CS369E: Communication Complexity (for Algorithm Designers) Lecture #4: Boot Camp on Communication Complexity*

Tim Roughgarden†

January 29, 2015

1 Preamble

This lecture covers the most important basic facts about deterministic and randomized communication protocols in the general two-party model, as defined by Yao [8]. Some version of this lecture would normally be the first lecture in a course on communication complexity. How come it's the fourth one here?

The first three lectures were about one-way communication complexity — communication protocols where there is only one message, from Alice to Bob — and its applications. One reason we started with the one-way model is that several of the "greatest hits" of algorithmic lower bounds via communication complexity, such as space lower bounds for streaming algorithms and row lower bounds for compressive sensing matrices, already follow from communication lower bounds for one-way protocols. A second reason is that considering only one-way protocols is a gentle introduction to what communication protocols look like. There are already some non-trivial one-way protocols, like our randomized protocol for Equality. On the other hand, proving lower bounds for one-way protocols is much easier than proving them for general protocols, so it's also a good introduction to lower bound proofs.

The rest of our algorithmic applications require stronger lower bounds that apply to more than just one-way protocols. This lecture gives a "boot camp" on the basic model. We won't say much about applications in this lecture, but the final five lectures all focus on applications. We won't prove any hard results today, and focus instead on definitions and vocabulary, examples, and some easy results. One point of today's lecture is to get a feel for what's involved in proving a communication lower bound for general protocols. It generally boils down to a conceptually simple, if sometimes mathematically challenging, combinatorial

†Department of Computer Science, Stanford University, 474 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

problem — proving that a large number of "rectangles" of a certain type are need to cover a matrix.[1]

2 Deterministic Protocols

2.1 Protocols

We are still in the two-party model, where Alice has an input $\mathbf{x} \in X$ unknown to Bob, and Bob has an input $\mathbf{y} \in Y$ unknown to Alice. (Most commonly, $X = Y = \{0,1\}^n$.) A deterministic communication protocol specifies, as function of the messages sent so far, whose turn it is to speak. A protocol always specifies when the communication has ended and, in each end state, the value of the computed bit. Alice and Bob can coordinate in advance to decide upon the protocol, and both are assumed to cooperative fully. The only constraint faced by the players is that what a player says can depend only on what the player knows — his or her own input, and the history of all messages sent so far.

Like with one-way protocols, we define the cost of a protocol as the maximum number of bits it ever sends, ranging over all inputs. The communication complexity of a function is then the minimum communication cost of a protocol that correctly computes it.

The key feature of general communication protocols absent from the special case of one-way protocols is *interaction* between the two players. Intuitively, interaction should allow the players to communicate much more efficiently. Let's see this in a concrete example.

2.2 Example: Clique-Independent Set

The following problem might seem contrived, but it is fairly central in communication complexity. There is a graph $G = (V, E)$ with $|V| = n$ that is known to both players. Alice's private input is a clique C of G — a subset of vertices such that $(u, v) \in E$ for every distinct $u, v \in C$. Bob's private input is an independent set I of G — a subset of vertices such that $(u, v) \notin E$ for every distinct $u, v \in I$. (There is no requirement that C or I is maximal.) Observe that C and I are either disjoint, or they intersect in a single vertex (Figure 1). The players' goal is to figure out which of these two possibilities is the case. Thus, this problem is a special case of Disjointness, where players' sets are not arbitrary but rather a clique and an independent set from a known graph.

The naive communication protocol for solving the problem using $\Theta(n)$ bits — Alice can send the characteristic vector of C to Bob, or Bob the characteristic vector of I to Alice, and then the other player computes the correct answer. Since the number of cliques and independent sets of a graph is generally exponential in the number n of vertices, this protocol cannot be made significantly more communication-efficient via a smarter encoding. An easy

[1] There are many other methods for proving communication lower bounds, some quite deep and exotic (see e.g. [5]), but all of our algorithmic applications can ultimately be derived from combinatorial covering-type arguments. For example, we're not even going to mention the famous "rank lower bound." For your edification, some other lower bound methods are discussed in the Exercises.

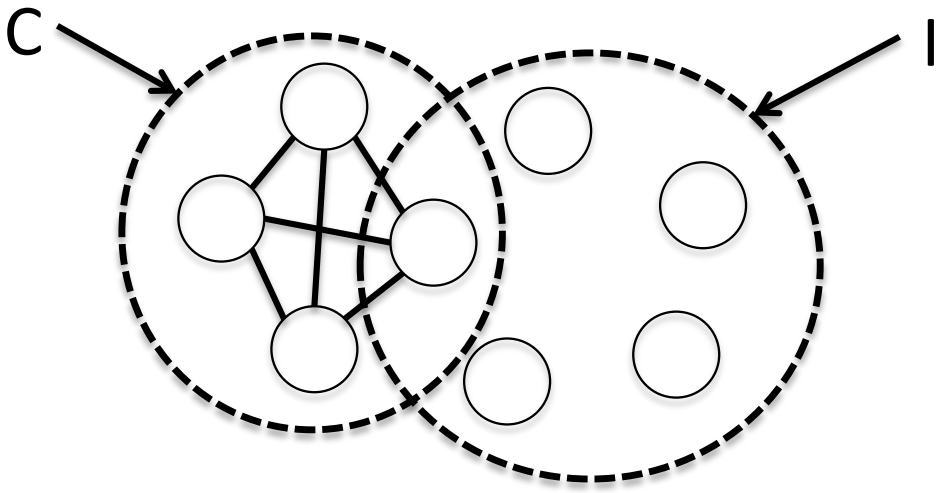

Figure 1: A clique C and an independent set I overlap in zero or one vertices.

reduction from INDEX shows that one-way protocols, including randomized protocols, require $\Omega(n)$ communication (exercise).

The players can do much better by interacting. Here is the protocol.

1. If there is a vertex $v \in C$ with $\deg(v) < \frac{n}{2}$, then Alice sends the name of an arbitrary such vertex to Bob ($\approx \log_2 n$ bits).

   (a) Bob announces whether or not $v \in I$ (1 bit). If so, the protocol terminates with conclusion "not disjoint."

   (b) Otherwise, Alice and Bob recurse on the subgraph H induced by v and its neighbors.

   [Note: H contains at most half the nodes of G. It contains all of C and, if I intersects C, it contains the vertex in their intersection. C and I intersect in G if and only if their projections to H intersect in H.]

2. Otherwise, Alice sends a "NULL" message to Bob ($\approx \log_2 n$ bits).

3. If there is a vertex $v \in I$ with $\deg(v) \geq \frac{n}{2}$, then Bob sends the name of an arbitrary such vertex to Bob ($\approx \log_2 n$ bits).

   (a) Alice announces whether or not $v \in C$ (1 bit). If so, the protocol terminates ("not disjoint").

   (b) If not, Alice and Bob recurse on the subgraph H induced by v and its non-neighbors.

   [Note: H contains at most half the nodes of G. It contains all of I and, if C intersects I, it contains the vertex in their intersection. Thus the function's answer in H is the same as that in G.]

4. Otherwise, Bob terminates the protocol and declares "disjoint."

   [Disjointness is obvious since, at this point in the protocol, we know that $\deg(v) < \frac{n}{2}$ for all $v \in C$ and $\deg(v) \geq \frac{n}{2}$ for all $v \in I$.]

Since each iteration of the protocol uses $O(\log n)$ bits of communication and cuts the number of vertices of the graph in half (or terminates), the total communication is $O(\log^2 n)$. As previously noted, such a result is impossible without interaction between the players.

2.3 Trees and Matrices

The Clique-Independent Set problem clearly demonstrates that we need new lower bound techniques to handle general communication protocols — the straightforward Pigeonhole Principle arguments that worked for one-way protocols are not going to be good enough. At first blush this might seem intimidating — communication protocols can do all sorts of crazy things, so how can we reason about them in a principled way? How can we connect properties of a protocol to the function that it computes? Happily, we can quickly build up some powerful machinery for answering these questions.

First, we observe that deterministic communication protocols are really just binary trees. We'll almost never use this fact directly, but it should build some confidence that protocols are familiar and elementary mathematical objects.

The connection is easiest to see by example; see Figure 2. Consider the following protocol for solving Equality with $n = 2$ (i.e., $f(\mathbf{x}, \mathbf{y}) = 1$ if and only if $\mathbf{x} = \mathbf{y}$). Alice begins by sending her first bit. If Bob's first bit is different, he terminates the protocol and announces "not equal." If Bob's first bit is the same, then he transmits the same bit back. In this case, Alice then sends her second bit. At this point, Bob knows Alice's whole input and can therefore compute the correct answer.

In Figure 2, each node corresponds to a possible state of the protocol, and is labeled with the player whose turn it is to speak. Thus the labels alternate with the levels, with the root belonging to Alice.[2] There are 10 leaves, representing the possible end states of the protocol. There are two leaves for the case where Alice and Bob have different first bits and the protocol terminates early, and eight leaves for the remaining cases where Alice and Bob have the same first bit. Note that the possible transcripts of the protocol are in one-to-one correspondence with the root-leaf nodes of the tree — we use leaves and transcripts interchangeably below.

We can view the leaves as a partition $\{Z(\ell)\}$ of the input space $X \times Y$, with $Z(\ell)$ the inputs $(\mathbf{x}, \mathbf{y})$ such that the protocol terminates in the leaf ℓ. In our example, there are 10 leaves for the 16 possible inputs $(\mathbf{x}, \mathbf{y})$, so different inputs can generate the same transcript — more on this shortly.

Next note that we can represent a function (from $(\mathbf{x}, \mathbf{y})$ to $\{0, 1\}$) a matrix. In contrast to the visualization exercise above, we'll use this matrix representation *all the time*. The rows are labeled with the set X of possible inputs of Alice, the columns with the set Y of

[2] In general, players need not alternate turns in a communication protocol.

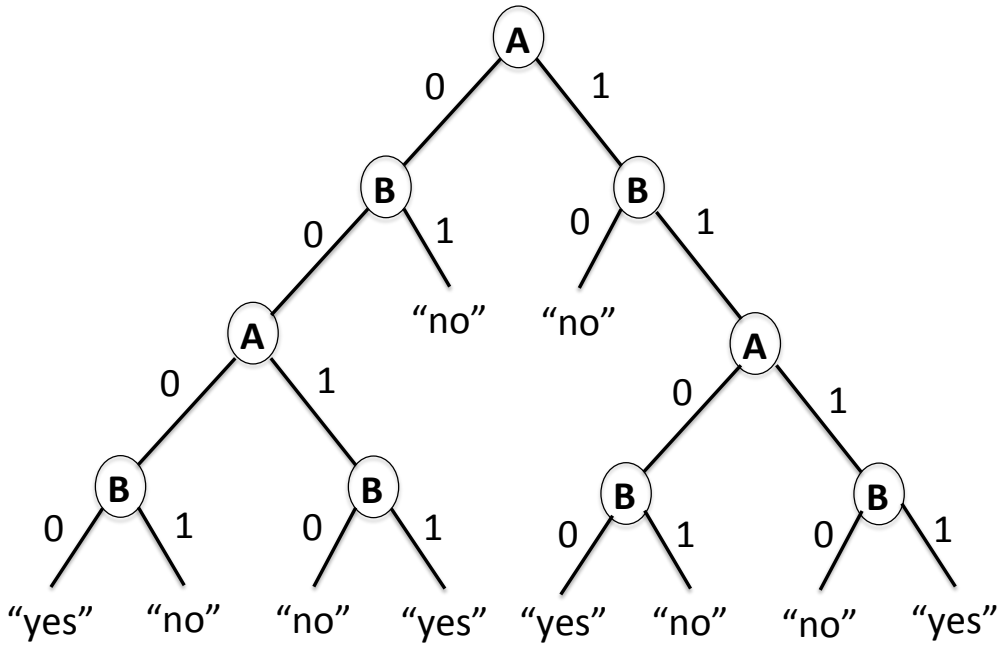

Figure 2: The binary tree induced by a communication protocol for EQUALITY with $n = 2$.

possible inputs of Bob. Entry $(\mathbf{x}, \mathbf{y})$ of the matrix is $f(\mathbf{x}, \mathbf{y})$. Keep in mind that this matrix is fully known to both Alice and Bob when they agree on a protocol.

For example, suppose that $X = Y = \{0,1\}^2$, resulting in 4×4 matrices. EQUALITY then corresponds to the identity matrix:

$$\begin{array}{c} \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{1}$$

If we define the GREATER-THAN function as 1 whenever $\mathbf{x}$ is at least $\mathbf{y}$ (where $\mathbf{x}$ and $\mathbf{y}$ are interpreted as non-negative integers, written in binary), then we just fill in the lower triangle with 1s:

$$\begin{array}{c} \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

We also write out the matrix for DISJOINTNESS, which is somewhat more inscrutable:

$$\begin{array}{c} \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

2.4 Protocols and Rectangles

How can we reason about the behavior of a protocol? Just visualizing them as trees is not directly useful. We know that simple Pigeonhole Principle-based arguments are not strong enough, but it still feels like we want some kind of counting argument.

To see what might be true, let's run the 2-bit EQUALITY protocol depicted in Figure 2 and track its progress using the matrix in (1). Put yourself in the shoes of an outside observer, who knows neither $\mathbf{x}$ nor $\mathbf{y}$, and makes inferences about $(\mathbf{x}, \mathbf{y})$ as the protocol proceeds. When the protocol terminates, we'll have carved up the matrix into 10 pieces, one for each leaf of protocol tree — the protocol transcript reveals the leaf to an outside observer, but nothing more.

Before the protocol beings, all 16 inputs are fair game. After Alice sends her first bit, the outside observer can narrow down the possible inputs into a set of 8 — the top 8 if Alice sent a 0, the bottom 8 if she sent a 1. The next bit sent gives away whether or not Bob's first bit is a 0 or 1, so the outsider observer learns which quadrant the input lies in. Interestingly, in the northeastern and southwestern quadrants, all of the entries are 0. In these cases, even though ambiguity remains about exactly what the input $(\mathbf{x}, \mathbf{y})$ is, the function's value $f(\mathbf{x}, \mathbf{y})$ has been determined (it is 0, whatever the input). It's no coincidence that these two regions correspond to the two leaves of the protocol in Figure 2 that stop early, with the correct answer. If the protocol continues further, then Alice's second bit splits the northwestern and southeastern quadrants into two, and Bob's final bit splits them again, now into singleton regions. In these cases, an outside observer learns the entire input $(\mathbf{x}, \mathbf{y})$ from the protocol's transcript.[3]

What have we learned? We already knew that every protocol induces a partition of the input space $X \times Y$, with one set for each leaf or, equivalently, for each distinct transcript. At least for the particular protocol that we just studied, each of the sets has a particularly nice submatrix form (Figure 3). This is true in general, in the following sense.

Lemma 2.1 (Rectangles) *For every transcript* $\mathbf{z}$ *of a deterministic protocol* P*, the set of inputs* $(\mathbf{x}, \mathbf{y})$ *that generate* $\mathbf{z}$ *are a* rectangle*, of the form* $A \times B$ *for* $A \subseteq X$ *and* $B \subseteq Y$*.*

[3]It's also interesting to do an analogous thought experiment from the perspective of one of the players. For example, consider Bob's perspective when the input is (00,01). Initially Bob knows that the input lies in the second column but is unsure of the row. After Alice's first message, Bob knows that the input is in the second column and one of the first two rows. Bob still cannot be sure about the correct answer, so the protocol proceeds.

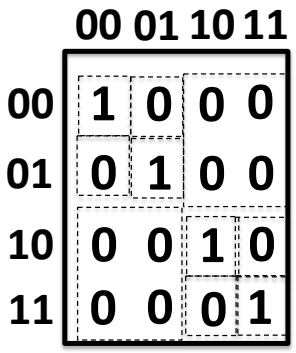

Figure 3: The partition of the input space $X \times Y$ according to the 10 different transcripts that can be generated by the EQUALITY protocol.

A rectangle just means a subset of the input space $X \times Y$ that can be written as a product. For example, the set $\{(00, 00), (11, 11)\}$ is *not* a rectangle, while the set $\{(00, 00), (11, 00), (00, 11), (11, 11)\}$ is. In general, a subset $S \subseteq X \times Y$ is a rectangle if and only if it is closed under "mix and match," meaning that whenever $(\mathbf{x}_1, \mathbf{y}_1)$ and $(\mathbf{x}_2, \mathbf{y}_2)$ are in S, so are $(\mathbf{x}_1, \mathbf{y}_2)$ and $(\mathbf{x}_2, \mathbf{y}_1)$ (see the Exercises).

Don't be misled by our example (Figure 3), where the rectangles induced by our protocol happen to be "contiguous." For example, if we keep the protocol the same but switch the order in which we write down the rows and columns corresponding to 01 and 10, we get an analogous decomposition in which the two large rectangles are not contiguous. In general, you shouldn't even think of X and Y as ordered sets. Rectangles are sometimes called *combinatorial* rectangles to distinguish them from "geometric" rectangles and to emphasize this point.

Lemma 2.1 is extremely important, though its proof is straightforward — we just follow the protocol like in our example above. Intuitively, each step of a protocol allows an outside observer to narrow down the possibilities for $\mathbf{x}$ while leaving the possibilities for $\mathbf{y}$ unchanged (if Alice speaks) or vice versa (if Bob speaks).

Proof of Lemma 2.1: Fix a deterministic protocol P. We proceed by induction on the number of bits exchanged. For the base case, all inputs $X \times Y$ begin with the empty transcript. For the inductive step, consider an arbitrary t-bit transcript-so-far $\mathbf{z}$ generated by P, with $t \geq 1$. Assume that Alice was the most recent player to speak; the other case is analogous. Let $\mathbf{z}'$ denote $\mathbf{z}$ with the final bit $b \in \{0, 1\}$ lopped off. By the inductive hypothesis, the set of inputs that generate $\mathbf{z}'$ has the form $A \times B$. Let $A_b \subseteq A$ denote the inputs $\mathbf{x} \in A$ such that, in the protocol P, Alice sends the bit b given the transcript $\mathbf{z}'$. (Recall that the message sent by a player is a function only of his or her private input and the history of the protocol so far.) Then the set of inputs that generate $\mathbf{z}$ are $A_b \times B$, completing the inductive step. ■

Note that Lemma 2.1 makes no reference to a function f — it holds for any deterministic protocol, whether or not it computes a function that we care about. In Figure 3, we can clearly see an additional property of all of the rectangles — with respect to the matrix in (1),

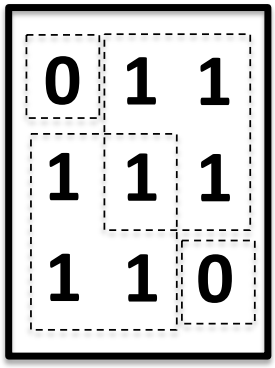

Figure 4: A covering by four monochromatic rectangles that is not a partition.

every rectangle is *monochromatic*, meaning all of its entries have the same value. This is true for any protocol that correctly computes a function f.

Lemma 2.2 *If a deterministic protocol P computes a function f, then every rectangle induced by P is monochromatic in the matrix $M(f)$.*

Proof: Consider an arbitrary combinatorial rectangle $A \times B$ induced by P, with all inputs in $A \times B$ inducing the same transcript. The output of P is constant on $A \times B$. Since P correctly computes f, f is also constant on $A \times B$. ■

Amazingly, the minimal work we've invested so far already yields a powerful technique for lower bounding the deterministic communication complexity of functions.

Theorem 2.3 *Let f be a function such that every partition of $M(f)$ into monochromatic rectangles requires at least t rectangles. Then the deterministic communication complexity of f is at least $\log_2 t$.*

Proof: A deterministic protocol with communication cost c can only generate 2^c distinct transcripts — equivalently, its (binary) protocol tree can only have 2^c leaves. If such a protocol computes the function f, then by Lemmas 2.1 and 2.2 it partitions $M(f)$ into at most 2^c monochromatic rectangles. By assumption, $2^c \geq t$ and hence $c \geq \log_2 t$. ■

Rather than applying Theorem 2.3 directly, we'll almost always be able to prove a stronger and simpler condition. To partition a matrix, one needs to cover all of its entries with disjoint sets. The disjointness condition is annoying. So by a *covering* of a 0-1 matrix, we mean a collection of subsets of entries whose union includes all of its elements — overlaps between these sets are allowed. See Figure 4.

Corollary 2.4 *Let f be a function such that every covering of $M(f)$ by monochromatic rectangles requires at least t rectangles. Then the deterministic communication complexity of f is at least $\log_2 t$.*

Communication complexity lower bounds proved using covers — including all of those proved in Section 2.5 — automatically apply also to more general "nondeterministic" communication protocols, as well as randomized protocols with 1-sided error. We'll discuss this more next lecture, when it will be relevant.

2.5 Lower Bounds for Equality and Disjointness

Armed with Corollary 2.4, we can quickly prove communication lower bounds for some functions of interest. For example, recall that when f is the Equality function, the matrix $M(f)$ is the identity. The key observation about this matrix is: *a monochromatic rectangle that includes a "1" contains only one element.* The reason is simple: such a rectangle is not allowed to contain any 0's since it is monochromatic, and if it included a second 1 it would pick up some 0-entries as well (recall that rectangles are closed under "mix and match"). Since there are 2^n 1's in the matrix, every covering by monochromatic rectangles (even of just the 1's) has size 2^n.

Corollary 2.5 *The deterministic communication complexity of* Equality *is at least* n.[4]

The exact same argument gives the same lower bound for the Greater-Than function.

Corollary 2.6 *The deterministic communication complexity of* Greater-Than *is at least* n.

We can generalize this argument as follows. A *fooling set* for a function f is a subset $F \subseteq X \times Y$ of inputs such that:

(i) f is constant on F;

(ii) for each distinct pair $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2) \in F$, at least one of $(\mathbf{x}_1, \mathbf{y}_2), (\mathbf{x}_2, \mathbf{y}_1)$ has the opposite f-value.

Since rectangles are closed under the "mix and match" operation, (i) and (ii) imply that every monochromatic rectangle contains at most one element of F.

Corollary 2.7 *If F is a fooling set for f, then the deterministic communication complexity of f is at least* $\log_2 |F|$.

For Equality and Greater-Than, we were effectively using the fooling set $F = \{(\mathbf{x}, \mathbf{x}) : \mathbf{x} \in \{0,1\}^n\}$.

The fooling set method is powerful enough to prove a strong lower bound on the deterministic communication complexity of Disjointness.

[4]The 0s can be covered using another 2^n monochromatic rectangles, one per row (rectangles need not be "contiguous"!). This gives a lower bound of $n+1$. The trivial upper has Alice sending her input to Bob and Bob announcing the answer, which is a $(n+1)$-bit protocol. Analogous "+1" improvements are possible for the other examples in this section.

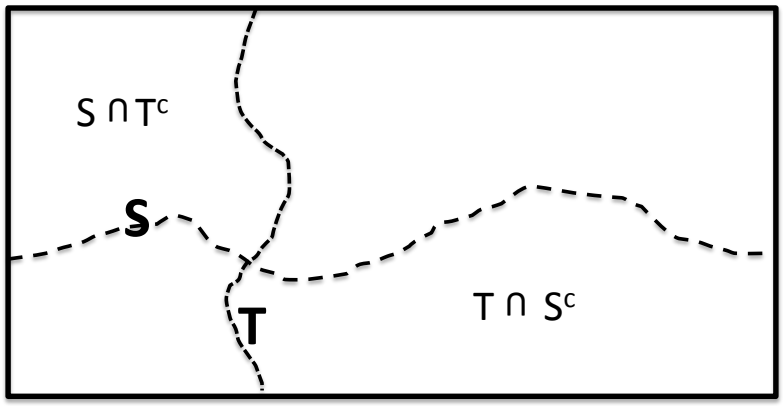

Figure 5: If S and T are different sets, then either S and T^c or T and S^c are not disjoint.

Corollary 2.8 *The deterministic communication complexity of* DISJOINTNESS *is at least* n.

Proof: Take $F = \{(\mathbf{x}, \mathbf{1} - \mathbf{x}) : \mathbf{x} \in \{0,1\}^n\}$ — or in set notation, $\{(S, S^c) : S \subseteq \{1, 2, \ldots, n\}\}$. The set F is a fooling set — it obviously consists only of "yes" inputs of DISJOINTNESS, while for every $S \neq T$, either $S \cap T^c \neq \emptyset$ or $T \cap S^c \neq \emptyset$ (or both). See Figure 5. Since $|F| = 2^n$, Corollary 2.7 completes the proof. ■

2.6 Take-Aways

A key take-away point from this section is that, using covering arguments, we can prove the lower bounds that we want on the deterministic communication complexity of many functions of interest. These lower bounds apply also to nondeterministic protocols (discussed next week) and randomized protocols with 1-sided error.

As with one-way communication complexity, proving stronger lower bounds that apply also to randomized protocols with two-sided error is more challenging. Since we're usually perfectly happy with a good randomized algorithm — recall the F_2 estimation algorithm from Lecture #1 — such lower bounds are very relevant for algorithmic applications. They are our next topic.

3 Randomized Protocols

3.1 Default Parameter Settings

Our discussion of randomized one-way communication protocols in Lecture #2 remains equally relevant for general protocols. Our "default parameter settings" for such protocols will be the same.

Public coins. By default, we work with public-coin protocols, where Alice and Bob have shared randomness in the form of an infinite sequence of perfectly random bits written on

a blackboard in public view. Such protocols are more powerful than private-coin protocols, but not by much (Theorem 3.1). Recall that public-coin randomized protocols are equivalent to distributions over deterministic protocols.

Two-sided error. We allow a protocol to error with constant probability ($\frac{1}{3}$ by default), whether or not the correct answer is "1" or "0." This is the most permissive error model.

Arbitrary constant error probability. Recall that all constant error probabilities in $(0, \frac{1}{2})$ are the same — changing the error changes the randomized communication complexity by only a constant factor (by the usual "independent trials" argument, detailed in the exercises). Thus for upper bounds, we'll be content to achieve error 49%; for lower bounds, it is enough to rule out low-communication protocols with error %1.

Worst-case communication. We define the communication cost of a randomized protocol as the maximum number of bits ever communicated, over all choices of inputs and coin flips. Measuring the expected communication (over the protocol's coin flips) could reduce the communication complexity of a problem, but only by a constant factor.

3.2 Newman's Theorem: Public- vs. Private-Coin Protocols

We mentioned a few times that, for our purposes, it usually won't matter whether we consider public-coin or private-coin randomized protocols. What we meant is the following result.

Theorem 3.1 (Newman's Theorem [6]) *If there is a public-coin protocol for a function f with n-bit inputs that has two-sided error $1/3$ and communication cost c, then there is a private-coin protocol for the problem that has two-sided error $1/3$ and communication cost $O(c + \log n)$.*

Thus, for problems with public-coin randomized communication complexity $\Omega(\log n)$, like most of the problems that we'll study in this course, there is no difference between the communication complexity of the public-coin and private-coin variants (modulo constant factors).

An interesting exception is Equality. Last lecture, we gave a public-coin protocol — one-way, even — with constant communication complexity. Theorem 3.1 only implies an upper bound of $O(\log n)$ communication for private-coin protocols. (One can also give such a private-coin protocol directly, see the Exercises.) There is also a matching lower bound of $\Omega(\log n)$ for the private-coin communication complexity of Equality. (This isn't very hard to prove, but we won't have an occasion to do it.) Thus public-coin protocols can save $\Theta(\log n)$ bits of communication over private-coin protocols, but no more.

Proof of Theorem 3.1: Let P denote a public-coin protocol with two-sided error $1/3$. We begin with a thought experiment. Fix an input $(\mathbf{x}, \mathbf{y})$, with $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$. If we run P on this input, a public string $\mathbf{r}_1$ of random bits is consumed and the output of the protocol is correct with probability at least $2/3$. If we run it again, a second (independent) random string $\mathbf{r}_2$ is consumed and another (independent) answer is given, again correct with probability at least $2/3$. After t such trials and the consumption of random strings $\mathbf{r}_1, \ldots, \mathbf{r}_t$, P produces t answers. We expect at least $2/3$ of these to be correct, and Chernoff bounds (with $\delta = \Theta(1)$

and $\mu = \Theta(t)$) imply that at least 60% of these answers are correct with probability at least $1 - \exp\{-\Theta(t)\}$.

We continue the thought experiment by taking a Union Bound over the $2^n \cdot 2^n = 2^{2n}$ choices of the input $(\mathbf{x}, \mathbf{y})$. With probability at least $1 - 2^{2n} \cdot \exp\{-\Theta(t)\}$ over the choice of $\mathbf{r}_1, \ldots, \mathbf{r}_t$, for every input $(\mathbf{x}, \mathbf{y})$, running the protocol P with these random strings yields at least $.6t$ (out of t) correct answers. In this event, the single sequence $\mathbf{r}_1, \ldots, \mathbf{r}_t$ of random strings "works" simultaneously for all inputs $(\mathbf{x}, \mathbf{y})$. Provided we take $t = cn$ with a large enough constant c, this probability is positive. In particular, such a set $\mathbf{r}_1, \ldots, \mathbf{r}_t$ of random strings exist.

Here is the private-coin protocol.

(0) Before receiving their inputs, Alice and Bob agree on a set of strings $\mathbf{r}_1, \ldots, \mathbf{r}_t$ with the property that, for every input $(\mathbf{x}, \mathbf{y})$, running P t times with the random strings $\mathbf{r}_1, \ldots, \mathbf{r}_t$ yields at least 60% correct answers.

(1) Alice picks an index $i \in \{1, 2, \ldots, t\}$ uniformly at random and sends it to Bob. This requires $\approx \log_2 t = \Theta(\log n)$ bit of communication (recall $t = \Theta(n)$).

(2) Alice and Bob simulate the private-coin protocol P as if they had public coins given by $\mathbf{r}_i$.

By the defining property of the $\mathbf{r}_i$'s, this (private-coin) protocol has error 40%. As usual, this can be reduced to 1/3 via a constant number of independent repetitions followed by taking the majority answer. The resulting communication cost is $O(c + \log n)$, as claimed. ■

We stated and proved Theorem 3.1 for general protocols, but the exact same statement holds (with the same proof) for the one-way protocols that we studied in Lectures #1–3.

3.3 Distributional Complexity

Randomized protocols are significantly harder to reason about than deterministic ones. For example, we've seen that a deterministic protocol can be thought of as a partition of the input space into rectangles. A randomized protocol is a distribution over such partitions. While a deterministic protocol that computes a function f induces only monochromatic rectangles, this does not hold for randomized protocols (which can err with some probability).

We can make our lives somewhat simpler by using Yao's Lemma to translate distributional lower bounds for deterministic protocols to worst-case lower bounds for randomized protocols. Recall the lemma from Lecture #2.

Lemma 3.2 (Yao [9]) *Let D be a distribution over the space of inputs $(\mathbf{x}, \mathbf{y})$ to a communication problem, and $\epsilon \in (0, \frac{1}{2})$. Suppose that every deterministic protocol P with*

$$\mathbf{Pr}_{(\mathbf{x},\mathbf{y})\sim D}[P \text{ wrong on } (\mathbf{x}, \mathbf{y})] \leq \epsilon$$

has communication cost at least k. Then every (public-coin) randomized protocol R with (two-sided) error at most ϵ on every input has communication cost at least k.

We proved Lemma 3.2 in Lecture #2 for one-way protocols, but the same proof holds verbatim for general communication protocols. Like in the one-way case, Lemma 3.2 is a "complete" proof technique — whatever the true randomized communication complexity, there is a hard distribution D over inputs that can in principle be used to prove it (recall the Exercises).

Summarizing, proving lower bounds for randomized communication complexity reduces to:

1. Figuring out a "hard distribution" D over inputs.
2. Proving that every low-communication deterministic protocol has large error w.r.t. inputs drawn from D.

Of course, this is usually easier said than done.

3.4 Case Study: Disjointness

3.4.1 Overview

We now return to the Disjointness problem. In Lecture #2 we proved that the *one-way* randomized communication complexity of this problem is linear ($\Omega(n)$). We did this by reducing Index to Disjointness— the former is just a special case of the latter, where one player has a singleton set (i.e., a standard basis vector). We used Yao's Lemma (with D the uniform distribution) and a counting argument (about the volume of small-radius balls in the Hamming cube, remember?) to prove that the one-way randomized communication complexity of Index is $\Omega(n)$. Unfortunately, for general communication protocols, the communication complexity of Index is obviously $O(\log n)$ — Bob can just send his index i to Alice using $\approx \log_2 n$ bits, and Alice can compute the function. So, it's back to the drawing board.

The following is a major and useful technical achievement.

Theorem 3.3 ([4, 7]) *The randomized communication complexity of* Disjointness *is* $\Omega(n)$.

Theorem 3.3 was originally proved in [4]; the simplified proof in [7] has been more influential. More recently, all the cool kids prove Theorem 3.3 using "information complexity" arguments; see [3].

If you only remember one result from the entire field of communication complexity, it should be Theorem 3.3. The primary reason is that the problem is unreasonably effective for proving lower bounds for other algorithmic problems — almost every subsequent lecture will include a significant example. Indeed, many algorithm designers simply use Theorem 3.3 as a "black box" to prove lower bounds for other problems, without losing sleep over its proof.[5,6] As a bonus, proofs of Theorem 3.3 tend to showcase techniques that are reusable in other contexts.

[5]Similar to, for example, the PCP Theorem and the Parallel Repetition Theorem in the context of hardness of approximation (see e.g. [1]).

[6]There's no shame in this — life is short and there's lots of theorems that need proving.

For a trivial consequence of Theorem 3.3 — see future lectures for less obvious ones — let's return to the setting of streaming algorithms. Lectures #1 and #2 considered only one-pass algorithms. In some contexts, like a telescope that generates an exobyte of data per day, this is a hard constraint. In other settings, like database applications, a small constant number of passes over the data might be feasible (as an overnight job, for example). Communication complexity lower bounds for one-way protocols say nothing about two-pass algorithms, while those for general protocols do. Using Theorem 3.3, all of our $\Omega(m)$ space lower bounds for 1-pass algorithms become $\Omega(m/p)$ space lower bounds for p-pass algorithms, via the same reductions.[7] For example, we proved such lower bounds for computing F_∞, the highest frequency of an element, even with randomization and approximation, and for computing F_0 or F_2 exactly, even with randomization.

So how would one go about proving Theorem 3.3? Recall that Yao's Lemma reduces the proof to exhibiting a hard distribution D (a bit of dark art) over inputs and proving that all low-communication deterministic protocols have large error with respect to D (a potentially tough math problem). We next discuss each of these steps in turn.

3.4.2 Choosing a Hard Distribution

The uniform distribution over inputs is not a hard distribution for DISJOINTNESS. What is the probability that a random input $(\mathbf{x}, \mathbf{y})$ satisfies $f(\mathbf{x}, \mathbf{y}) = 1$? Independently in each coordinate i, there is a 25% probability that $x_i = y_i = 1$. Thus, $f(\mathbf{x}, \mathbf{y}) = 1$ with probability $(3/4)^n$. This means that the zero-communication protocol that always outputs "not disjoint" has low error with respect to this distribution. The moral is that a hard distribution D must, at the very least, have a constant probability of producing both "yes" and "no" instances.

The next idea, motivated by the Birthday Paradox, is to define D such that each of Alice and Bob receive a random subset of $\{1, 2, \ldots, n\}$ of size $\approx \sqrt{n}$. Elementary calculations show that a random instance $(\mathbf{x}, \mathbf{y})$ from D has a constant probability of satisfying each of $f(\mathbf{x}, \mathbf{y}) = 1$ and $f(\mathbf{x}, \mathbf{y}) = 0$.

An obvious issue with this approach is that there is a trivial deterministic protocol that uses $O(\sqrt{n} \log n)$ communication and has zero error: Alice (say) just sends her whole input to Bob by describing each of her $\sqrt{n}$ elements explicitly by name ($\approx \log_2 n$ bits each). So there's no way to prove a linear communication lower bound using this distribution. Babai et al. [2] prove that one can at least prove a $\Omega(\sqrt{n})$ communication lower bound using this distribution, which is already quite a non-trivial result (more on this below). They also showed that for every product distribution D — meaning whenever the random choices of $\mathbf{x}$ and of $\mathbf{y}$ are independent — there is a zero-error deterministic protocol that uses only $O(\sqrt{n} \log n)$ bits of communication (see the Exercises).[8]

[7] A p-pass space-s streaming algorithm S induces a communication protocol with $O(ps)$ communication, where Alice and Bob turn their inputs into data streams, repeatedly feed them into S, repeatedly sending the memory state of S back and forth to continue the simulation.

[8] This does not imply that a linear lower bound is impossible. The proof of the converse of Lemma 3.2 — that a tight lower bound on the randomized communication complexity of a problem can always be proved through a distributional lower bound for a suitable choice of D — generally makes use of distributions in

Summarizing, if we believe that DISJOINTNESS really requires $\Omega(n)$ communication to solve via randomized protocols, then we need to find a distribution D that meets all of the following criteria.

1. There is a constant probability that $f(\mathbf{x}, \mathbf{y}) = 1$ and that $f(\mathbf{x}, \mathbf{y}) = 0$. (Otherwise, a constant protocol works.)

2. Alice and Bob need to usually receive inputs that correspond to sets of size $\Omega(n)$. (Otherwise, one player can explicitly communicate his or her set.)

3. The random inputs $\mathbf{x}$ and $\mathbf{y}$ are correlated. (Otherwise, the upper bound from [2] applies.)

4. It must be mathematically tractable to prove good lower bounds on the error of all deterministic communication protocols that use a sublinear amount of communication.

Razborov [7] proposed a distribution that obviously satisfies the first three properties and, less obviously, also satisfies the fourth. It is:

1. With probability 75%:

   (a) $(\mathbf{x}, \mathbf{y})$ is chosen uniformly at random subject to:

      i. $\mathbf{x}, \mathbf{y}$ each have exactly $n/4$ 1's;
      ii. there is no index $i \in \{1, 2, \ldots, n\}$ with $x_i = y_i = 1$ (so $f(\mathbf{x}, \mathbf{y}) = 1$).

2. With probability 25%:

   (a) $(\mathbf{x}, \mathbf{y})$ is chosen uniformly at random subject to:

      i. $\mathbf{x}, \mathbf{y}$ each have exactly $n/4$ 1's;
      ii. there is exactly one index $i \in \{1, 2, \ldots, n\}$ with $x_i = y_i = 1$ (so $f(\mathbf{x}, \mathbf{y}) = 0$).

Note that in both cases, the constraint on the number of indices i with $x_i = y_i = 0$ creates correlation between the choices of $\mathbf{x}$ and $\mathbf{y}$.

3.4.3 Proving Error Lower Bounds via Corruption Bounds

Even if you're handed a hard distribution over inputs, there remains the challenging task of proving a good error lower bound on low-communication deterministic protocols. There are multiple methods for doing this, with the *corruption method* being the most successful one so far. We outline this method next.

At a high level, the corruption method is a natural extension of the covering arguments of Section 2 to protocols that can err. Recall that for deterministic protocols, the covering approach argues that every covering of the matrix $M(f)$ of the function f by monochromatic

which the choices of $\mathbf{x}$ and $\mathbf{y}$ are correlated.

rectangles requires a lot of rectangles. In our examples, we only bothered to argue about the 1-inputs of the function.[9] We'll do something similar here, weighted by the distribution D and allowing errors — arguing that there's significant mass on the 1-inputs of f, and that a lot of nearly monochromatic rectangles are required to cover them all.

Precisely, suppose you have a distribution D over the inputs of a problem so that the "1-mass" of D, meaning $\mathbf{Pr}_{(\mathbf{x},\mathbf{y})\sim D}[f(x,y)=1]$, is at least a constant, say .5. The plan is to prove two properties.

(1) For every deterministic protocol P with error at most a sufficiently small constant ϵ, at least 25% of the 1-mass of D is contained in "almost monochromatic 1-rectangles" of P (defined below). We'll see below that this is easy to prove in general by an averaging argument.

(2) An almost monochromatic 1-rectangle contains at most 2^{-c} mass of the distribution D, where c is as large as possible (ideally $c = \Omega(n)$). This is the hard step, and the argument will be different for different functions f and different input distributions D.

If we can establish (1) and (2), then we have a lower bound of $\Omega(2^{-c})$ on the number of rectangles induced by P, which proves that P uses communication $\Omega(c)$.[10]

Here's the formal definition of an *almost monochromatic 1-rectangle (AM1R)* $R = A \times B$ of a matrix $M(f)$ with respect to an input distribution D:

$$\mathbf{Pr}_{(\mathbf{x},\mathbf{y})\sim D}[(\mathbf{x},\mathbf{y}) \in R \text{ and } f(\mathbf{x},\mathbf{y})=0] \leq 8\epsilon \cdot \mathbf{Pr}_{(\mathbf{x},\mathbf{y})\sim D}[(\mathbf{x},\mathbf{y}) \in R \text{ and } f(\mathbf{x},\mathbf{y})=1]. \quad (2)$$

Here's why property (1) is true in general. Let P be a deterministic protocol with error at most ϵ with respect to D. Since P is deterministic, it partitions the matrix $M(f)$ into rectangles, and in each rectangle, P's output is constant. Let $R_1, \ldots, R_\ell$ denote the rectangles in which P outputs "1."

At least 50% of the 1-mass of D — and hence at least 25% of D's overall mass — must be contained in $R_1, \ldots, R_\ell$. For if not, on at least 25% of the mass of D, $f(\mathbf{x},\mathbf{y}) = 1$ while P outputs "0". This contradicts the assumption that P has error ϵ with respect to D (provided $\epsilon < .25$).

Also, at least 50% of the mass in $R_1, \ldots, R_\ell$ must lie in AM1Rs. For if not, using (2) and the fact that the total mass in $R_1, \ldots, R_\ell$ is at least .25, it would follow that D places more than $8\epsilon \cdot .125 = \epsilon$ mass on 0-inputs in $R_1, \ldots, R_\ell$. Since P outputs "1" on all of these inputs, this contradicts the assumption that P has error at most ϵ with respect to D. This completes the proof of step (1), which applies to every problem and every distribution D over inputs with 1-mass at least .5.

Step (2) is difficult and problem-specific. Babai et al. [2], for their input distribution D over DISJOINTNESS inputs mentioned above, gave a proof of step (2) with $c = \Omega(\sqrt{n})$, thus

[9] Since f has only two outputs, it's almost without loss to pick a single output $z \in \{0,1\}$ of f and lower bound only the number of monochromatic rectangles needed to cover all of the z's.

[10] Why call it the "corruption method"? Because the argument shows that, if a deterministic protocol has low communication, then most of its induced rectangles that contain 1-inputs are also "corrupted" by lots of 0-inputs — its rectangles are so big that (2) fails. In turn, this implies large error.

giving an $\Omega(\sqrt{n})$ lower bound on the randomized communication complexity of the problem. Razborov [7] gave, for his input distribution, a proof of step (2) with $c = \Omega(n)$, implying the desired lower bound for DISJOINTNESS. Sadly, we won't have time to talk about these and subsequent proofs (as in [3]); perhaps in a future course.

References

[1] S. Arora and C. Lund. Hardness of approximations. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 10, pages 399–446. PWS Publishing Company, 1997.

[2] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 337–347, 1986.

[3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 209–218, 2002.

[4] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[5] T. Lee and A. Shraibman. Lower bounds in communication complexity. *Foundations and Trends in Theoretical Computer Science*, 3(4):263–399, 2009.

[6] I. Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39:67–71, 1991.

[7] A. A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

[8] A. C.-C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

[9] A. C.-C. Yao. Lower bounds by probabilistic arguments. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 420–428, 1983.