# Lecture 4. Multiparty Communication Complexity

## Models

In this class we study communication complexity models in which input variables are distributed to k>2 parties. Two most studied ways of distributing input variables are

* *Number in Hand (NiH)*: The input is (x1, …, xk) where for some set , and is given to Player i.
* *Number on the Forehead (NoF)*: The input is (x1, …, xk) where for some set , and is given to Player i.

Typical ways of communication models include:

* *Blackboard*: There is a common blackboard for players to write on, and it’s visible to everyone. The communication complexity is the total number of bits written on the blackboard.
* *Message passing*: There are channels between every pair of players, and the communication complexity is the total number of transmitted bits during the whole protocol.
* *One-way*: Player 1 sends a message to Player 2, who then sends a message to Player 3, and so on. The last player outputs a value.
* *SMP*: One can further restrict the allowed communication channels: each of the k players sends one message to a Referee who then announces the result.

The NiH model draws much attention in recent years because of its connection to *data streaming algorithms*. NiH with SMP has an intimate connection to the *data sketching algorithms*. Earlier studies in the multiparty communication complexity focused more on the NoF model with a blackboard because of its connection to *circuit complexity*. (We will come to the connection later in the class.)

Across today’s lecture, we assume that there is a blackboard. It is easy to see that some of the results hold for other communication models as well.

## Number in Hand, with a blackboard.

### 2.1 From rectangle to tensor

The counterpart of rectangle in NiH case is tensor. A tensor in is a set where . The following is a straightforward generalization of the rectangle property.

**Proposition 2.1**. Any c-bit deterministic protocol for a function f in NiH model partitions the set X at most into 2c monochromatic tensors.

We sometimes consider *partial functions*, which are functions with domain not the entire X but only a subset of it. Equivalently, we can imagine that there is a promise that the input comes from a fixed subset of X. For inputs that fall outside the subset, we don’t care. A protocol is correct for this partial function if the protocol outputs the correct value on inputs coming from the subset; the protocol is allowed to output anything on other inputs. For partial functions, a tensor is *monochromatic* if it doesn’t contain both 1- and 0-inputs. Indeed, as long as a protocol reaches a monochromatic tensor, then it can output a value that is right for all inputs on which the function is defined.

### 2.2 Disjointness

Denote the input by (x1, …, xk) where each xi = xi1…xin is an n-bit string. Recall that Disjn in two-party case is defined as, in our current notation, Disjn(x1,x2) = 1 iff j s.t. x1j = x2j = 1. When generalized to k-party case, different requirements can be imposed: We can require i s.t. x1i = … = xki = 1, or only require the sets x1, …, xk are not pairwise disjoint. Let’s consider the partial function

.

**Theorem 2.2**.

Proof. We show the following two properties.

1. *There are (k+1)n 1-inputs*. The input is basically a partition of [n] into k+1 parts, with the i-th part given to player i and the last part unassigned to any player.
2. *Any monochromatic tensor without a 0-input contains at most kn 1-inputs*. Fix a monochromatic tensor. For any element , there is a player’s input xi not containing it. The size the tensor is at most the number of assignments of elements in [n] to [k].

Then, there should be at least monochromatic tensors. Therefore, the theorem follows from the previous proposition. □

### 2.3 Computing frequency moments in data stream

Suppose that there is a stream of data . We hope to compute some statistical value of the data using a small amount, ideally bits, of *space*. The requirement for the algorithms is that we only see the data passing once in front of us. So we can process x1 in whatever way we like, but then it’s gone and never comes back, and we see x2 and process x2, and so on. After formally initialized in a paper [AMS99], the model has been rapidly developed in the last decade or so.

To see why sometimes techniques can be used to achieve low space complexity, let’s consider some interesting puzzles first.

**Example**. Suppose m = n-1, and all xi’s are different. Then there is exactly one number in [n] that is missing in the data. They want to find out the missing number. What’s the lowest space your algorithm can achieve?

**Example**. What if m = n-2 and they want to find the two missing numbers?

From the above examples, you can see that algorithms can be crafty, which also raises the issue of proving lower bounds. Communication complexity is one powerful tool to prove lower bounds for space complexity of streaming algorithms. Here we illustrate the main idea using one classic example.

In the data , suppose the number j appears rj times. Define the d-th frequency moment of x to be In particular, f0(x) is the number of distinct elements in x, and f1(x) = m. For , fd(x) gives useful statistical information about the string.

**Theorem 2.3**. For *,*

Proof. The idea is, as always, design a protocol to simulate an algorithm. Suppose that there is an algorithm A computing fd using only c bits of space. Let . We want to solve the Disjn,k problem on input (y1, …, yk). Note that each yi is a subset of [n]. Player 1 gives the elements in y1 to A as a stream of data items and runs A. When A finishes reading these elements, Player 1 pass her space to Player 2 who takes over to continue to run A upon seeing elements in y2, and so on. Since A always uses c bits of space, the communication is at most (k-1)c bits. Since the items yi come in a stream for A, we can use a one-way protocol to simulate it. (Since blackboard model is stronger, we can of course also use the blackboard protocol to simulate this.)

Now we say that computing the d-th moment can solve the Disjn,k problem. Indeed,

* If (y1, …, yk) are pairwise disjoint, then
* If there is a common element in y1, …, yk, then

Therefore,

c

which gives □

## Number on the Forehead

We are back to total functions, i.e. the domain of f is the whole .

### 3.1 Cylinder intersection

Let’s first see what the analog of monochromatic tensors is in the NoF model. A subset C of is a *cylinder* in the i-th dimension if for some set . Namely, the membership of C doesn’t depend on the i-th coordinate. A subset C of X is a *cylinder intersection* if , where each is a cylinder in dimension i. The following theorem is an analog of the monochromatic rectangle/tensor decomposition in previously studied models.

**Theorem 3.1**. Any c-bit deterministic protocol for a function f in NoF model partitions the set X into at most 2c monochromatic cylinder intersections.

### 3.2 Discrepancy and binary cube bound

We’ve seen discrepancy in two-party randomized communication complexity. We can surely extend it to multiparty as well. For the sake of simplicity, let’s consider the uniform case. For any Boolean function and any subset S of X, its discrepancy is

.

Let’s then define

,

where the maximum is taken over all cylinder intersections S.

**Theorem 3.2**. .

Proof. For any monochromatic cylinder intersections C, we have disc(f,C) = |C|/|X|. Therefore

,

and thus we need at least 1/disc(f) many monochromatic cylinder intersections to be able to cover the entire X. Applying Theorem 3.1 gives the claimed bound. □

The discrepancy bound is not easy to use, since one need to argue for all cylinder intersections. An easier bound is the binary cube bound. For any , define a binary cube. Note that here *ai* and *bi* may be the same, so D is a multi-set. Define . Choose *ai* and *bi* uniformly at random from Xi, and define . The following theorem was firstly given by Chung [Chu90]; see [Raz00] for a simplified proof.

**Theorem 3.3**. , thus .

### 3.3 A specific function: GIP

The *Generalized Inner Product* (GIP) function is defined as follows.

*.*

Namely, the input x is an matrix, and each of the k players gets all but one columns.

**Theorem 3.4**. .

Proof. We’ll show , which, combined with Theorem 3.3, gives the claimed bound. Since the binary cube bound works for the {+1,-1} range, let’s first change f to . Note that the change doesn’t affect the communication complexity. We need to compute where .

where we used the indicator function **1**[]. Now we take the expectation of D. Consider to pick *ai*’s first, then pick *bi*’s.

,

as desired. □

The bound deteriorates exponentially with k, but this is inevitable in view of the following fact, proven by Grolmusz [Gro94]. (We’ll omit the proof here.)

**Fact**. D(GIPn,k) = O(kn/2k).

## References

[AMS99] N. Alon, Y.Matias, and M. Szegedy. **The space complexity of approximating the frequency moments**, *J. Comput. Syst. Sci.* 58 (1), 137–147, 1999.

[Chu90] Fan R. K. Chung: **Quasi-random classes of hypergraphs**, Rand. Struct. Algorithms 1, 363–382, 1990.

[Raz00] Ran Raz. **The BNS-Chung criterion for multi-party communication complexity**, *Computational Complexity* 9, 113-122, 2000.

[Gro94] V. Grolmusz. **The BNS lower bound for multi-party protocols is nearly optimal**, *Information and Computation* 112(1), 51-54, 1994.