Data Streaming Lower Bounds from Communication Complexity

Purdue University
CS 590 SLA Lecture 19

Young-San Lin
March 25, Spring 2021

1 Preliminaries

We consider space lower bound for streaming problems by reductions from communication problems. Alice and Bob each has an input σ_1 and σ_2, respectively. We want to compute $f(\sigma_1, \sigma_2)$ with minimum number of bits communicated. Intuitively, if we have a streaming algorithm with space s bits, then there exists a communication protocol in which Alices sends the s bits to Bob, and Bob can compute $f(\sigma_1, \sigma_2)$ by viewing s and σ_2.

This implies that if we can show a reduction from a communication problem with a lower bound to a streaming problem, then we have a space lower bound for the streaming problem.

Suppose we have a function $f : X \times Y \to \{0, 1\}$, where Alice gets input $x \in X$ and Bob gets input $y \in Y$. Our goal is to output $f(x, y)$ with minimum amount of bits exchanged. Here, communication is expensive, while time and space does not matter. Clearly, the communication complexity is at least $\min\{\log |X|, \log |Y|\} + 1$ since one side can send the message of the entier X or Y so that the other side can compute the outcome of f.

There are two ways to model this setting. First, we can consider a $|X| \times |Y|$ 0-1 matrix where each entry is $f(x, y)$, Alice denotes the row, and Bob denotes the column. They cooperate and iteratively reduce the search space via communication until the submatrix consists of all zeros or ones, then return $f(x, y)$. Another way is to consider the decision tree model, where each level denotes Alice (Bob) sends a bit to Bob (Alice), and all the possible routes afterwards. At some point, they will reach a leaf which denotes $f(x, y)$. The height of the tree is the communication complexity.

Let the communication protocol be π and $out^\pi(x, y)$ denote the outcome of protocol π on x and y. We introduce some notations used later.

Definition 1.1. π computes f if $out^\pi(x, y) = f(x, y)$ for all $(x, y) \in X \times Y$.

Definition 1.2. The *cost* π is the number of bits communicated over the worst case input in $X \times Y$.

There are two types of protocols:

Deterministic: $D(f)$ denotes the minimum cost of a deterministic protocol[1].

Randomized:

Private coin: the randomization is based on each players private random bits.

Public coin: all players use a public string of random bits.

[1]We are taking the minimum among all protocols, where each protocol considers the worse case input with the maximum number of communication bits needed.

Definition 1.3. π computes f with error δ if for all $(x, y) \in X \times Y$,

$$\Pr_R[out^\pi(x, y, R) \neq f(x, y)] \leq \delta$$

where R denotes the public random bits.

Definition 1.4. $R_\delta(f)$ is the minimum cost of a δ-error protocol for f.

Claim 1.5. For all $\delta \in (0, 1/3)$, $R_\delta(f) = \Theta(R_{1/3}(f))$. We denote $R(f) := R_{1/3}(f)$.

Definition 1.6. $\vec{D}(f)$ ($\vec{R}(f)$) denotes the minimum cost of deterministic (randomized) one-way protocols, i.e., only Alice sends messages.

Definition 1.7. $D^k(f)$ ($R^k(f)$) denotes the minimum cost of deterministic (randomized) k-round protocols.

Here are some basic observations:

Claim 1.8.

1. $R(f) \leq D(f) \leq \min\{\log |X|, \log |Y|\}$.
2. $D(f) \leq D^k(f)$ for all $k > 0$.
3. $F(f) \leq R^k(f) \leq R^\ell(f)$ for all $k \leq \ell$.

2 Specific Communication Problems

- $EQ_N(x, y) = \begin{cases} 1 \text{ if } x = y, \\ 0 \text{ otherwise.} \end{cases}$

  Claim 2.1. $D(EQ_N) \geq N$ and $\vec{R}^{priv}(EQ_N) = O(\log N)$.

- $IND(x, i) = x_i$ where $x \in \{0, 1\}^N$ and $i \in [N]$. Alice gets x and Bob gets i.

  Claim 2.2. $\vec{D}(IND_N) \geq N$, $\vec{R}(IND_N) \geq \Omega(N)$, and $D^2(IND_N) \geq \log N$.[2]

- $DISJ_N(x, y) = \begin{cases} 1 \text{ if } x \cap y = \emptyset, \\ 0 \text{ otherwise.} \end{cases}$ Here, $x, y \in \{0, 1\}^N$. Alice gets x and Bob gets i. x and y can be regarded as vectors for the two sets where each index denotes if an element is in the set (with value 1) or not (with value 0).

  Claim 2.3. $R(DISJ_n) \geq \Omega(N)$.

We show the following claim.

[2]For $D^2(IND_N)$, Bob sends the index and Alice sends the bit of the index. This takes $\log N$ bits of communication.

Claim 2.4. $\vec{D}(IND_N) \geq N$.

Proof. Let $m(x)$ denote the message that Alice sends to Bob given x. Bob must recover x_j from $m(x)$ exactly no matter what j is. Alice has 2^N many messages x and Bob needs to recover every bit of x from $m(x)$. This implies that the mapping from x to $m(x)$ must be one-to-one and onto. There are $\sum_{i=1}^{N-1} 2^i = 2^N - 1$ messages of length smaller than N. So if there is a protocol of length at most $N-1$, the message will not be a bijection, a contradiction. □

Claim 2.5. $\vec{R}^{priv}(EQ_N) = O(\log N)$.

Proof. Alice and Bob can view their inpts as polynomials. We map x to $P_x(t) := \sum_{i=1}^{N} x_i t^i$ and y to $P_y(t) := \sum_{i=1}^{N} y_i t^i$, where $P_x, P_y \in \mathbb{F}$ where $\mathbb{F}$ is a field with size to be decided later. The protocol works as follows.

1. Alice picks a random element $r \in \mathbb{F}$ and send r and $P_x(r)$ to Bob.
2. Bob compares $P_y(r)$ with $P_x(r)$. If $P_y(r) = P_x(r)$, then output 1; otherwise, output 0.

The number of bits communicated by this protocal is the number of bits used to encode r and $P_x(r)$, which is $\log|\mathbb{F}| + \log|\mathbb{F}| = 2\log|\mathbb{F}|$. The remaining is to show that the protocol succeeds w.h.p..

When $x \neq y$, by the fundamental theorem of algebra, we have that

$$\Pr_r[P_x(r) = P_y(r)] \leq \frac{N}{|\mathbb{F}|}$$

since the number of roots cannot exceed the degree divided by the size of the field. Here, we can pick $|\mathbb{F}| = N^2$. □

3 Communication Matrix and Complexity Bounds

We recall the communication matrix and bring the connection to communication bounds.

Definition 3.1. $C \subseteq X \times Y$ is a combinatorial rectangle if $C = X' \times Y'$ where $X' \subseteq X$ and $Y' \subseteq Y$.

Definition 3.2. C is monochromatic w.r.t. f if $f(x,y) = f(x',y')$ for all $(x,y),(x',y') \in C$.

Theorem 3.3. Let M_f be the communication matrix and suppose $D(f) \leq \log(\text{rank}(M_f))$. Let ℓ be a leaf of the decision tree of the protocol π. Let $C_\ell = \{(x,y)\text{whose transcript ends at } \ell\}$. Then C_ℓ is a combinatorial rectangle, i.e., for all $(x,y),(x',y') \in C_\ell$, we have that $(x',y),(x,y') \in C_\ell$.

Claim 3.4. $D(f)$ is NP-hard to compute.

Conjecture 3.5. $D(f) = O(\text{polylog}(\text{rank}(M_f)))$.

Theorem 3.6 (Lovett)**.** $D(f) = O(\sqrt{\text{rank}(M_f)}\log(\text{rank}(M_f)))$.

4 From Communication Complexity to Streaming

We now use communication complexity to show streaming space lower bounds.

Theorem 4.1. Any single-pass deterministic (randomized) streaming algorithm that decides if an n-vertex graph $G = (V, E)$ has a perfect matching requires $\Omega(n^2)$ space.

Proof. We reduce from the INDEX problem and recall that $\vec{D}(IND_N) = \Omega(N)$, $\vec{R}(IND_N) = \Omega(N)$. Let $N = n^2$ and denote each index of x as (i, j) where $i, j \in [n]$. Alice gets the entire matrix x and Bob gets (i, j).

Given the matrix x and (i, j), let $x_{pq} \in \{0, 1\}$ denote the value at index (p, q), we construct the graph G as follows.

1. Create vertices p and q' for $p, q \in [n]$.
2. Create edges (p, q') iff $x_{pq} = 1$.
3. Create vertices ℓ_p and r_q for $p \in [n] \setminus \{i\}$ and $q \in [n] \setminus \{j\}$.
4. Create edges (ℓ_p, p) and (q', r_q).

We can observe that $x_{ij} = 1$ iff G has a perfect matching.

The (p, q) edges are Alice's edges and the other edges are Bob's edges. We stream Alice's edges before Bob's edges and this simulates the communication problem. This implies that if we can solve in $o(n^2)$ space, then we can solve IND_{n^2} with $o(n^2)$ bits communicated. □

Theorem 4.2. Any single-pass streaming approximation algorithm for shorest paths with approximation ratio at most 5/3 uses $\Omega(n^2)$ space.

Proof. We reduce from IND_{n^2}. We use the same square matrix notion for x as before. Given x and (i, j), we construct the graph G as follows.

1. Create vertices p and q' for $p, q \in [n]$.
2. Create edges (p, q') iff $x_{pq} = 1$.
3. Create vertices s and t.
4. Create edges (s, i) and (j', t).

We observe that if $x_{ij} = 1$, then $d(s, t) = 3$; otherwise $d(s, t) \leq 5$.

This implies that if we have a 5/3-approximate streaming algorithm with $o(n^2)$ space, then we can use $o(n^2)$ space to solve the INDEX problem. □