11 Deterministic Communication Complexity (April 1 and 3)

Consider the following communication problem between two players, Alice and Bob.[1] each hold an n-bit string. They want to agree on the value of some boolean function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where the first argument is Alice's string x, and the second argument is Bob's string y. Both Alice and Bob know the function F, and they've agreed beforehand on a communication protocol, but neither knows the value of the other player's bit string. Obviously, Alice could simply tell Bob what x is, and then Bob could send back the value of $F(x, y)$. But there's a catch—Alice is on the earth and Bob is on the moon.[2] Any communication between the two players is much more expensive than any local computation by either player. How many bits do Alice and Bob need to exchange in order to compute $F(x, y)$?

To convince you that this game is interesting, consider the parity function

$$F(x, y) = \sum_{i=1}^{n} (x_i + y_i) \bmod 2.$$

Alice and Bob can easily compute this function by exchanging only two bits. First, Alice sends the parity of her input string

$$\sum_{i=1}^{n} x_i \bmod 2;$$

Bob then sends back the answer. Note that the bit that Alice transmits is *not* one of the input variables; our model allows either player to compute and transmit arbitrary boolean functions of their inputs and all previous transmissions.

11.1 One-Way Communication

Let's start by considering a simple one-way version of this game: Alice must send Bob a sequence of bits, and then Bob must send back the one-bit answer. We assume that Alice and Bob agree in advance on how many bits Alice will send.

We can identify any function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ with a $2^n \times 2^n$ boolean matrix; at the risk of confusing the reader, I'll also call the matrix F. Every row of this matrix corresponds to a possible input value for Alice; every column represents a possible input value for Bob. For example, if F is the equality function

$$F(x, y) = [x = y] = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y, \end{cases}$$

then F is the identity matrix.

Lemma 1. *In the worst case, Alice must send at least n bits in order for Bob to correctly compute the equality function $F(x, y)$.*

Proof: Every row of the matrix F is different. If Alice transmits k bits, then Bob can only divide the rows of F into $2^k < 2^n$ equivalence classes. In particular, if $k < n$, there are at least two rows x and x' that Bob cannot distinguish. Bob must transmit the same value for the inputs (x, x) and (x', x), so the protocol is incorrect. □

[1] Alice and Bob are fairly common characters in the theoretical computer science mythos. For some reason, I always imagine Alice in Wonderland and J. R. "Bob" Dobbs.

[2] Or Alice is behind the looking glass and "Bob" is on a Yeti spaceship.

This proof introduces the idea of a *fooling pair*: two inputs that cannot be distinguished by a protocol that ends too soon. Since there is an obvious n-bit algorithm—transmit everything—we have the exact one-way communication complexity for the equality function.

The previous lemma is a special case of the following easy result:

Theorem 2. *If F has N distinct rows, then in the worst case, Alice must transmit exactly $\lceil \lg N \rceil$ bits in order for Bob to correctly compute $F(x, y)$.*

11.2 Two-Way Communication

Now suppose Alice and Bob alternately send one bit at a time, following a protocol they've agreed on in advance. Formally, a *protocol* is a pair of functions $P_A, P_B : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}$. Alice and Bob alternately send the following sequence of bits, where x and y are their respective input strings:

$$\begin{aligned} a_1 &= P_A(x, \varepsilon) && [\varepsilon \text{ is the empty string}] \\ b_1 &= P_B(y, a_1) \\ a_2 &= P_A(x, a_1 b_1) \\ b_2 &= P_B(y, a_1 b_1 a_2) \\ &\vdots \end{aligned}$$

We refer to the string of transmitted bits $a_1 b_1 a_2 b_2 \ldots$ as a *transcript* or *crossing sequence* of the protocol for the input strings (x, y). We require that the last bit transmitted is the value of the function $F(x, y)$. Since both players know the protocol in advance, both know when the function value is transmitted.[3]

The *cost* $C(P, (x, y))$ of computing $F(x, y)$ using protocol $P = (P_A, P_B)$ is the number of bits transmitted. As usual, the cost of the protocol is the worst-case cost over all inputs of length n,

$$C(P) = \max_{x,y \in \{0,1\}^n} C(P, (x, y)),$$

and the cost of the function is the minimum cost of any protocol that computes it:

$$C(F) = \min_{P \text{ computes } F} C(P) = \min_{P \text{ computes } F} \max_{x,y \in \{0,1\}^n} C(P, (x, y))$$

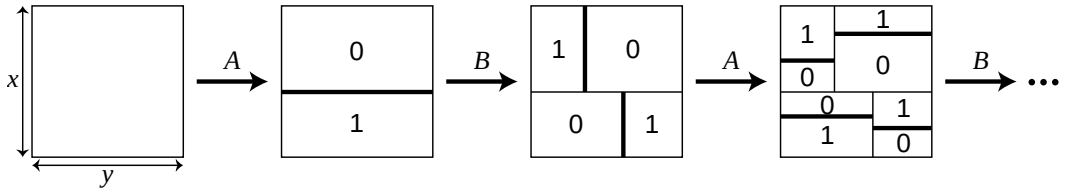
Clearly, $C(F) \leq 2n$ for any function F: Alice can simply transmit her bits to Bob, while Bob transmits back 0s until the very end. (Each player must transmit a bit on their turn, even if they have nothing useful to say.)

$C(F)$ is implicitly a function of n, since the function F has a fixed range. Most of the time, we're *really* interested in the complexity of an infinite sequence of functions $F_1, F_2, F_3, \ldots$, one for each possible input size. The bounds we get will be inherently non-uniform—in principle, we don't care if the protocol for F_{42} resembles the protocol for F_{31337}; on the other hand, we also don't care if the functions themselves are similar!

Recall that we can express the function F as a $2^n \times 2^n$ matrix; the goal is for the two players to determine the value of a particular cell, where initially Alice knows only the row and Bob knows only the column. Any protocol recursively divides F into smaller and smaller blocks, as follows.

[3] Some sources redundantly define the protocol functions as $P_A, P_B : \{0,1\}^n \times \{0,1\}^* \to \{0, 1, \textsc{Accept}, \textsc{Reject}\}$, where transmitting ACCEPT or REJECT ends the protocol.

Alice's first transmission a_1 depends only on the row, so her possible transmissions partition the rows into two subsets. Bob's first transmission b_1 depends both on the column and on a_1, so his possible responses partition the columns of each of Alice's sub-matrices into two subsets.



Any communication protocol recursively decomposes the function matrix.

In general, any successful k-bit protocol partitions the matrix F into 2^k disjoint *monochromatic minors*. A minor is the product of a subset of the rows with a subset of the columns; a minor is monochromatic if all its entries are the same. Note that some of these minors may be empty. Minors are sometimes also suggestively called *rectangles*. Despite the intuition you might get from the figure above, the rows and columns of each minor need not be contiguous. (Think of the parity protocol.)

Let $\Delta(F)$ denote the minimum number of disjoint monochromatic minors needed to cover the matrix F. We immediately have the following lower bound on the communication complexity of F.

Theorem 3. $C(F) \geq \lceil \lg \Delta(F) \rceil$.

11.3 Lower Bounds for $\Delta(F)$

The simplest method to derive lower bounds for $\Delta(F)$ is to look for the largest monochromatic minor. If A is the maximum area of any monochromatic minor in the matrix, we immediately have $\boxed{\Delta(F) \geq 2^{2n}/A}$, which implies that $C(F) \geq 2n - \lfloor \lg A \rfloor$. For example, consider the one-bit parity function $F : \{0,1\} \times \{0,1\} \to \{0,1\}$:

$$F = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

The largest monochromatic minor in this matrix is a single cell, so $\Delta(F) = 4$, which implies that $C(F) = 2$. (Duh. Alice can't just transmit the answer.)

A second method uses an object called a fooling set. A *fooling pair* is a pair of inputs (x, y) and (x', y') such that the function values $F(x, y)$, $F(x, y')$, $F(x', y)$, and $F(x', y')$ are not all equal. Equivalently, (x, y) and (x', y') are a fooling pair if and only if there is no monochromatic minor that contains them both. A *fooling set* is a set of possible inputs

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$$

such that every pair (x_i, y_i) and (x_j, y_j) are a fooling pair. If we parrtition F into monochromatic minors, then each element of any fooling set must lie in a different minor.

Lemma 4. *$\Delta(F)$ is at least the size of the largest fooling set in F.*

For example, consider the n-bit equality function, whose matrix is the $2^n \times 2^n$ identity matrix. The entire diagonal of this matrix constitutes a fooling set of size 2^n, so $\Delta(F) \geq 2^n$, which implies

that $C(F) \geq n$. In other words, at least n bits must be transmitted for Alice and Bob to decide whether their inputs agree.[4]

Finally, a third method for bounding $\Delta(F)$ uses (you guessed it) linear algebra. I'll leave the proof of this theorem as a homework exercise.

Lemma 5. $\Delta(F) \geq \text{rank}(F)$, *where* $\text{rank}(F)$ *denotes the rank of* F *over the rationals.*

11.4 A Similar Upper Bound

We can also derive *upper* bounds on communciation complexity using monochromatic minors. Let $\Omega(F)$ denote the minimum number of *possibly overlapping* monochormatic minors needed to cover F. Note that $\Omega(F)$ can be significiantly smaller than $\Delta(F)$.

Theorem 6 (AUY '83). $C(F) \leq 6 \lg^2 \Omega(F)$.

Proof: I'll describe a protocol that runs in at most $3 \lg \Omega(F)$ phases, where the players transmit at most $2 \lg \Omega(F)$ bits in each phase. Let $\mathcal{M}$ be a minimum monochromatic cover of F. Any minor in $\mathcal{M}$ can be specified by a unique ID consisting of $\lceil \lg \Omega(F) \rceil$ bits. Both $\mathcal{M}$ and the unique IDs of each minor are known to both players.

In the middle of any communication protocol, the *information state* of the two players can be described as a minor $X \times Y$ of the matrix F, where $X \subseteq [2^n]$ is a subeet of the rows and $Y \subset [2^n]$ is a subset of the coolumns—Alice knows that $y \in Y$, and Bob knows that $x \in X$. The protocol ends when this minor becomes monochromatic. Let $X_i \times Y_i$ denote the information state after the first i phases of our protocol. Initailly, we have $X_0 \times Y_0 = [2^n] \times [2^n]$. Let $\mathcal{M}_i = \{A \times B \in \mathcal{M} \mid A \cap X_i \neq \varnothing \neq B \cap Y_i\}$ be the set of minors in $\mathcal{M}$ that are consistent with the information state after phase i.

Alice and Bob each maintain a graph whose vertices are the minors in $\mathcal{M}_i$. Alice's graph $\mathcal{A}_i$ has an edge between any two minors with overlapping rows; Bob's graph $\mathcal{B}_i$ has an edge between any two minors with overlapping columns. Finally, let $\mathcal{C}_i = \mathcal{A}_i \cap \mathcal{B}_i$ be the graph over $\mathcal{M}_i$ with an edge between any two minors that contain a common cell. Neither Alice nor Bob knows $\mathcal{C}_i$.

We say that a minor in $\mathcal{M}_i$ is *good for Alice* if it has degree at most $3|\mathcal{M}_i|/4$ in her graph $\mathcal{A}_i$. Similarly, a minor in $\mathcal{M}_i$ is *good for Bob* if it has degree at most $3|\mathcal{M}_i|/4$ in $\mathcal{B}_i$. Finally, we say that a minor is *bad* if it is good for neither player; every bad minor has degree at least $|M_i|/2$ in $\mathcal{C}_i$.

Phase $i+1$ of the protocol proceeds as follows. If any minor in $\mathcal{M}_i$ is good for Alice, Alice transmits a 1, followed by the unique ID of her good minor; meanwhile, Bob transmits zeros. If Alice has no good minor, she transmits a 0, and if Bob has a good minor, he transmits a 1, followed by the unique ID of his good minor; meanwhile, Alice transmits zeros. If Bob has no good minor, he transmits a 0, and then Alice transmits the color of any minor in $\mathcal{M}_i$, ending the protocol. The total number of bits transmitted during each phase is at most $\max\{3, 2 + 2\lceil \lg \Omega(F) \rceil\}$.

If either Alice or Bob have a good minor, their transmission reduces the size of the information state by a constant factor: $|\mathcal{M}_{i+1}| \leq 3|M_i|/4$. It follows that the algorithm ends after at most $\lceil \log_{4/3} \Omega(F) \rceil < 3 \lg \Omega(F)$ phases.

On the other hand, if every minor is bad, then every minor has degree at least $|M_i|/2$ in $\mathcal{C}_i$. It follows that $\mathcal{C}_i$ is connected; in particular, for every pair of minors in $\mathcal{M}_i$, either they overlap, or there is a third minor that overlaps both of them. But any two overlapping monochromatic minors

[4]This is off by a factor of 2 from the tirivial protocol. Seth points out that there is a protocol of complexity $n+2$: Alice transmits the first half of her bits, Bob transmits the second half of his bits, Alice reports whether the first halves are equal, and Bob responds with the final answer.

must have the same color. We conclude that every minor in $\mathcal{M}_i$ has the same color as the input cell (x, y) (which must be contained in at least one minor in $\mathcal{M}_i$), so Alice's final transmission is correct. □

Theorems 3 and 6 give us the following very weak relation between $\Delta(F)$ and $\Omega(F)$:

Corollary 7. $\Delta(F) \leq \Omega(F)^{6 \lg \Omega(F)}$.

Intuitively, $\Omega(F)$ is related to the *certificate complexity* of the function F—to prove that $F(x, y)$ is correct, the players only need to identify the minor containing (x, y) in some minimum monochromatic cover.

More generally, we can define a *non-deterministic* protocol as follows. Both players, in addition to their input strings, have access to a common *proof string*. Based on their individual input and the proof string, each player transmits one bit. The protocol is successful if both players transmit the correct output value $F(x, y)$. The model of computation requires that it is impossible for *both* players to transmit the incorrect output value, no matter waht proof string is presented, although the players may disagree. The *complexity* of the protocol is the number of bits in the shortest successful proof string.

Given a cover of F by monochromatic minors, wee can take the identifier of any minor containing (x, y) as the proof string; thus, there is a non-deterministic protocool with complexity $\lceil \lg \Omega(F) \rceil$. On the other hand, any non-detemrinistic protocol determines a cover of F by monochromatic minors, which implies that every non-deterministic protocol has complexity at least $\lceil \lg \Omega(F) \rceil$.

Thus, both communciation complexity and decision-tree complexity share the following common feature: Up to constant factors, the deterministic complexity is at most quadratic in the non-deterministic complexity!

11.5 Space-Time Tradeoffs for Turing Machines (!)

At the beginning of the course, I introduced concrete complexity as a way to avoid the tremendous difficulty of proving lower bounds in universal models of computation like Turing machines. Even for very well-behaved problems like sorting, very little is known about Turing machine complexity.

However, using communication complexity results, it is possible to prove lower bounds for very simple problems in the *one-head, one-tape* Turing machine model. This version of the Turing machine has a read-only input tape of length n, with a single read head, and a work tape of unbounded length, with a single read/write head. The *time* taken by the Turing machine is as usual the number of state transitions; the *space* is the number of cells of the work tape that the machine accesses.

Theorem 8. *Suppose there is a one-tape Turing machine that recognizes the language* $L = \{x\#^{|x|}x \mid x \in \{0,1\}^*\}$ *in time* $T(n)$ *and space* $S(n)$. *Then* $S(n) \cdot T(n) = \Omega(n^2)$.

Proof: Let M be a Turing machine that recognizes L in time $T(n)$ using space $S(n)$. Suppose M has q states. We can construct from M a communication protocol for the n-bit equality function with complexity $O(S(n)T(n)/n)$ as follows.

Given strings x and y of length n, Alice and Bob will simulate M with the input string $x\#^n y$, each using only their own portion of the string. Alice begins by simulating M on the input string $x\#^n?$. As soon as the input read head moves to the ? character, Alice transmits the current state ($\lg q = O(1)$ bits) and the contents of the work tape ($S(n)$ bits) to Bob. Now Bob continues the simulation as though the input were $?\#^n y$, starting the read head at the first symbol of y. When

Bob's simulation reads the ?, Bob sends the current state and the contents of the work tape back to Alice. If the Turing machine accepts, then Alice and Bob must have the same string.

The Turing machine must take at least n steps between any pair of transmissions. Thus, the total numebr of transmissions is at most $T(n)/n$. Each transmission consists of $S(n) + O(1)$ bits. Thus, the total number of bits transmitted is $O(S(n)T(n)/n)$. On the other hand, we know that any protocol for the n-bit equality function requires at least n bits to be transmitted. It follows that $n = O(S(n)T(n)/n)$, or equivalently, $S(n)T(n) = \Omega(n^2)$. □

11.6 Multi-Party Communication Complexity (aka "Indian Poker")

There is a card game called "Indian Poker" where each player sticks a playing card to his forehead, so that each player can see all the cards but his own. Based on what he can see and how the other players bet, each player bets on whether he has the highest card, or lowest card, or the median card, or whatever. Usually beer is involved.

Two-party communication complexity can be seen as a game of Indian poker between Alice and Bob, where the cards are n-bit strings. Here I'll consider the k-player version of this game. Each player P_i has an associated input string x_i, drawn from some finite set N. Initially, each player knows all the input strings except his own. The goal of the game is to compute a function $F(x_1, x_2, \ldots, x_k)$ by broadcasting bits in a round-robin fashion. Thus, the following sequence of bits is broadcast, for some *protocol* function $P : N^{k-1} \times \{0,1\}^* \to \{0,1\}$:

$$\begin{aligned}
w_1 &= P(x_2, x_3, x_4, \ldots, x_k;\, \varepsilon) \\
w_2 &= P(x_1, x_3, x_4, \ldots, x_k;\, w_1) \\
w_3 &= P(x_1, x_2, x_4, \ldots, x_k;\, w_1 w_2) \\
&\vdots \\
w_i &= P(x_1, \ldots, \widehat{x}_{i \bmod k}, \ldots, x_k;\, w_1 w_2 \cdots w_{i-1}) \\
&\vdots
\end{aligned}$$

(Here, the hat means "omit this".) As in the two-party case, we require that the last bit transmitted is $F(x_1, x_2, \ldots, x_k)$, and the complexity of the protocol is th enumber of bits transmitted in the worst case. The string of transmitted bits $w(x_1, x_2, \ldots, x_k) = w_1 w_2 w_3 \ldots$ is called the *transcript* of the protocol for that particular input.

For example, consider the "exactly n" function F_n. Here, we have $N = [n] = \{1, 2, \ldots, n\}$, and

$$F_n(x_1, x_2, \ldots, x_k) = 1 \iff \sum_{i=1}^{k} x_i = n.$$

Theorem 9 (Chandra, Furst, Lipton, 1983). *For any fixed k, the complexity of the exactly-n function goes to infinity as n goes to infinity.*

Before we get to the proof, let's look at some easy special cases. First, suppose $n = 2$. The matrix associated with the "'exactly-n" function is all zeros, except for a back-diagonal of $n - 1$ ones. Since any pair of 1s forms a fooling pair, we have $C(F_2) \geq \lceil \lg(n-1) \rceil$. On the other hand, there is a trivial protocol of complexity $2\lceil \lg n \rceil$: Alice sends her number Bob (while he sends 0s) and Bob replies with the answer. We can reduce this to $\lceil \lg n \rceil + 3$ using Seth's trick: Alice sends the low bits, Bob sends the high bits, Alice sends a confirmation bit and a carry bit, and Bob replies with the final answer.

What about $k = 3$? In this case, the function we want to collectively can be represented by an $n \times n \times n$ matrix of zeros and ones. Each player knows *two* of the coordinates of the input cell (x, y, z). For the exactly-n function, the matrix is all zeros, except for a diagonal 'triangle' of $\binom{n}{2}$ ones. It's not hard to show that the complexity of the three-player exactly n function is at most $O(\sqrt{\lg n})$.

To prove lower bounds in the multi-party setting, we need the following generalization of fooling sets. A *forbidden k-pattern* for a function $F : N^k \to \{0, 1\}$ is a collection of $k + 1$ inputs $x, y^1, y^2, \ldots, y^k \in N^k$ such that $F(y^i) = 1$ for all i, and x differs from each y^i exactly in the ith component. For example, if $k = 3$ and $n = 17$, the following inputs form a forbidden 3-pattern for the exactly-17 function:

$$x = (3, 5, 6) \qquad y^1 = (6, 5, 6) \qquad y^2 = (3, 8, 6) \qquad y^3 = (3, 5, 9)$$

The following lemma is easy to prove by induction on the numebr of transmitted bits:

Lemma 10. *Let $x, y^1, y^2, \ldots, y^k$ constitute a forbidden k-pattern for F. For any protocol for F, if $w(y^i) = w(y^j)$ for all i and j, then $w(x) = w(y^i)$ for all i.*

To prove our general bounds, let's fix n and k. Let $H = \{(x_1, x_2, \ldots, x_k) \in [n]^k \mid \sum_{i=1}^{n} x_i\}$ be the set of inputs for which to correct anser is 1, and let R be the minimum number of colors needed to color H so that no forbidden k-pattern is monochromatic.

Theorem 11. $\lceil \lg R \rceil \le C(\textit{exactly-}n) \le \lceil \lg R \rceil + k$

Proof: The lower bound is easier. Take any protocol of length t. Color each input $y \in H$ with the transcript $w(y)$. The number of possible colors is at most 2^t. Lemma 10 implies that no forbidden k-pattern is monochromatic, so $2^t \ge R$, or equivalently, $t \ge \lceil \lg R \rceil$.

For the upper bound, assume that all k players agree in advance on a minimal coloring for H such that no forbidden k-pattern is monochromatic, as well as a labeling for the R colors. Let $x_i \in [n]$ denote player i's input; recall that this is the only input that player i does *not* know. Let y^i denote the only input in H that's consistent with the information available to player i, that is,

$$y^i = (x_1, x_2, \ldots, x_{i-1}, x_i^*, x_{i+1}, \ldots, x_k) \in [n]^k$$

where

$$x_i^* = n - \sum_{j \ne i} x_i.$$

(If $x^* < 1$ or $x^* > n$, then y^i is undefined.)

The protocol proceeds in two phases. In the ith transmission of the first phase, player $i \bmod k$ transmits the ith bit of the color of $z^{i \bmod k}$. At the end of this phase, the players have specified a putative color C for the input x. In the second phase, the players immedaitely reject. Otherwise, each player announces whether the color of y^i agrees with C; the input is accepted if and only if everyone agrees. The total number of transmitted bits is at most $\lceil \lg R \rceil + k$.

If the actual input $x = (x_1, x_2, \ldots, x_k)$ is in H, then $y^i = x$ for all i, and everyone's color will agree. Otherwise, the inputs $x, y^1, y^2, \ldots, y^k$ form a forbidden k-pattern, so by assumption, some pair y^i and y^j have different colors, so the players will not agree in the second phase. □

Finally, to prove Theorem 9, we pull a rabbit our of our hat.

Lemma 12 (Gallai). *For any fixed k, let $R(n)$ denote the minimum number of colors need to color $H(n)$ so that no forbidden k-pattern is monochromatic. Then $R(n) \to \infty$ as $n \to \infty$.*

The proof of this lemma uses Ramsey theory, which is a formalization of the statement "complete disorder is impossible": if there are too few colors, then some forbidden k-pattern *must* be monochromatic. Nobody knows the asymptotic growth rate of $R(n)$; Gallai's bound is very small.