CS 354: Unfulfilled Algorithmic Fantasies April 8, 2019

Lecture 3: Communication Complexity

Lecturer: Aviad Rubinstein *Scribe: Nick Guo*

1 After thoughts about Query Complexity

1.1 Recap from last week

Last week we discussed Query complexity, also referred to as decision tree complexity or "blackbox" complexity. What are some pros and cons of studying query complexity?

Pro: easy hardness Because of the concrete restrictive model for accessing the input, it's relatively easy to prove hardness for query complexity.

Con: specific query model may be too restrictive Because our lower bounds hold for a very specific method of accessing the input, it may fail to capture other very plausible queries. Examples include:

Sorting The $\Omega(n \log n)$ lower bound only holds for comparison-based access. If we're in a different setting that allows for more than just direct comparisons (e.g. i-th digit queries), then we can do better (e.g. radix sort).

Submodular optimization The Symmetry Gap Theorem that we proved last week holds only for *value queries*: given a set S, return $f(S)$. In economic applications, another appealing model is *demand queries*: given a vector of prices (p_i for each $i \in [n]$), return $\max_S \{f(S) - \sum_{i \in S} p_i\}$.

Pro: lifting query lower bounds In this and subsequent lectures, we will show that query complexity lower bounds may sometimes be "lifted" to other, more interesting, models.

2 Communication Complexity

There are two parties are communicating, Alice and Bob, and they want to work together to compute the solution to some problem. Alice has input $x \in \{0,1\}^n$, and Bob has $y \in \{0,1\}^n$. There are multiple rounds of communication between Alice and Bob where each is sending some number of bits.

Definition 2.1. (Communication Complexity, CC). The communication complexity of a problem is the minimum number of bits that Alice and Bob need to exchange to compute a solution in the worst case.

Example 1. (Set Disjointness Problem) Alice and Bob each have $x, y \subseteq [n]$, and want to compute whether or not $x \cap y = \emptyset$. This problem is very hard, and has a lower bound CC of $\Theta(n)$ even for randomized or non-deterministic protocols.

2.1 Applications of Communication Complexity

1) **Distributed Computing**. We can use communication complexity to prove lower bounds on how many bits one component of our system needs to send to another.

2) **Circuit Complexity**. Say that we have a function and we want to compute how many logical gates (ANDs, ORs, NOTs) are needed to compute that function. This has a nontrivial connection to communication complexity (e.g. [5]).

3) **Streaming Algorithms**. A streaming algorithm is not allowed to see the entirety of the input all at once due to memory constraints. Instead, it must process a piece of the input, and then discard that in order to process the next piece. Thus, we can model it as a "one-way" communication between the algorithm's computation on past parts of the stream and the future parts.

4) **Game Theory**. We will focus mostly on game theory for the rest of this lecture. Communication complexity is a very natural application to game theory since we inherently are already dealing with interactions between different agents. Communication complexity lower bounds hold for very general models of each agent, so they demonstrate that an agent answering some function query cannot do better than the established communication complexity lower bound.

3 Nash Equilibrium

Alice and Bob are engaged in a game where each player chooses among N available actions. They both choose one of their available actions without knowing what the other has chosen, and then the choices made are revealed at which point they each receive some reward according to their respective payoff matrix. The payoff matrix is an N by N matrix of reward values corresponding to each Alice-Bob action pair. In a zero-sum game, Alice's reward matrix A has entries that are the negation of Bob's reward matrix B (that is, $A_{ij} = -B_{ij}$), but this relation doesn't necessarily have to hold if the game is not zero-sum. For simplicity, we will assume that Alice and Bob have the same number of available actions.

A pure strategy is one where the player always chooses the same action. A mixed strategy is some probability distribution over all of the available actions. The *support* of a strategy is the set of pure strategies used to construct it.

In plain words, Alice and Bob are playing in Nash equilibrium if neither has incentive to deviate from their current strategies.

Definition 3.1. (Nash Equilibrium) Let A be Alice's payoff matrix. Let B be Bob's payoff matrix. Let x, y be probability distributions over the actions available to Alice and Bob respectively. Then the strategies x, y are in Nash equilibrium (NE) if

1. for every strategy e_a available to Alice, $x^T Ay \geq e_a^T Ay$
2. for every strategy e_b available to Bob, $x^T By \geq x^T Be_b$

Example 2. (Rock Paper Scissors) For this game of Rock Paper Scissors, we say that a player receives a reward of 1 for winning, -1 for losing, and 0 for tying. Then, if Alice is the row player and Bob is the column player, their payoff matrices are

Alice	R	P	S
R	0	-1	1
P	1	0	-1
S	-1	1	0

Bob	R	P	S
R	0	1	-1
P	-1	0	1
S	1	-1	0

The NE for this game is $[1/3, 1/3, 1/3]$ for both Alice and Bob.

Definition 3.2. (ϵ Nash Equilibrium, ϵ-NE) Nash equilibrium is hard to compute, so we will work with ϵ-NE. Similar to before with NE, we say that the strategies x, y are in ϵ-NE if

1. for every strategy e_a available to Alice, $x^T Ay \geq e_a^T Ay - \epsilon$
2. for every strategy e_b available to Bob, $x^T By \geq x^T Be_b - \epsilon$

3.1 Motivating Questions

Do players reach an equilibrium? If no equilibrium is possible, can we prove it?

Definition 3.3. (Uncoupled Dynamics) Players play the game many times, but each player's strategy depends only on their own utility (payoff) matrix and history of actions for both themselves and their opponent. A player knows the entirety of their utility matrix, and does not have access to the other player's utility matrix.

We may try to ambitiously prove the following:
Wishful Statement: There are no uncoupled dynamics that converge to NE.

Inconveniently, brute-force search dynamics will trivially converge to a Nash equilibrium. Instead we will aim to prove the following:
Revised Goal: There are no FAST uncoupled dynamics that converge to NE.

Remark. The desideratum that the dynamics are "fast" is a surrogate to the elusive desideratum of "natural" dynamics.

Observation 3.4. *The communication complexity (CC) of finding a NE problem (almost) exactly captures the number of rounds of uncoupled dynamics:*

$$\frac{CC}{\log N} \leq \#\ of\ rounds \leq CC.$$

Proof sketch. First inequality: Given uncoupled dynamics, we can obtain a CC protocol by converting Dynamics-Alice action $i \in [N]$ into $\log(N)$ bits sent by CC-Alice, and same for Bob.
Second inequality: Given a CC protocol, we can design uncoupled dynamics where Alice and Bob only use the first two actions. Dynamics-Alice plays action 0 whenever CC-Alice sends 0 in the protocol, and same for Bob.[1] □

[1] We can prove a slightly stronger inequality, namely: # of rounds $\leq \frac{CC}{\log \log N}$. (Good exercise!)

How fast is FAST?

- Naive Benchmark (upper bound): $O(n^2)$. Alice sends Bob her entire payoff matrix.
- Optimistic Benchmark (lower bound): The complexity of writing down the ϵ-NE.

Theorem 3.5. (Carathéodory's Theorem). *If v is a convex combination of n points in d dimensions, then it's a convex combination of $d+1$ points.*

Theorem 3.6. (Approximate Carathéodory's Theorem). *If v is a convex combination of n points in d dimensions, then there exists v' such that for any $p \geq 2$, $||v - v'||_p < \epsilon$ where v' is a convex combination of $O(p/\epsilon^2)$ points* [2].

Theorem 3.7 ([6, 2])**.** *There exists an ϵ-NE $(\hat{x}, \hat{y})$ with support size $O(\frac{\log N}{\epsilon^2})$.*

Proof. Let (x, y) be an exact Nash equilibrium for a game where Alice has utility matrix A and Bob has utility matrix B, each of size $N \times N$. Then, $v = Ay$ is the vector of payoffs that Alice faces.

Observe that v is a convex combination of columns of A. Hence, we can apply the approximate Carathéodory theorem to see that there exists a convex combination of $O(p/\epsilon^2)$ columns of A forming v' such that $||v' - v||_p < \epsilon$. Essentially, there exists some y' such that $v' = Ay'$, and since v' is a sparse combination of columns, y' is sparse. This y' is some mixed strategy for Bob, and it achieves support size $O(p/\epsilon^2)$.
Which norm p should we use? One hint is that our proof would go through if we choose $p = \Theta(\log N)$. As we will soon see, we actually want $p = \infty$ norm. Fortunately, we have that for any vector $w \in [-1, 1]^N$,

$$||w||_\infty \approx ||w||_{\log(N)}.$$

(This is a generally very useful observation!)

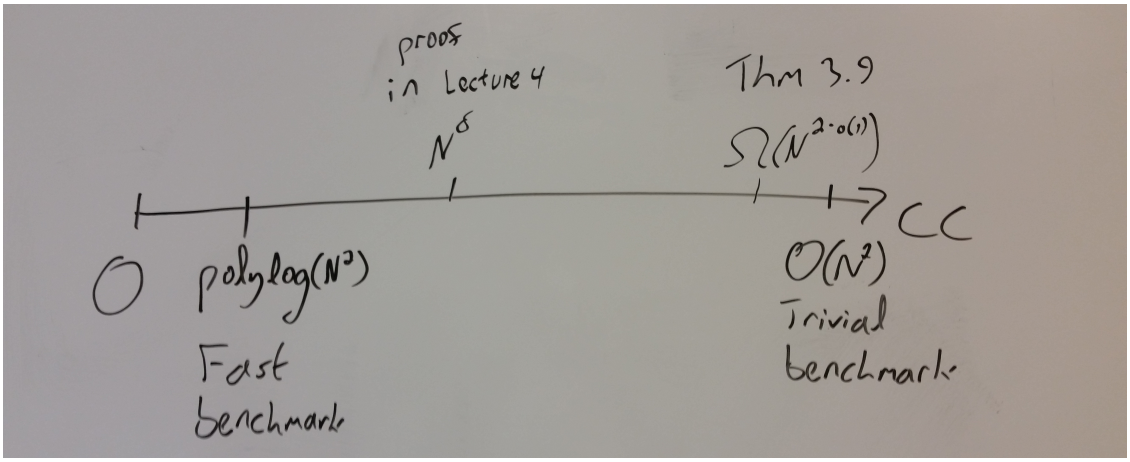
$$\begin{aligned}
||w||_\infty \leq ||w||_{\log(N)} &= (\sum_i w_i^{\log(N)})^{1/\log(N)} \\
&\leq (\sum_i \max_j w_j^{\log(N)})^{1/\log(N)} \\
&= (N \max_j w_j^{\log(N)})^{1/\log(N)} \\
&= N^{1/\log(N)} \max_j |w_j| \\
&= 2||w||_\infty.
\end{aligned}$$

Thus, for any strategy e_i that Alice chooses, $e_i^T Ay \approx e_i^T Ay' \pm \epsilon$. Her payoff remains approximately the same as when Bob was playing the NE strategy. Repeating the above process starting with Bob's vector of payoffs yields an ϵ-NE strategy for Alice. □

Corollary 3.8. *The non-deterministic communication complexity of ϵ-NE is $O(\frac{\log^2 N}{\epsilon^2})$.*

However, what we really care about is the randomized communication complexity of ϵ-NE.

Theorem 3.9 ([1, 4])**.** *The communication complexity of ϵ-NE is $\Omega(N^{2-o(1)})$.*



4 Simulation/Lifting Theorem

The main idea behind this tool is that we are lifting the weaker query complexity to a higher communication complexity, or we are simulating a communication complexity protocol with a query complexity protocol.

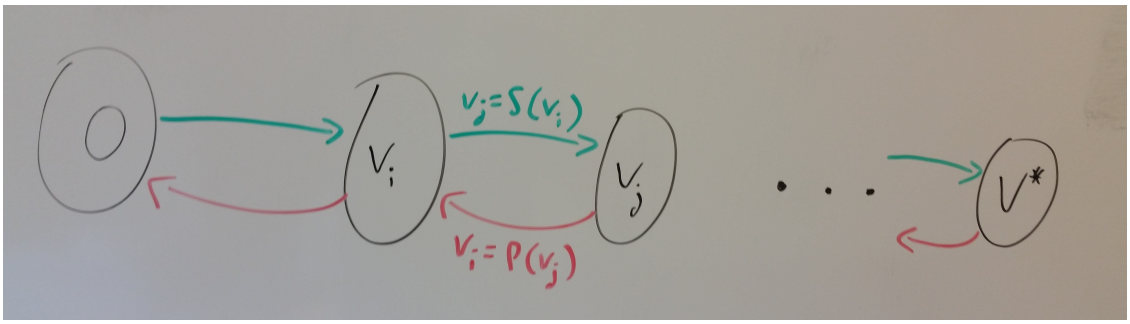
Theorem 4.1 ([3, 7], informal)**.** *There exists a way to partition any query problem (meaning that we give some of the inputs to Alice, and some to Bob) such that the communication complexity of the problem is at least the query complexity. $CC \geq QC$.*

To recap the above theorem, we can take any query problem—we don't care at all about what it looks like—and there is a blackbox way of partitioning the inputs so that the same lower bounds hold in the communication complexity model of the problem, which is a much easier model.

However, the partitioning of the inputs is really important because there are partitions that do not preserve the query complexity lower bounds.

For example, consider the following examples for how **not** to partition the inputs.

Reminder: End-of-the-Line Problem. Consider the same scenario as in the previous lecture where we have as input functions S and P for which a given vertex tells us the outgoing and incoming edge respectively.



- **Attempt 1:** Give S to Alice, and P to Bob. This is not what we want since Bob and Alice can each solve the problem on their own with zero communication.
- **Attempt 2:** Randomly color the edges blue and red, and give Alice all the red edges and Bob all the blue edges. This partition is flawed in that it allows the end of the line to be computed with just $\log(N)$ communication. Consider the following protocol:

  1. Label each vertex with an $\log(N)$-bit index, with the start vertex being index 0.
  2. Alice coordinate-wise XORs the endpoints of every edge that she has, and Bob does the same.
  3. Alice sends her XOR result to Bob who then XORs her result with his own, and this final result is the end of the line.

  Notice that each vertex on the line, except for the start and end vertices, belongs to two edges. Thus their indices cancel in the XOR. Bob's result is then the XOR of the start and end vertices, but since the start vertex has index 0, this is simply the index of the end vertex.

We cannot naively partition the inputs since the communication complexity may become too tiny. The next lecture will go into detail on how we can perform this blackbox partitioning to preserve query complexity lower bounds.

References

[1] Yakov Babichenko and Aviad Rubinstein. Communication complexity of approximate nash equilibria. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 878–889, 2017.

[2] Siddharth Barman. Approximating nash equilibria and dense subgraphs via an approximate version of carathéodory's theorem. *SIAM J. Comput.*, 47(3):960–981, 2018.

[3] Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 132–143, 2017.

[4] Mika Göös and Aviad 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*, pages 397–403, 2018.

[5] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 539–550, New York, NY, USA, 1988. ACM.

[6] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Proceedings 4th ACM Conference on Electronic Commerce (EC-2003), San Diego, California, USA, June 9-12, 2003*, pages 36–41, 2003.

[7] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999.