after Alice inserted the last element of a_x, per the assumption, it consists of s bits, which Alice then sends to Bob. From there, Bob continues the simulation of $\mathcal{A}$ by inserting the elements of b_y in an arbitrary order. Finally, if $\mathcal{A}$'s output is < 2, Bob outputs x, y are disjoint, otherwise Bob outputs that x, y intersect.

Note now that if $x \cap y \neq \emptyset$, and let i be a coordinate s.t. $i \in x \cap y$, then $i \in a_x \cap b_y$ and in particular the number of occurrences of the most frequent element is at least 2, thus w.p. at least $2/3$ Bob outputs that x, y intersect as $\mathcal{A}$ answer will be strictly larger than 2 with that probability. If x, y dont intersect however, then a_x, b_y dont intersect as well, and thus the F_∞ value of the stream can be at most 1, which means that $\mathcal{A}$ answer is w.p. at least $2/3$ strictly smaller than 2, thus Bob outputs that x, y don't intersect with the same probability.

This protocol allows Alice and Bob to randomly solve $\mathsf{DISJ_n}$ with error probability at most $1/3$ while communicating $O(s)$ bits, thus $s = \Omega(n)$. $\square$

Remark 1. *It is possible to generalize the above result to show that for every $k \neq 1$, computing F_k* exactly *requires $\Omega(n)$ space in the streaming model.*

1.2 Property Testing

Next, we turn to applications in property testing. In property testing problems we are given a usually combinatorial object f from some predetermined domain D and some property $P \subseteq D$, the goal is to decide whether $f \in P$ or whether f is *far* from P, for an appropriate definition of distance between objects in D. Specifically, we will focus on functions $D \to R$ for some domain D and range R. Concrete examples include the following.

- **Linearity.** $D = \mathbb{F}^n$, $R = \mathbb{F}$, P is the set of linear functions. I.e. all functions $D \to R$ that satisfy $f(x + y) = f(x) + f(y)$ for all $x, y \in \mathbb{F}^n$.
- **Monotone functions.** $D = \{0,1\}^n$, $R = \{0,1\}$, P is the set of all monotone functions, i.e. functions f such that $f(x) \leq f(x')$ whenever $x \leq x'$.
- **Graph properties.** $D = \{0,1\}^{\binom{n}{2}}$, $R = \{0,1\}$. P is any graph property, e.g. contains a k-clique.

The goal in property testing is to design an algorithm that queries a function f on a very small set of inputs, and from that data deduces with high confidence whether $f \in P$ or is far from P.

Definition 1. *Let D, R be some domain and range, and let $f, g \in R^D$, we say that f, g are ϵ-far if $d(f, g) \geq \epsilon|D|$, where $d(\cdot,\cdot)$ is the hamming distance, i.e. number of entries on which f and g disagree.*

Linearity testing. As a first example, we consider linearity testing. We get query access to a function $f : \{0,1\}^n \to \{0,1\}$, and are tasked with deciding whether f is linear, or ϵ-far from linear. I.e., on inputs which are not linear or ϵ-far from linear, the algorithm is allowed to answer arbitrarily. A seminal result by Blum, Ruby, Rubinfeld [4], shows that the following simple algorithm solves the problem with one-sided constant error probability.

- Repeat the following $O(1/\epsilon)$ times:
    1. Sample $x, y \in \{0,1\}^n$ uniformly at random.
    2. If $f(x) \oplus f(y) \neq f(x \oplus y)$, reject
- Accept.

Note that the query complexity of the algorithm is $O(1/\epsilon)$, completely independent of n.

Monotone functions. As our next example, we consider the family of monotone functions, denoted by P, this is the set of functions $f : \{0,1\}^n \to R$, for some well-ordered range R s.t. $f(x) \leq f(x')$ whenever $x \leq x'$. The first result we discuss is the following upper bound by [6] for testing monotonicity in the Boolean case, i.e. $R = \{0,1\}$. We denote by (b, x_{-i}) the vector in which the i-th entry is b and the rest of the vector agrees with $x \in \{0,1\}^n$.

- Repeat the following $O(n/\epsilon)$ times:
    1. Sample i, x_{-i} uniformly at random.
    2. If $f(0, x_{i-1}) > f(1, x_{-i})$, reject.
- Accept.

Notice that unlike linearity testing, here the query complexity has linear dependency on n. Furthermore, clearly monotone functions are accepted by the above algorithm, and one can prove that any function which is ϵ-far form being monotone is rejected with constant probability.

[6] also shows that one can generalize the above algorithm to work any well-ordered range R by repeating the internal sample test $O(n|R|/\epsilon)$ times instead of $O(n/\epsilon)$. Another work [5] presented an algorithm that exponentially improves the dependence on $|R|$ in the sample complexity, i.e. $O(\frac{n}{\epsilon}\log|R|)$.

One can now ask whether this dependence on n is necessary. We address this question by presenting a lower bound proved of [3] that scales with the size of the range of the considered functions. Specifically, the following is shown in [3].

Theorem 4. *Any property testing algorithm for monotonicity with range $R = O(\sqrt{n})$ requires $\Omega(|R|^2)$ queries. Otherwise, the lower bound is $\Omega(n)$.*

We prove a slightly weaker version of the theorem, namely that if $|R| = \Omega(n)$, then $\Omega(n)$ queries are required. this theorem is proved via a reduction from UDISJ_n. The general template for the reduction is as follows.

- Map 1-inputs (x, y) to a function $h_{x,y} \in P$.
- Map 0 inputs to functions $h_{x,y}$ far from P
- Employ a tester Π for P and a short protocol to solve $\mathsf{UDISJ_n}$.

For the proof, we make use of the following Lemma.

Lemma 5. *For $A, B \subseteq [n]$, let $h_{A,B} : \{0,1\}^n \to \mathbb{Z}$ defined by $h_{A,B}(x) = 2|x| + (-1)^{|A\cap x|} + (-1)^{|B\cap x|}$. It holds that:*

- *$A \cap B = \emptyset$ implies that $h_{A,B}$ is monotone.*
- *$|A \cap B| = 1$ implies that $h_{A,B}$ is ϵ-far from monotone for some constant epsilon.*

From this lemma the reduction protocol is straight forward, if we assume Π is a monotonicity tester making q queries, and let A, B be inputs to UDISJ, we note that every query to $h_{A,B}$ made by Π can be simulated with 2 bits of communication, by exchanging $(-1)^{|A\cap x|}, (-1)^{|B\cap x|}$. When Π outputs an answer, we can solve UDISJ by answering according to the lemma.

All that is left is to prove the lemma.

Proof. Let A, B be disjoint and let S be some vector and i s.t. $i \notin S$, then we want to show that $h_{A,B}(S \cup i) - h_{A,B}(S) \geq 0$. Since A, B are disjoint, either $i \notin A$ or $i \notin B$, assume w.l.o.g that $i \notin A$, then we have that

$$h_{A,B}(S \cup i) - h_{A,B}(S) = 2 + (-1)^{|S\cap A|} + (-1)^{|(S\cup i)\cap B|} - (-1)^{|S\cap A|} - (-1)^{|S\cap B|} \geq 2 - 1 - 1 = 0.$$

Thus $h_{A,B}$ is monotone.

Now assume A, B intersect at some entry i, we making the following observation, whose proof we leave as an exercise.

Observation 6. *If $A \cap B \neq \emptyset$, then* $\Pr[|S \cap A|$ *is even and* $|S \cap B|$ *is even*$] \geq \frac{1}{4}$.

With this observation in mind, we can check that for sets S for which this holds we have that $h_{A,B}(S \cup i) - h_{A,B}(S) = -2$. Thus $h_{A,B}$ is not monotone at S for at least a quarter fraction of the S's, which is at least a $\frac{1}{8}$ fraction of the inputs. Thus assuming that $|R| = \Omega(n)$, $h_{A,B}$ is well defined and we can deduce from the communication lower bound on UDISJ_n that if Π tests whether a function is monotone or ϵ-far form monotone for $\epsilon < 1/8$, that Π makes $\Omega(n)$ queries. $\square$

The above can be generalized to show a lower bound of $\Omega(n)$ whenever $|R| = \Omega(\sqrt{n})$, and $\Omega(|R|^2)$ otherwise.

We remark that if one restricts the discussion to Boolean functions, the best lower bound stands at $\tilde{\Omega}(n^{1/3})$[2], and the best upper bound is $O(\sqrt{n})$[8].

1.3 Game Theory

Next we turn to applications in game theory, specifically the communication complexity of deciding whether Nash Equilibrium exists for 2 given payoff matrices A, B of size $n \times n$. Given such payoff matrices to players Alice and Bob, we call a pair (i, j) a *Nash Equilibrium* if i is the optimal Alice strategy given that Bob plays j, and vice versa.

Similarly to the previous section, we show by a reduction from $\mathsf{DISJ}_{\mathsf{n}^2}$ that deciding whether a Nash Equilibrium exists requires $\Omega(n^2)$ communication given inputs of size $n \times n$.

The reduction. Let Π be a protocol that decides whether a Nash Equilibrium exists for any 2 given matrices A, B of size $n \times n$ using $T(n)$ bits of communication. Let A, B be inputs of size n^2 for DISJ, which we view as $n \times n$ matrices. The reduction will pad the matrices with 2 additional rows and columns as follows: From A, we construct A' as follows: $A'[i, j] = A[i, j]$ for all $1 \le i, j \le n$, $A'[n+1, j] = 1$ for $j \in [n+1]$, $A'[n+1, n+2] = 0$, $A'[n+2, j] = 1$ for $j = n+2$ or $j \in [n]$, and $A'[n+2, n+1] = 0$. Finally, for the columns, $A'[i, n+1] = A'[i, n+2] = 0$ for all $i \in [n]$. For B we pad it in the same manner with *transposing* the added rows and columns, i.e. the $n+1$'th row of A appears as the $n+1$'th column of B, and vice versa with the columns, and similarly for $n+2$.

For a visual representation of the reduction, refer to the lecture slides on the course website.

One can check that now in A', B', which are matrices of size $(n+2) \times (n+2)$, there is a Nash Equilibrium iff there exists $i, j \in [n]$ such that $A[i, j] = B[i, j] = 1$, i.e. A, B intersect. Similarly to the previous results, from this we obtain a lower bound of $\Omega(n^2)$ on $T(n)$.

Approximate Nash Equilibrium. One can generalize the above setting to consider Approximate Nash Equilibrium, in which we call a pair of vectors (x, y) an ϵ-*Approximate Nash Equilibrium* if given any $x^T A y \ge x'^T A y - \epsilon$ for all x', and $x^T B y \ge x^T B y' - \epsilon$ for all y'. A recent result by Mika Goos and Aviad Rubinstein [7] showed that that randomized communication complexity of ϵ-approximate Nash Equilibrium is $n^{2-o(1)}$ on inputs of size $n \times n$.

1.4 Time-Space Lower Bounds

As our last application for this lecture, we consider time-space trade-offs for Turing machines. concretely, we have a single *read-only* tape, and O(1) read/write tapes. Given $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we say that such a Turing machine M computes f if $f(x, y) = M(x0^n y)$ for all input pairs. Our goal is to prove the following. The proof can also be found in [9].

Theorem 7. *Let f be as above and let M compute f. Then*

$$\mathsf{P}^{\mathsf{CC}}(f) \le \frac{T(M, n) \cdot S(M, n)}{n}$$

Where $T(M, n), S(M, n)$ are running time and space bounds respectively on M on inputs of length n.

In particular, if $\mathsf{P}^{\mathsf{CC}}(f) = \Omega(n)$ then $T(M, n) \cdot S(M, n) = \Omega(n^2)$.

Proof. The proof idea is to simulate $M(x0^n y)$ using a communication protocol. We start with Alice simulating M on $(x0^n y)$ as long as the input read only tape is on $x0^n$, when(and if) the tape moves to y, Alice stops the simulation, and sends Bob the entire content of the read/write tapes, and Bob continues the simulation as long as the input tape is on $0^n y$, and similarly, if the tape moves back to x, Bob sends the entire content of the work tapes to Alice.

All in all, since 0^n has length n, at least n steps of computation are executed after every exchange of the contents of the work tapes, thus the total number of rounds of communication is $\frac{T(M,n)}{n}$. Each round contains at most $S(M,n)$ bits as this is the bound on the size of the work tapes. Thus in total Alice and Bob can simulate the computation of M on $x0^n y$ using $\frac{T(M,n)\cdot S(M,n)}{n}$ bits, thus establishing the theorem. $\square$

References

[1] N. Alon, Y. Matias, and M. Szegedy, *The space complexity of approximating the frequency moments*, J. Comput. Syst. Sci., 58 (1999), pp. 137–147. 1

[2] A. Belovs and E. Blais, *A polynomial lower bound for testing monotonicity*, SIAM J. Comput., 50 (2021). 4

[3] E. Blais, J. Brody, and K. Matulef, *Property testing lower bounds via communication complexity*, Comput. Complex., 21 (2012), pp. 311–358. 3

[4] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA, H. Ortiz, ed., ACM, 1990, pp. 73–83. 3

[5] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky, *Improved testing algorithms for monotonicity*, in Randomization, Approximation, and Combinatorial Algorithms and Techniques, Third International Workshop on Randomization and Approximation Techniques in Computer Science, and Second International Workshop on Approximation Algorithms for Combinatorial Optimization Problems RANDOM-APPROX'99, Berkeley, CA, USA, August 8-11, 1999, Proceedings, D. S. Hochbaum, K. Jansen, J. D. P. Rolim, and A. Sinclair, eds., vol. 1671 of Lecture Notes in Computer Science, Springer, 1999, pp. 97–108. 3

[6] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky, *Testing monotonicity*, Comb., 20 (2000), pp. 301–337. 3

[7] M. Göös and A. Rubinstein, *Near-optimal communication lower bounds for approximate nash equilibria*, in 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018, M. Thorup, ed., IEEE Computer Society, 2018, pp. 397–403. 5

[8] S. Khot, D. Minzer, and M. Safra, *On monotonicity testing and boolean isoperimetric-type theorems*, SIAM J. Comput., 47 (2018), pp. 2238–2276. 4

[9] E. Kushilevitz and N. Nisan, *Communication complexity*, Cambridge University Press, 1997. 5

[10] T. Roughgarden, *Communication complexity (for algorithm designers)*, Found. Trends Theor. Comput. Sci., 11 (2016), pp. 217–404. 1