Notes on Communication Complexity

Anil Ada

Random variables are denoted with boldface letters, not necessarily capital. If $\mathbf{x}$ is a random variable and μ a distribution, $\mathbf{x} \sim \mu$ means that $\mathbf{x}$ is distributed according to μ. The notation $\mathbf{E}[\cdot]$ and $\mathbf{Pr}[\cdot]$ is used for expectation and probability respectively. When the random variable(s) and the distribution(s) are clear from the context, the expectations and the probabilities do not have any subscripts, e.g. $\mathbf{E}[f(\mathbf{x})]$. If the distribution is clear but we would like to explicitly point out the random variables, we put the random variables as subscript, e.g. $\mathbf{E}_{\mathbf{x}}[f(\mathbf{x})]$. We also sometimes choose to make the distribution explicit in this notation, e.g. $\mathbf{E}_{\mathbf{x}\sim\mu}[f(\mathbf{x})]$. The uniform distribution is always denoted by U and the underlying set will always be clear from the context.

1 2 Player Deterministic Model

The most basic and fundamental model in communication complexity is the 2 player deterministic model (introduced in [Yao79]). The setting is as follows. We have a function $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ and two players Alice and Bob. In these notes, we'll assume that $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$ and $\mathcal{Z} = \{1,-1\}$. Alice gets $x \in \mathcal{X}$ and Bob gets $y \in \mathcal{Y}$. They want to collaboratively compute $F(x,y)$ by communicating with each other. Their communication consists of bits that are being transferred from one player to the other. They carry out this communication according to a **protocol** that they have agreed upon beforehand. More precisely, the protocol tells each player:

1. Whose turn it is to send a bit; the protocol determines this purely based on the communicated bits thus far, and we assume without loss of generality that Alice sends the first bit.

2. What bit to send; the protocol determines this based on the communicated bits thus far as well as the input of the player sending the bit.

The protocol also determines when communication stops and the value of the output based on the whole transcript of the communicated bits (which implies both players know the output at the end). The resource of interest is the number of communicated bits, or in other words, the length of the transcript. The goal is to compute the function with the shortest transcript possible. It is worth explicitly noting that we put no restriction on the computational capacities of Alice and Bob, and the sole interest is in the number of bits needed to communicate in order to compute the function.

Let P denote a protocol that correctly computes a function F. Denote by $\Pi_P(x,y)$ the **transcript** of protocol P for the input (x,y) (i.e. the sequence of communicated bits). The cost of P is

$$\text{cost}(P) \stackrel{\text{def}}{=} \max_{(x,y)\in\mathcal{X}\times\mathcal{Y}} |\Pi_P(x,y)|.$$

The **deterministic communication complexity** of F, denoted $\mathbf{D}(F)$, is the cost of the most efficient protocol that computes F correctly. That is,

$$\mathbf{D}(F) \stackrel{\text{def}}{=} \min_{\text{protocol } P \text{ that computes } F} \text{cost}(P).$$

Unless explicitly stated otherwise (for example, we will do so in Chapter **??**), we deal with the standard setting of $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$ and $\mathcal{Z} = \{1,-1\}$, and we are interested in how fast $\mathbf{D}(F)$ grows as a function of n. Observe that every function can be trivially computed with $n+1$ bits of communication: Alice sends x to Bob, Bob computes $F(x,y)$ and sends the result back to Alice. Hence for any F:

$$0 \le \mathbf{D}(F) \le n+1.$$

In view of this, protocols of cost at most poly-log(n) are considered to be efficient and protocols of larger cost are deemed inefficient. As an example of an efficient protocol, suppose we want to determine if the majority of the bits in x *and* y is 1, i.e. is $|x|+|y| \ge n$? This function can be computed using $\lceil \log n \rceil + 1$ bits since Bob can compute the output if Alice sends him $|x|$. A canonical example of a hard function is the *equality* function which evaluates to -1 if and only if $x = y$. Intuitively one expects that for Alice and Bob to be sure that $x = y$, or detect a difference, they would have to compare x_i and y_i for all $i \in [n]$. That is, our intuition tells us that $\mathbf{D}(\text{EQUALITY}) \ge n$. But is this correct, and if it is, how do we formally prove it?

In order to prove lower bounds on communication complexity, we need to have a combinatorial understanding of what protocols do. To this end, we first observe that a protocol can be conveniently described with a binary tree as follows (see Figure 1). Each node v of the tree is labelled with the letter A or B (indicating whether the node belongs to Alice or Bob) and a function f_v. This function is of the form $f_v : \mathcal{X} \to \{0,1\}$ if the label is A or it is of the form $f_v : \mathcal{Y} \to \{0,1\}$ if the label is B, and it determines what bit the corresponding player communicates. Let us trace the behaviour of the protocol to understand the meaning of this tree. As always, Alice gets x and Bob gets y. First, without loss of generality, the root r is always labelled A, which means that Alice is the first to communicate a bit. Then the protocol determines what bit Alice will send by evaluating $f_r(x)$, i.e. Alice sends Bob $f_r(x)$. If $f_r(x)$ is 0, we move to the left child of the root and if $f_r(x) = 1$ we move to the right child. Without loss of generality let's assume we are at the right child, which we denote by v. If v is labelled with A, then it is again Alice's turn to speak. If it is labelled B, it is Bob's turn. And as before, the function f_v tells the player what bit to send. In this fashion we make our way down the tree until we reach a leaf node. Leaf nodes are special and they determine the output of the protocol.

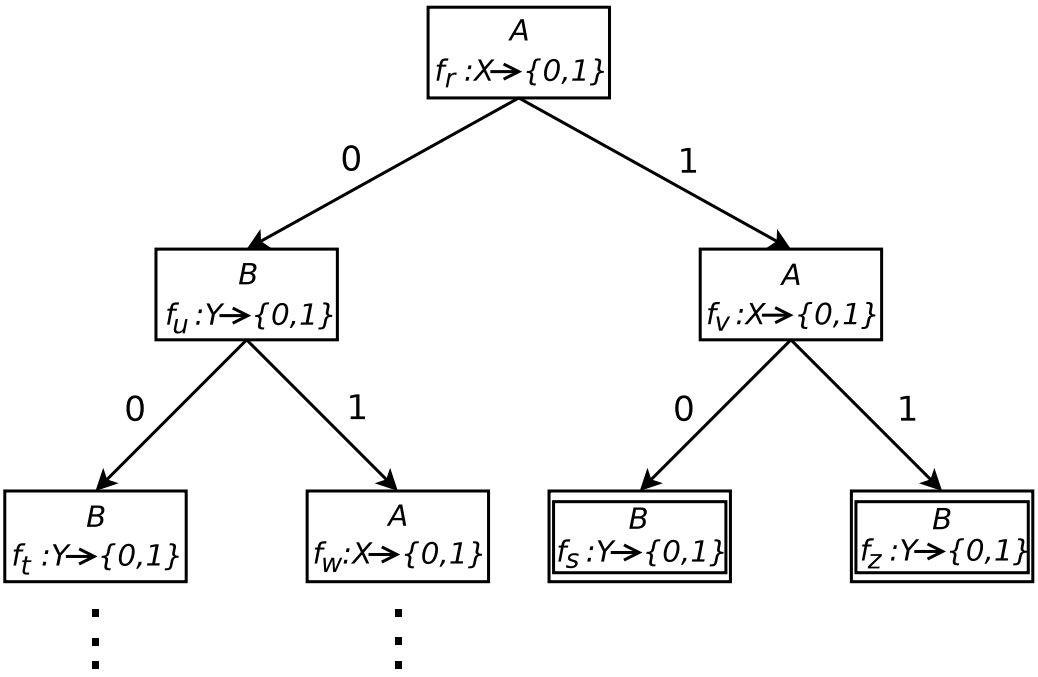

Figure 1: A binary tree representing a protocol. Each node is labelled with A or B to indicate whose turn it is to speak. A function associated with a node tells the player what to send. Depending on whether 0 or 1 is sent, we move to the left or the right child of the node. The leaf nodes are indicated with double lines. The functions associated with them determine the output of the protocol.

Observe that every protocol can be described with such a tree and this tree description is entirely consistent with the description we provided in the beginning. In particular, whose turn it is to speak is determined based only on the communicated bits thus far and what a player sends is determined by the communicated bits as well as the input of the player. Obviously the cost of the protocol is the height of the tree.

With this point of view, we will be able to gain a very good understanding of what a protocol does when computing a function F. First we represent F by a $|\mathcal{X}| \times |\mathcal{Y}|$ matrix M_F where the rows are labelled with $x \in \mathcal{X}$, columns are labelled with $y \in \mathcal{Y}$, and $M_F[x, y] = F(x, y)$. A submatrix $\mathcal{S} \times \mathcal{T}$ where $\mathcal{S} \subseteq \mathcal{X}$ and $\mathcal{T} \subseteq \mathcal{Y}$ is called a **rectangle**. The rectangle is said to be **monochromatic** if M_F restricted to $\mathcal{S} \times \mathcal{T}$ has the same value on all of its entries. We will now see that a protocol of cost c that computes F *partitions*[1] M_F into at most 2^c monochromatic rectangles. In fact, this is the most important property of a protocol and all lower bound techniques will be based on this observation.

Proposition 1.1. *Let P be a protocol that computes $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ with at most c bits of communication. Then P induces a partition of M_F into at most 2^c monochromatic rectangles.*

To see why this is the case, let's trace once again the behaviour of the protocol down the associated tree. We start at the root which is labelled with A. The root corresponds to the whole matrix $\mathcal{X} \times \mathcal{Y}$. The function f_r is boolean and therefore partitions $\mathcal{X}$ into two sets $\mathcal{X}_0$ and $\mathcal{X}_1$: for all $x \in \mathcal{X}_0$ Alice sends 0 to Bob, and for all $x \in \mathcal{X}_1$ she sends 1. Therefore the left child of r corresponds to the rectangle $\mathcal{X}_0 \times \mathcal{Y}$ and the right child corresponds to $\mathcal{X}_1 \times \mathcal{Y}$. In some sense, if we go to the left child, we eliminate (disregard) the inputs $\mathcal{X}_1 \times \mathcal{Y}$ and our new matrix is $\mathcal{X}_0 \times \mathcal{Y}$ (this is where the input (x, y) lives). If we go to the right child, we eliminate $\mathcal{X}_0 \times \mathcal{Y}$ and our new matrix is $\mathcal{X}_1 \times \mathcal{Y}$. Note that $\mathcal{X}_0 \times \mathcal{Y}$ and $\mathcal{X}_1 \times \mathcal{Y}$ are disjoint. This process inductively continues, so for each node of the tree, there corresponds a rectangle. If a node is the descendent of another, the rectangle of the descendent will be a subset of the other. Otherwise the rectangles are disjoint. Once we reach a monochromatic rectangle, there is no need to partition it further since we can safely declare $F(x, y)$ as the value of this rectangle. Hence each leaf node corresponds to a monochromatic rectangle. Suppose the height of the tree is c, i.e. the protocol has cost c. Then there are at most 2^c leaves. Thus, the protocol partitions M_F into at most 2^c monochromatic rectangles.

It is instructive to see a different proof of the above fact. The following gives an alternative definition of a rectangle.

Proposition 1.2. *A set $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ is a rectangle if and only if for all $(x, y), (x', y') \in \mathcal{R}$, we have $(x, y') \in \mathcal{R}$.*

An important observation is that if a protocol produces the same transcript for (x, y) and (x', y'), i.e. $\Pi(x, y) = \Pi(x', y')$, then $\Pi(x, y) = \Pi(x', y') = \Pi(x, y')$. This

[1]The word *partition* here is important. The rectangles are mutually disjoint and together cover the whole matrix M_F.

implies that all the inputs that produce a particular transcript form a rectangle. There are at most 2^c different transcripts and therefore we have at most 2^c monochromatic rectangles that partition M_F.

Proposition 1.1 immediately suggests a lower bound strategy: to show a function F has high communication complexity, show that no matter how you partition M_F into monochromatic rectangles, you need many rectangles. Let's denote by $C^D(F)$ the minimum number of rectangles in any monochromatic disjoint cover of M_F. The lower bound strategy can be restated as follows.

Corollary 1.3.

$$\mathbf{D}(F) \geq \lceil \log C^D(F) \rceil .$$

With this tool, it is now easy to show $\mathbf{D}(\text{EQUALITY}) \geq n + 1$. The matrix corresponding to the *equality* function is basically the identity matrix: the diagonal elements are -1 and the off-diagonal elements are 1. Observe that no monochromatic rectangle can contain more than one -1 since if a rectangle contains the entries (a, a) and (b, b), then it also has to contain (a, b), which corresponds to a 1 entry. This means that we need at least 2^n rectangles to cover the diagonal elements, plus we need at least one rectangle to cover the 1's in the matrix. So in total we need at least $1 + 2^n$ rectangles and hence $\mathbf{D}(\text{EQUALITY}) \geq \lceil \log(1 + 2^n) \rceil = n + 1$.

Although every protocol that computes F induces a partition of M_F into monochromatic rectangles, simple examples show that the converse is not true. So if some monochromatic partitions do not correspond to any protocol, how tight is Corollary 1.3? The next theorem states that the gap is not very large.

Theorem 1.4.

$$\mathbf{D}(F) \leq O(\log^2 C^D(F)).$$

Let's reiterate that Proposition 1.1 and Corollary 1.3 are the basis for all lower bound techniques in communication complexity, including the randomized model which we will discuss in the next section. In most cases it is not easy to exactly determine $C^D(F)$ so all the various lower bound techniques try to find a suitable lower bound for $C^D(F)$. For instance one might try to upper bound the size of the largest monochromatic rectangle in M_F. If all monochromatic rectangles are small, then we can conclude that we need many rectangles to partition M_F. A more interesting lower bound technique uses the rank of M_F.

Proposition 1.5.

$$\mathbf{D}(F) \geq \log \operatorname{rank} M_F.$$

Proof. Suppose a protocol P of cost c computes F and denote by $\mathcal{S}_1 \times \mathcal{T}_1, \ldots, \mathcal{S}_t \times \mathcal{T}_t$ the t monochromatic rectangles that the protocol induces ($t \leq 2^c$). For each of these rectangles $\mathcal{S}_i \times \mathcal{T}_i$, define the $|\mathcal{X}| \times |\mathcal{Y}|$ matrix $M_{\mathcal{S}_i \times \mathcal{T}_i}$ by

$$M_{\mathcal{S}_i \times \mathcal{T}_i}[x, y] = \begin{cases} M_F[x, y] & \text{if } (x, y) \in \mathcal{S}_i \times \mathcal{T}_i \\ 0 & \text{otherwise} \end{cases}$$

These matrices are like the indicator matrices of the rectangles. Obviously we have $M_F = \sum_{i=1}^{t} M_{\mathcal{S}_i \times \mathcal{T}_i}$. By the subadditivity of the rank, we have $\operatorname{rank} M_F \leq \sum_{i=1}^{t} \operatorname{rank} M_{\mathcal{S}_i \times \mathcal{T}_i}$. Since each $M_{\mathcal{S}_i \times \mathcal{T}_i}$ has rank at most 1, we conclude that $\operatorname{rank} M_F$ is at most $t \leq 2^c$, i.e. $c \geq \log \operatorname{rank} M_F$. □

Arguably the most famous open problem in communication complexity is whether the rank lower bound is close to being tight.

Conjecture 1.6 (Log Rank Conjecture [LS88])**.** *There is some universal constant k such that*

$$\mathbf{D}(F) \leq O(\log^k \operatorname{rank} M_F).$$

Needless to say there are other lower bound techniques and each has its own advantages depending on the particular function we are dealing with.

One of the well-studied restrictions of the deterministic model is called the **simultaneous** model. Here, the players are not allowed to interact with each other. Upon receiving their inputs, the players send a message to an external referee. The referee, who does not see the players' inputs, determines the output based on these messages. The cost is the number of bits sent to the referee and we denote by $\mathbf{D}^{\|}(F)$ the deterministic simultaneous communication complexity of F.

2 Randomized Model

The previous section introduced the most basic communication complexity model. In this section we will introduce the randomized model which has a variety of interesting applications.

A natural way to extend the deterministic model to utilize randomness is to allow each player to privately flip coins and make decisions based on the outcomes of those coin flips. Normally, we have to allow some probability of error in computing the function correctly. To make this more concrete, let's say that Alice has access to a random binary string $\mathbf{r}_A$ and Bob has access to a random binary string $\mathbf{r}_B$. Then a randomized protocol computes F with ϵ error if

$$\forall (x, y) \in \mathcal{X} \times \mathcal{Y}, \quad \mathbf{Pr}\left[F(x, y) \neq P(x, y)\right] \leq \epsilon,$$

where $P(x, y)$ denotes the output of the protocol and the probability is over the random choices of $\mathbf{r}_A$ and $\mathbf{r}_B$. The cost of a randomized protocol is the maximum number of bits communicated, where the maximum is over all inputs and random strings. It is worth making it clear that the random strings being used by the players do not count towards the cost at all. We denote by $\mathbf{R}^{\epsilon}_{\text{pri}}(F)$ the **randomized communication complexity** of F with ϵ-error, i.e. the cost of the most efficient randomized protocol that computes F with ϵ-error (the subscript 'pri' will be clarified shortly). We are mainly interested in the case where $\epsilon < 1/2$ is some constant. The particular choice of the constant does