CS 2429 - Comm. Complexity: Applications and New Directions

Distributed Computing and Communication Complexity

Lecturer: Trevor Brown

In these notes, we study the intersection of communication complexity and distributed computing. To understand why distributed computing researchers care about communication complexity tools and results, we briefly turn our attention to distributed computing. In traditional distributed computing models, local computation is free, and communication between parties is expensive. Typical complexity measures include the number of messages sent (*message complexity*), the total number of bits sent (*bit complexity*), and the total number of rounds of computation in synchronous models (*round complexity*). Communication complexity is interested in these same complexity measures, and the most common communication complexity models are very similar (in some cases identical) to distributed computing models. In part, this is because communication complexity emerged from the study of distributed computing, and early interest in its models was driven by applications in distributed computing. For example, number-on-forehead models, wherein each player can see only the inputs of other players, were initially considered unrealistic (and less interesting) because they did not coincide with distributed computing models. In contrast, number-in-hand models, wherein each player sees only her own input, can be applied immediately to distributed computing problems, and have been studied extensively.

1 k-party communication complexity models

1.1 Blackboard model

In the blackboard model, players communicate by writing on a shared blackboard. Players take turns writing a bit, and the protocol decides who goes next. Each player sees everything written to the blackboard. When there are k players, each bit written to the blackboard is broadcasted to every player, so this model is very powerful. (In a message passing model, a player might have to send k messages to simulate writing a single bit in the blackboard model.)

1.2 Point-to-point channel model

A communication graph determines which players can communicate. Each player has a direct communication channel with each neighbour in the communication graph. For simplicity, the communication graph is usually assumed to be fully connected (meaning each player p_i can communicate directly with each player p_j). Figure 1 depicts a 4-party instance of this model.

Synchronous version. The computation is divided into *rounds*. In each round, each player sends a message and receives all incoming messages. This model is the same as the *synchronous message passing* model in distributed computing.

Asynchronous version. In this model, there are no rounds of communication. Each player has an inbox that behaves like a FIFO queue. Messages can be delayed for an arbitrarily long time,

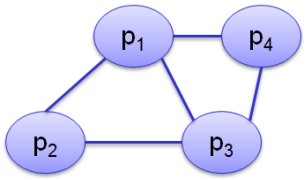

Figure 1: 4-party instance of the point-to-point channel model

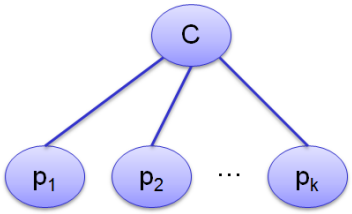

Figure 2: k-party instance of the coordinator model

but any message sent will eventually arrive in its recipient's inbox. This model is the same as the *asynchronous message passing* model in distributed computing.

1.3 Coordinator model

The coordinator model is a message passing model in which k players can communicate only with a special coordinator player C ($k+1$ parties in total). This is a special case of the point-to-point channel model, where the communication graph contains edges only from C to each of the k players. See Figure 2 for a depiction of the k-party coordinator model.

Synchronous version. In each round, every player sends a message to C, and C sends a (different) message to each player.

Asynchronous version. One can still think of the computation as being divided into rounds, but the length of time to complete each round can be different. C acts as a scheduler. A *round* consists of C sending a message to some player p_i, and p_i responding.

See [3] for a more detailed description of the (a)synchronous point-to-point channel and coordinator models, and for results on the communication overhead of having these models simulate one another.

2 Symmetrization

In 2012, Philips, Verbin and Zhang [4] introduced **symmetrization**, which is a technique for obtaining k-party lower bounds from 2-party lower bounds. The high level idea is to perform a reduction that computes a 2-party function $f_2(x, y)$ if we are given a protocol to compute a k-party function $f_k(I_1, I_2, ..., I_k)$. The goal of the symmetrization technique is to show that computing f_k requires roughly k times as much communication as computing f_2. In this section, we focus on the coordinator model with public coin (shared) randomness.

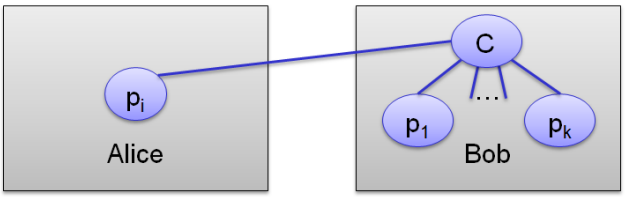

Figure 3: 2-party simulation of a k-party protocol in the coordinator model

2.1 The technique

Given a protocol P_k that computes f_k, we design a protocol P_2 to compute f_2. First, we find a hard input distribution for f_2. Then, Alice and Bob use inputs drawn from this hard distribution to construct **symmetric** inputs $I_1, I_2, ..., I_k$ for f_k. In this context, a set of k inputs drawn from a distribution μ is symmetric if players' inputs can be interchanged without changing the input distribution. Alice and Bob use the inputs they construct to simulate an execution of P_k, and use the output of P_k to solve for f_2.

Simulating P_k. Alice uniformly chooses a random player p_i to simulate. Since randomness is shared, Bob knows which player Alice will simulate. Bob simulates the other $k-1$ players and coordinator C. This scenario is depicted in Figure 3. Any time p_i sends a message to C, Alice sends a message to Bob. Any time C sends a message to p_j, Bob sends a message to Alice. All other messages sent in P_k are internal to Bob, so he can simulate the transmission and receipt of the message entirely in his internal state (requiring no communication).

Obtaining a lower bound. We use $Cost(P_2)$ to denote the communication used by P_2 on input (x, y). Similarly, $Cost(P_k)$ denotes the communication used by P_k on input $(I_1, I_2, ..., I_k)$. Since inputs are symmetric, in expectation, P_2 sends at most a $1/k$ fraction of the bits sent by P_k. To see why, note that, when Alice uniformly randomly chooses her player p_i to simulate, she is effectively choosing an edge (p_i, C). This edge is the only edge out of k edges in the communication graph for protocol P_k that causes communication between Alice and Bob in P_2. Alice is equally likely to choose any edge between the coordinator and a player so, in expectation, the communication along the edge she chooses can be at most a $1/k$ fraction of the total communication over all k edges. Therefore, $E[Cost(P_2)] \leq (1/k)Cost(P_k)$. Equivalently,

$$Cost(P_k) \geq kE[Cost(P_2)].$$

2.2 Preliminaries

Distributional communication complexity D_ϵ^μ with error ϵ and input distribution μ is defined as:

$$D_\epsilon^\mu = min_P \; max_x \; cost(P(x))\mu(x),$$

where $cost(P(x))$ is the communication needed to run protocol P with input x. In other words, D_ϵ^μ worst-case communication performed by the *best protocol*, which is the protocol that does the least communication on its worst input.

Since the k-party communication is k times the *expected* communication for the 2-party protocol, we need some notion of expected communication complexity. *Expected distributional communica-*

tion complexity $E[D^{\mu}_{\epsilon}]$ (a slight abuse of notation) with error ϵ and distribution μ is:

$$E[D^{\mu}_{\epsilon}] = min_P \; E_x[cost(P(x))\mu(x)].$$

Expected distributional communication complexity is the same as distributional communication complexity, except that the *expected value* over all inputs x is taken instead of the *maximum* over x. It follows that $D^{\mu}_{\epsilon} \geq E[D^{\mu}_{\epsilon}]$. Then, from Yao's minimax theorem we obtain the following.

$$R_{\epsilon} \geq D^{\mu}_{\epsilon} \geq E[D^{\mu}_{\epsilon}]$$

We will use a simple information theoretic lemma from [4] to obtain lower bounds on the following two sections.

Lemma 1 *Suppose Alice and Bob have uniform random n-bit inputs. For Alice to learn Bob's input with probability* $(1-\epsilon)$*, Bob must send* $(1-\epsilon)n$ *bits, in expectation.*

2.3 Easy lower bound: k-XOR w/coordinator

In the *coordinate-wise* k-XOR problem, k players have inputs $I_1, I_2, ..., I_k$, each an n-bit string. The output is an n-bit string, whose i-th bit is equal to the XOR of the ith bits of $I_1, I_2, ..., I_k$. Let P_k be a protocol that computes k-XOR. Our goal is to use P_k to build a protocol P_2 that solves 2-XOR. We can then plug in a lower bound for 2-XOR to get a lower bound on the amount of communication performed by P_k.

Constructing symmetric inputs for P_k. We start by choosing an input distribution for the 2-party problem that is sufficiently hard to give us a good lower bound on 2-XOR. It turns out that uniform random inputs are hard enough to get an $\Omega(n)$ lower bound for 2-XOR. So, Alice uniformly randomly chooses a player p_i, and sets its input I_i to be her input x. Bob uses his input y to construct inputs for the rest of the players $p_j \neq p_i$. (Recall that the randomness is shared, so Bob can tell which player p_i was chosen by Alice.) Bob constructs his players' inputs as follows.

$$\{I_j \mid j \neq i\} = \text{a set of independent uniformly random bit strings}$$

$$\text{subject to constraint: } \text{XOR}(I_1, I_2, ..., I_{i-1}, I_{i+1}, ..., I_k) = y.$$

Since $I_i = x$ and $\text{XOR}(I_1, I_2, ..., I_{i-1}, I_{i+1}, ..., I_k) = y$, the output of P_k is exactly 2-XOR(x, y).

To see why the inputs to P_k are symmetric, consider a set of inputs $I_1, I_2, ..., I_k$ created by Alice and Bob. Relabel the players, permuting their inputs. The player that now gets Alice's input x is equally likely to have been chosen by Alice to receive input x. The rest of the inputs are uniform and independent, by Bob's construction. Therefore, permuting the players' inputs does not change the input distribution.

Simulating P_k. Alice simulates p_i and Bob simulates every other player $p_j \neq p_i$, as well as the coordinator C. Whenever p_i sends a message to C in the simulated protocol P_k, Alice sends a message to Bob, and vice versa. Every other message is internal to Bob, and its transmission and receipt can be simulated entirely within Bob's internal state.

Lower bound on communication of P_k. Since inputs are symmetric, the expected communication between p_i and the coordinator is at most a $1/k$ fraction of the communication performed

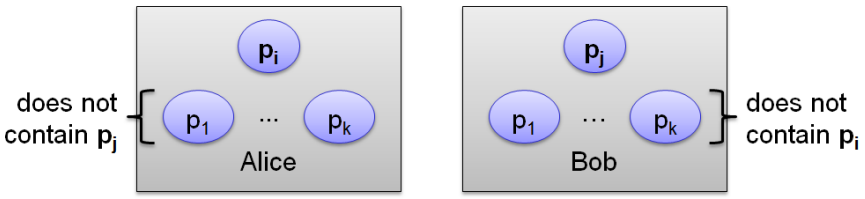

Figure 4: 2-party simulation of a k-party protocol in the blackboard model

by P_k. (To see why, recall: Alice uniformly randomly chooses one of k edges in the communication graph when she chooses p_i, and this is the only edge that causes messages between Alice and Bob in the 2-party protocol.) So, $Cost(P_2) \leq (1/k)E[Cost(P_k)]$. Equivalently, $Cost(P_k) \geq kE[Cost(P_2)]$.

We now compute a simple lower bound on $E[Cost(P_2)]$. Given XOR(x, y), Alice can recover y by computing XOR$(x$,XOR$(x, y)) = y$. So, Bob can communicate y to Alice in $E[Cost(P_2)]$ bits. By the information theoretic lemma, $E[Cost(P_2)] \geq n$. So, we have $Cost(P_k) \geq nk$.

This lower bound was for a protocol P_k that does not make any errors. However, if P_k can err on an ϵ fraction of inputs, then it is easy to see that the same reduction solves 2-XOR with error bound ϵ. Then, the information theoretic lemma says that, in expectation, at least $(1 - \epsilon)n$ bits must be sent to communicate Bob's input to Alice with error ϵ, so the lower bound becomes $Cost(P_k) \geq (1 - \epsilon)nk$.

2.4 Easy lower bound: k-XOR w/blackboard

We now consider the same problem, but in the blackboard model. The blackboard model changes the way we construct inputs to P_k and simulate its execution.

Setting up the simulation of p_k. Alice and Bob uniformly pick players p_i and p_j $(i \neq j)$, respectively, using shared randomness. Alice simulates every player except for p_j. Bob simulates every player except for p_i. (So, $\{p_1, ..., p_k\} \setminus \{p_i, p_j\}$ are simulated by both Alice and Bob.) This scenario is depicted in Figure 4.

Constructing symmetric inputs for p_k. As in the coordinator model, uniform random inputs for 2-XOR are sufficiently hard to yield the lower bound we want for k-XOR. Alice sets $I_i = x$. Bob sets $I_j = y$. Alice and Bob each set I_m $(m \neq i, j)$ to a uniformly random bit string (using shared randomness, so they both choose the same bit string).

It is straightforward to see that the inputs $I_1, ..., I_k$ are symmetric, since x and y are uniform independent, and the inputs in $\{I_m \mid m \neq i, j\}$ are uniform independent.

Running the simulation of p_k. Every time p_i writes to the board in the k-party protocol, each of the other $k - 1$ parties must see what it wrote. However, since Bob does not simulate p_i, he cannot see what p_i writes unless Alice also writes to the board in the 2-party protocol. So, each time p_i writes, Alice must write. Similarly, every time p_j writes, Bob must write. Every time p_m $(m \neq i, j)$ writes, no communication is needed, since Alice and Bob are both simulating p_m (and both know p_m's input).

Communication complexity of P_k. Since the inputs are symmetric and Alice and Bob write to the blackboard only when two out of k players (chosen uniformly) write in P_k, the expected communication for P_2 is at most a $2/k$ fraction of the communication for P_k. So, $E[Cost(P_2)] \leq$

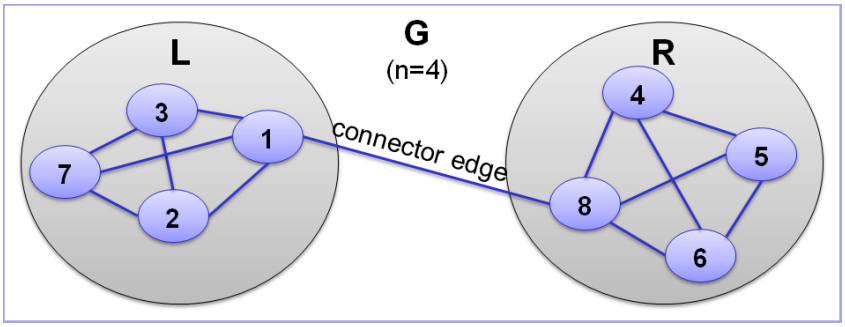

Figure 5: Example of the type of graph created by the reduction to k-CONN.

$(2/k)Cost(P_k)$. Equivalently, $Cost(P_k) \geq (k/2)E[Cost(P_2)]$. By the information theoretic lemma, $E[Cost(P_2)] = n$. So, $Cost(P_k) \geq nk/2$.

2.5 Harder lower bound: graph connectivity in the coordinator model

k-party graph connectivity (k-CONN). The edges of an $2n$-node graph G are partitioned amongst k players (so no one knows all of G). Each input I_j is a set of edges. The output of k-CONN is 1 if G is connected, and 0 otherwise. The lower bound assumes $k \geq 100 \log n$, which is reasonable for a large distributed system.
2-party disjointness (2-DISJ). Alice and Bob each receive n-bit inputs that either intersect in one coordinate or not at all.

High level idea. The lower bound follows from a randomized reduction from 2-DISJ to k-CONN. This reduction takes input (x, y) for 2-DISJ and produces input $(I_1, I_2, ..., I_k)$ to k-CONN by creating a graph G that, with high probability, has two cliques L and R joined by a **connector edge** precisely when x and y intersect. A connector edge is any edge with one endpoint in L and the other endpoint in R. Figure 5 shows an example graph G that could be created by our reduction.

Constructing a graph G for k-CONN. The first step is to find a hard input distribution for 2-DISJ that will yield the desired lower bound for k-CONN. It turns out that the following distribution is sufficiently hard. x and y contain a fixed number m of 1-bits. x and y intersect in one bit with probability $1/10k$, and nowhere with probability $1 - 1/10k$

Next, we must construct inputs $I_1, ..., I_k$ for k-CONN. Recall that, in the coordinator model, Alice simulates one player (in this case p_1) and Bob simulates the other $k - 1$ players and the coordinator (as shown in Figure 3). So, Alice will construct I_1 and Bob will construct $I_2, ..., I_k$. Consider the complete graph $K_{2n} = (V, E)$ with $2n$ vertices labeled $1, 2, ..., 2n$. Pick a random permutation σ of vertices $\{1, 2, ..., 2n\}$. This permutation induces a matching, i.e., a set E_σ of n edges in E. Specifically,

$$E_\sigma = \{(\sigma(2i-1), \sigma(2i)) \mid 1 \leq i \leq n\}.$$

For example, if $n = 4$ and $\sigma = \langle 3, 4, 7, 1, 8, 2, 5, 6\rangle$, then $E_\sigma = \{(3, 4), (7, 1), (8, 2), (5, 6)\}$. We will continue to develop this scenario as a running example for the remainder of this section.

Now, we want to turn inputs x and y into sets of edges E_x and E_y, respectively. We can define

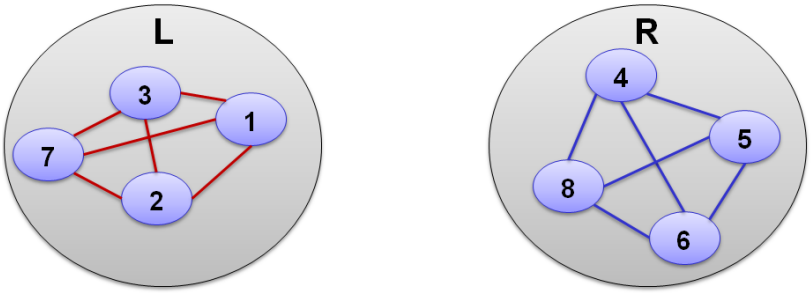

Figure 6: One possible outcome when Bob partitions the nodes in the running example

E_x and E_y as subsets of E_σ, as follows.

$$E_x = \{(\sigma(2i-1), \sigma(2i)) \mid x_i = 1\} \text{ and } E_y = \{(\sigma(2i-1), \sigma(2i)) \mid y_i = 1\}.$$

So, E_x and E_y are subsets of E_σ selected by the 1-bits of x and y, respectively. In our example, if $x = 0110$ and $y = 1010$, then $E_x = \{(7,1),(8,2)\}$ and $E_y = \{(3,4),(8,2)\}$.

Bob uses E_y to split V into two vertex sets V_L and V_R. For each edge in E_y, Bob puts one uniform random endpoint into V_L and the other into V_R. For each edge in $(E_\sigma - E_y)$, Bob puts both endpoints into either V_L or V_R uniformly randomly. To see why V_L and V_R form a partition of V, note that E_σ touches every node in V. To form the cliques L and R, we simply take the complete graphs on V_L and V_R, respectively. Let E_L and E_R be the sets of edges in L and R, respectively. Figure 6 illustrates one possible outcome when Bob partitions the nodes in our example.

At this point, it is helpful to make a few observations. Note that each edge in E_y has one endpoint in L and one in R, so it is a connector edge. It follows that, if x and y intersect, their intersection corresponds to a connector edge. Moreover, $E_\sigma \setminus E_y$ contains only non-connector edges, so any edge in E_x that is not in E_y is *not* a connector edge. We can build some intuition for why our construction makes sense, so far, by observing that the graph induced by $E_L \cup E_R \cup E_x \cup E_y$ is connected if and only if x and y intersect. It remains to divide these edges up amongst k players.

Constructing symmetric inputs for k-CONN. Alice sets $I_1 = E_x$. To construct I_j $(2 \leq j \leq k)$, Bob first chooses a uniform random subset of m (non-connector) edges from $E_L \cup E_R$. With probability $1/10k$, Bob throws out one of those m edges and replaces it with one (connector) edge from E_y.

We now argue that the inputs $I_2, ..., I_k$ come from the same input distribution as I_1 (which immediately implies $I_1, ..., I_k$ are symmetric). First, we characterize the input distribution for I_1. Since x contains m 1-bits, I_1 contains m edges. Since x and y intersect in one coordinate with probability $1/10k$ and nowhere with probability $1 - 1/10k$, I_1 contains one (connector) edge in E_y with probability $1/10k$ and no edges in E_y with probability $1 - 1/10k$. Recall that edges not in E_y are not connector edges, so with probability $1 - 1/10k$, I_1 contains m non-connector edges. Moreover, $E_L \cup E_R$ contains precisely the non-connector edges, so any edge in I_1 but not in E_y is in $E_L \cup E_R$. Therefore, with probability $1/10k$, I_1 contains one edge in E_y and $m - 1$ edges in $E_L \cup E_R$, and with probability $1 - 1/10k$, I_1 contains m edges in $E_L \cup E_R$. This input distribution precisely matches Bob's construction of $I_2, ..., I_k$.

Determining how often the reduction works. This reduction is randomized, so it does not always give the correct answer for 2-DISJ. We want to find a lower bound on the probability of

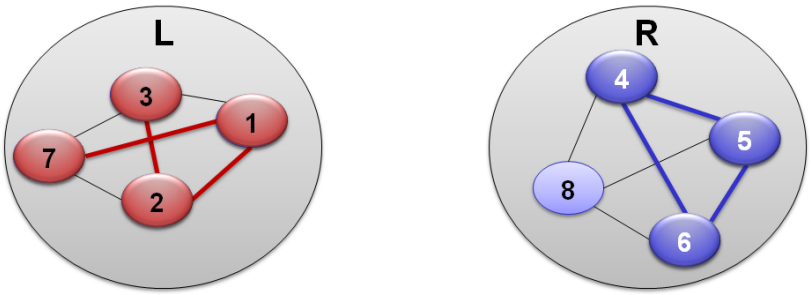

Figure 7: Depiction of $G_{Bob|L}$ and $G_{Bob|R}$ for our example. $G_{Bob|L}$ consists of the red nodes {1, 2, 3, 7} and edges. $G_{Bob|R}$ consists of the dark blue nodes {4, 5, 6} and edges.

computing the correct answer. It is helpful to consider the following edge induced graphs.

$$G_{Bob|L} = \sum_{i=2}^{k}(I_i \cap E_L) \text{ and } G_{Bob|R} = \sum_{i=2}^{k}(I_i \cap E_R)$$

Intuitively, $G_{Bob|L}$ is the graph induced by all edges in the inputs created by Bob, restricted to the graph L. ($G_{Bob|R}$ can be described similarly.) We extend our running example to better understand $G_{Bob|L}$ and $G_{Bob|R}$. Suppose $k = 5, I_1 = \{(1,3),(2,8)\}, I_2 = \{(4,6),(2,8)\}, I_3 = \{(3,2),(5,6)\}, I_4 = \{(4,5),(5,6)\}, I_5 = \{(1,2),(1,7)\}$. Figure 7 depicts $G_{Bob|L}$ and $G_{Bob|R}$ for our example.

We define two events that can help us understand when our reduction solves 2-DISJ.

- ζ_1 occurs if $G_{Bob|L}$ is connected and contains every node in L, **and** $G_{Bob|R}$ is connected and contains every node in R
- ζ_2 occurs if the union of inputs $I_2, ..., I_k$ created by Bob does **not** induce a connected graph

Intuitively, ζ_1 says we will have a connected graph if the input I_1 created by Alice contains a connector edge, and ζ_2 says that Alice's edges are needed to connect the graph.

Lemma 2 *If ζ_1 and ζ_2 both hold, then the reduction will compute the correct answer for 2-DISJ.*

Proof: Suppose ζ_1 holds. Then, $G_{Bob|L}$ and $G_{Bob|R}$ are each connected and include every node in L and R, respectively. Suppose ζ_2 holds. Then, no input created by Bob contains a connector edge. Thus, G is connected if and only if I_1 contains a connector edge, which happens if and only if x and y intersect. ■

It is interesting (but not necessary) to see why the reduction might give the wrong answer for 2-DISJ if ζ_1 and ζ_2 do not both hold. If ζ_1 does not hold, then k-CONN might say the graph is disconnected even if x intersects y. If ζ_1 holds, but ζ_2 does not, then k-CONN will say the graph is connected whether or not x intersects y.

The following lemma is proved in [4]. We simply state the lemma and use it to obtain a lower bound on the probability that the reduction computes the correct answer for 2-DISJ.

Lemma 3 *ζ_1 holds with probability at least $1 - 1/2n$.*

Recall that each input $I_2, ..., I_k$ created by Bob contains only non-connector edges with probability $1 - 1/10k$. It follows that every input created by Bob contains only non-connector edges with probability $(1 - 1/10k)^{k-1}$. Of course, if no input created by Bob contains a connector edge, then the graph induced by these edges cannot be connected. So, if ζ_1 holds, ζ_2 also holds with probability $(1 - 1/10k)^{k-1}$. Therefore, the reduction solves 2-DISJ with probability at least $(1 - 1/2n)(1 - 1/10k)^{k-1}$, which is greater than $8/9$ for large n.

Applying the symmetrization argument. Let P_k be the protocol for k-CONN and P_2 be the 2-DISJ protocol that uses P_k as a black box. We denote by $Cost(P_k)$ the amount of communication used by P_k running with inputs $I_1, ..., I_k$, and by $Cost(P_2)$ the amount of communication used by P_2 running with inputs x, y. Recall that Alice simulates one player and Bob simulates the other $k - 1$ and the coordinator (as shown in Figure 2). Since inputs are symmetric, in expectation, P_2 sends at most a $1/k$ fraction of the bits sent by P_k. So, the reduction solves 2-DISJ with $E[Cost(P_2)] \leq (1/k)Cost(P_k)$ and error bound $1/9$. Therefore, $Cost(P_k) \geq kE[Cost(P_2)]$.

It remains to obtain a lower bound on $E[Cost(P_2)]$. The following lemma is stated and proved in [4]. The proof is not difficult, but it does not inform our discussion, so it is omitted.

Lemma 4 *If x and y intersect with probability $1/10k$, then any 2-DISJ protocol with error bound $1/1000k$ has expected communication complexity $\Omega(n)$*

To apply this lemma, our reduction must err on at most a $1/1000k$ fraction of inputs. We reduce our protocol P_k's error of $1/9$ by running the randomized reduction $O(\log k)$ times, simulating P_k on each resulting input, and then carefully choosing one of the outputs we obtain. Details of this process can be found in the proof of Lemma 4.3 in [4]. After reducing our error, we can apply Lemma 4 to obtain $D_{1/4000k}(k\text{-CONN}) = \Omega(nk/\log k)$. (Note that we lost a multiplicative factor of $O(\log k)$ because the protocol had to be run $O(\log k)$ times.)

One weakness of this lower bound is that it says nothing about protocols with error $\epsilon > 1/4000k$. It turns out that this error bound can be changed to $1/3$ by paying an additional multiplicative factor of $O(\log k)$. Thus, we obtain $R_{1/3}(k\text{-CONN}) = \Omega(nk/\log^2 k)$.

2.5.1 Summary of symmetrization results

Results we covered:

- $D(k\text{-XOR}) = \Omega(nk)$ blackboard
- $R_\epsilon(k\text{-XOR}) = \Omega((1 - \epsilon)nk)$ blackboard
- $R_{1/3}(k\text{-CONN}) = \Omega(nk/\log^2 k)$ coordinator

There were also several more results in [4] that we did not cover, including results for k-party coordinate-wise OR and for the problem of determining, for each input coordinate, whether a majority of players has a 1-bit in that coordinate (k-MAJ).

- $R_{1/3}(k\text{-OR}) = \Omega(n \log k)$ blackboard (easy $O(n \log n)$ upper bound)
- $R_{1/800}(k\text{-OR}) = \Omega(nk)$ coordinator
- $R_{1/6}(k\text{-MAJ}) = \Omega(nk)$ coordinator

Crucially, the authors of [4] could not apply symmetrization to the k-party set disjointness problem (k-DISJ), which outputs 1 if all players' inputs intersect in some coordinate, and 0 otherwise. In fact, the authors believe that symmetrization *cannot* be used to obtain an $\Omega(nk)$ lower bound for k-DISJ.

2.6 Tight bounds on k-party set disjointness (k-DISJ)

The symmetrization paper created significant interest in k-DISJ. One of the most fundamental and well understood functions in 2-party communication complexity had thus far confounded researchers in a k-party setting. Some property of the k-DISJ problem rendered the symmetrization technique inapplicable, but there was hope that another technique might succeed where symmetrization failed. In 2013, Braverman et al. [1] obtained a tight $\Omega(nk)$ lower bound on the communication complexity of k-DISJ in the (asynchronous) coordinator model using information theoretic techniques.

In the context of a k-party protocol P, information theory studies the amount of information about the set of inputs $x = \{x^1, x^2, ..., x^k\}$ that can be learned by observing various subsets of the total communication between parties. For each execution of P, a transcript $\pi(x)$ records the communication between all parties. Two popular measures of the information that can be learned about the inputs are *external information cost* and *internal information cost*.

External information cost (EIC) captures the total amount of information that an external observer who can see the entire transcript $\pi(x)$ can learn about the inputs of the players. (In the coordinator model, EIC corresponds to what the coordinator can learn about the inputs.) Formally, EIC $= I(x; \pi(x))$ is the information about the set x of inputs that can be learned from the transcript of P running on x.

Internal information cost (IIC) captures the total amount of information that the players (together) can learn about the inputs, if each is initially given his own input. Let $x^{-p} = x \setminus \{x^p\}$, and $\pi^p(x)$ be the transcript $\pi(x)$ restricted to contain only communication to and from p. Formally, IIC $= \sum_p I(x^{-p}; \pi^p(x) \mid x^p)$ is the sum over all players p of the information about the inputs x^{-p} that can be learned from the restricted transcript $\pi^p(x)$, given p's input.

In order to obtain an $\Omega(nk)$ lower bound, it is necessary to combine IIC and EIC. To see why, we consider two simple protocols for solving k-DISJ, and determine the IIC and EIC of each.

In the first protocol, each player sends his input to the coordinator. The coordinator computes k-DISJ and tells each player the answer. This protocol has IIC $= 0$, since no player learns anything about the input of any other player. Since there is a k-DISJ protocol with IIC $= 0$, we cannot use IIC to obtain a lower bound on the information cost of k-DISJ. However, in this protocol, the coordinator learns the inputs of every player, so EIC $= \Omega(nk)$. Perhaps, then, it is possible to use EIC to obtain the desired lower bound.

We now describe the second protocol. For each coordinate j, and for each player p, the coordinator asks each player p if $x_{p,j} = 0$. If a player p ever responds that $x_{p,j} = 0$, then the coordinator skips to coordinate $j + 1$ without asking any player $m > p$. If, for some coordinate j, every player has a 1-bit, then the protocol outputs zero (since all inputs intersect in that coordinate). Otherwise, the protocol outputs one (since there was a zero-bit in each coordinate). It turns out that we can compress any execution transcript for this protocol to $O(n \log k)$ bits since, for each coordinate j, we can simply write the name of the first player p with $x_{p,j} = 0$ or write 0 if p does not exist. Since we can represent the entire transcript using $O(n \log k)$ bits, the protocol must reveal at most

$O(n \log k)$ bits of information to the coordinator. Consequently, we obtain EIC $\leq O(n \log k)$. Since there is a protocol for k-DISJ with EIC $\leq O(n \log k)$, we cannot use EIC to obtain an $\Omega(nk)$ lower bound on the information cost of k-DISJ. However, we can show that the IIC for this protocol is $\Omega(nk)$ for some inputs. Consider the input wherein x^p $(1 \leq p < k)$ contains all 1-bits and x^k contains all 0-bits. When any player p is asked whether $x_{p,j} = 0$, it learns that $x_{m,j} = 1$ for all $m < p$. So, every time player p is asked if $x_{p,j} = 0$, it learns $p - 1$ bits of other players' inputs. Summing over all players p and coordinates j, we get IIC $\geq \sum_p^k \sum_j^n (p-1) = n(k-1) = \Omega(nk)$.

For these two protocols, we obtain low-IIC/high-EIC and high-IIC/low-EIC, respectively, which suggests there might be a tradeoff between IIC and EIC when designing a protocol for k-DISJ. It turns out that this tradeoff is inherent, so IIC and EIC must be used in conjunction to prove an $\Omega(nk)$ lower bound on k-DISJ.

Carefully blending IIC and EIC and finding a hard input distribution represents one of the main challenges of the work. For simplicity, we avoid discussing the details of the input distribution, and use $InfoCost$ to denote an appropriate blend of IIC and EIC. At a high level, the proof for the lower bound begins by showing that $InfoCost$(n-bit k-DISJ) $\geq n \cdot InfoCost$(1-bit k-DISJ), where 1-bit k-DISJ is the same as k-DISJ except that each player has a 1-bit input. It then shows that $InfoCost$(1-bit k-DISJ) $\geq InfoCost$(1-bit k-AND), where 1-bit k-AND is the problem of computing the AND of the 1-bit inputs of k players. Finally, it shows that $InfoCost$(1-bit k-AND) $\geq \Omega(k)$.

Results. The authors obtain a tight $\Omega(nk)$ lower bound for k-DISJ with $k = \Omega(\log n)$ in the asynchronous coordinator model. They also obtain a slightly weaker $\Omega(nk/\log k)$ lower bound in the asynchronous message passing model.

2.7 Communication graph topology dependent lower bounds

The message passing model in which Braverman et al. proved their $\Omega(nk)$ lower bound on k-DISJ assumes a fully connected communication graph (meaning that every player can communicate directly with every other player). Therefore, this lower bound might leave room for improvement in other graph topologies, where communication is more restricted, and information must pass through some bottleneck.

A 2014 paper by Chattopadhyay, Radhakrishnan and Rudha [2] studied lower bounds for several problems in different graph topologies. They developed techniques for obtaining lower bounds that are sensitive to the topology of the communication graph. As a consequence, they were able to obtain $\omega(nk)$ (asymptotically greater than nk) lower bounds for several problems. One particularly interesting discovery to come out of their work was the fact that k-DISJ cannot be used to get $\omega(nk)$ lower bounds. Disjointness is widely used in 2-party lower bounds, partially because it is natural to reduce many problems to it, and no other 2-party problem is asymptotically harder. However, it turns out there are much harder functions in a k-party setting for some communication graph topologies.

To see that k-DISJ cannot be used to obtain $\omega(nk)$ lower bounds, observe that the following protocol computes set intersection (and, hence, disjointness) with communication $O(nk)$ for any connected communication graph. First, all players (locally) compute the same spanning tree for the communication graph. Let any arbitrary player p_{root} be the root of the spanning tree. Each player that is a leaf in the spanning tree sends his input to his parent. For every other player p, upon receiving n-bit strings $s_1, s_2, ..., s_c$ from all c of his children in the tree, p computes the

intersection of $s_1, s_2, ..., s_c$ and his own input, and forwards the result to his parent. The spanning tree contains exactly $k-1$ edges, each of which carries n bits over an execution of the protocol, for a total of $(k-1)n$ bits. At the end of this protocol, p_{root} knows the intersection of all inputs. If *every* player needs to know the intersection, then p_{root} can simply send the intersection to his children, who relay this information downwards in the tree, doubling the amount of communication used by the algorithm.

Since k-DISJ is not hard enough to get $\Omega(nk)$ lower bounds, we need another problem to study. The paper describes a number of harder problems. One such problem is the *element-distinctness* problem (k-DIST), which requires each player to output one if all inputs are pairwise distinct, and zero otherwise. There are several hard graph problems wherein k players each have a subgraph of a graph H as input, and each edge can appear in the inputs of multiple players. In this setting, hard problems include determining the degree of a vertex in H, and checking whether H is acyclic, triangle-free, bipartite or connected. For these problems (and others), the paper proves that the trivial solution of sending all inputs to a designated player who computes the answer is optimal up to $polylog(k)$ factors. It is not hard to see that, in a graph with diameter d, sending all inputs to any player takes $\Omega(nkd)$ communication (intuitively, since a constant fraction of players need to send their inputs over a constant fraction of d edges). Of course, when the graph is fully connected, $d = 1$, and the lower bound becomes $\Omega(nk)$. When the graph diameter is k (as large as possible), sending all inputs takes $\Omega(nk^2)$ communication. The paper obtains $\Omega(nk^2)$ lower bounds on the randomized communication complexity of many problems, beating previous lower bounds of $\Omega(nk)$ for fully connected communication graphs.

We briefly mention one open problem described in [2]. Currently, very little is known about the communication complexity of composed functions in arbitrary graph topologies. It is impossible to obtain non-trivial topology dependent lower bounds for all composed functions. However, it would be interesting to have broad conditions on the functions being composed that are sufficient to obtain non-trivial topology dependent lower bounds. Even for the special case where the outer function is k-AND or k-OR, the authors suspect that the problem is non-trivial.

References

[1] M. Braverman, F. Ellen, R. Oshman, T. Pitassi, and V. Vaikuntanathan, *A tight bound for set disjointness in the message-passing model*, in Proceedings of the 54th IEEE Symposium on Foundations of Computer Science, IEEE, 2013, pp. 668–677.

[2] A. Chattopadhyay, J. Radhakrishnan, and A. Rudra, *Topology matters in communication*, in Proceedings of the 55th IEEE Symposium on Foundations of Computer Science (to appear), IEEE, 2014.

[3] F. Ellen, R. Oshman, T. Pitassi, and V. Vaikuntanathan, *Brief announcement: Private channel models in multi-party communication complexity*, in Proceedings of the 27th ACM Symposium on Distributed Computing, Springer, 2013, p. 575.

[4] J. M. Phillips, E. Verbin, and Q. Zhang, *Lower bounds for number-in-hand multiparty communication complexity, made easy*, in Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2012, pp. 486–501.

3 Summary notes

Symmetrization ($k \geq 100 \log n$)

- k-XOR coordinator, blackboard $\Omega(nk)$ CC
- k-CONN coordinator $\Omega(nk/\log^2 k)$ randomized CC
- No results for k-DISJ

Set disjointness ($k = \Omega(\log n)$)

- k-DISJ coordinator $\Theta(nk)$ CC
- k-DISJ message passing $\Omega(nk/\log k)$ CC
- Only for strongly connected communication graphs

Topology matters

- k-DISJ requires $O(nk)$ communication for every connected communication graph
- New hard problems such as element-distinctness yield $\Omega(nk^2)$ lower bounds
- Open: topology dependent complexity of composed functions