Introduction

THE CONCEPT OF A CONVERSATION is universal. In this book, we develop methods to determine the most efficient conversations for specific tasks. We start by exploring some examples that illustrate how such conversations arise, and why they are worthy of study.

We begin the story with one of the earliest applications of communication complexity: proving lower bounds on the area required for digital chips.[1] A chip design specifies how to compute a function $f(x_1, \ldots, x_n)$ by laying out the components of the chip on a flat grid, as in Figure I.1. Each component either stores one of the inputs to the function, or performs some computation on the values coming from adjacent components. It is vital to minimize the area used in the design because this affects the cost, power consumption, reliability, and speed of the chip.

[1] Thompson, 1979.

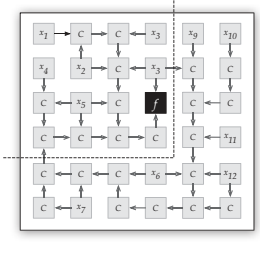

Figure I.1 Any chip can be broken into two pieces while cutting few wires.

Because there are n inputs, we need area of at least n to compute functions that depend on all of their inputs. Can we always find chip designs with area proportional to n? The framework of communication complexity can be used to show that many functions require area proportional to n^2, *no matter what chip design is used*!

The crucial insight is that the layout of the chip yields a conversation whose outcome is the value of f. If a chip design has area A, one can argue that there must be a way to cut the chip into two parts, each containing a similar number of the inputs, so that only $\approx \sqrt{A}$ wires are cut. Imagine that each part of the chip represents a person. The chip design describes how f can be computed by two people, each of whom knows roughly half of the input to f, with a conversation whose length is proportional to the number of wires that were cut. So, if we can show that computing f requires a conversation of length t, we can conclude that area A must be at least $\approx t^2$, no matter how the components of the chip are laid out. In this book, we will develop a wide variety of tools for proving that functions require conversations whose length is proportional to n. The area of any chip for such a function must be proportional to n^2.

A second example comes from a classical result about *Turing machines*. Turing machines are widely regarded as a universal model of computation – the extended Church-Turing thesis says that anything

that is efficiently computable is efficiently computable by a Turing machine. A Turing machine can be thought of as a program written for a computer that has access to one or more *tapes*. Each tape has a *head* that points at a location on the tape. In each step of computation, the machine can read or write a single symbol at the location corresponding to a head, or move the head to an adjacent location. A Turing machine with more tapes is more powerful than a Turing machine with fewer tapes, but how much more powerful?

A classic result[2] shows that one can simulate a Turing machine that has access to two tapes with a Turing machine that has access to just one tape. However, the simulation may increase the number of steps of the computation by a factor of t, where t is the running-time of the machine. One can use communication complexity to show that this loss is unavoidable.[3]

To see why this is the case, we use the communication complexity of the *disjointness* function. Imagine that Alice knows a set $X \subseteq [n]$, and Bob knows a set $Y \subseteq [n]$. Their common goal is to compute whether or not the sets are disjoint – namely whether or not there is an element that is in both sets (see Figure I.2). Later in the book, we prove that Alice and Bob must communicate $\Omega(n)$ bits in order to achieve this goal.

Now, a Turing machine with access to two tapes can compute disjointness in $O(n)$ steps. If the sets are represented by their indicator vectors $x, y \in \{0,1\}^n$, then the machine can copy y to the second tape and scan both x and y, searching for an index i with $x_i = 1 = y_i$. All of these operations can be carried out in $O(n)$ steps.

However, one can use communication complexity to prove that a Turing machine with one tape must take at least $\Omega(n^2)$ steps to compute disjointness. The idea is that a machine that computes disjointness in T steps can be used by Alice and Bob to compute disjointness using $\approx \frac{T}{n}$ bits of communication. Intuitively, Alice and Bob can write down the input (x, y) on the single tape of the machine and try to simulate the execution of the machine. Neither of them knows the contents of the whole tape, but they can still simulate the execution of the machine with a small amount of communication. Every time the machine transitions from Alice's part of the tape to Bob's part, she sends him a short message to indicate the line of code that should be executed next. Bob then continues the execution. One can show that this simulation can be carried out in such a way that each message sent between Alice and Bob corresponds to $\Omega(n)$ steps of the Turing machine. So, if we start with a one-tape machine that runs in time $\ll n^2$, we end up with a protocol of length $\ll n$ bits that computes disjointness. This is impossible.

The two examples we have discussed give some feel for communication problems and why we are interested in studying them. Next, we continue this informal introduction with several other interesting examples of communication problems and protocols.

[2] Hartmanis and Stearns, 1965.

Strictly speaking, the simulation increases by a factor of $\max\{n, t\}$, where n is the length of the input. However, for any computation that depends on the whole input, this maximum is t.

[3] Hennie, 1965.

Figure I.2 looks very similar to the famous Sierpinski gasket, which is a well-known fractal in the plane. The gasket's area is zero, and its Hausdorff dimension is ≈ 1.585. The properties of disjointness we establish in this text imply analogous statements for the Sierpinski gasket.

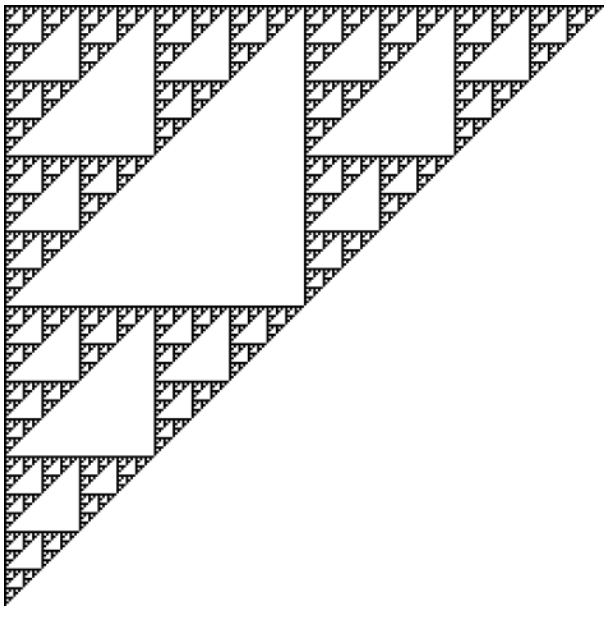

Figure I.2 Disjointness when $n = 8$. Each row corresponds to a set $X \subseteq [8]$, and each column corresponds to a set $Y \subseteq [8]$. The X, Y entry is black if and only if X and Y are disjoint.

Some Protocols

A *communication protocol* specifies a way for a set of people to have a conversation. Each person has access to a different source of information, which is modeled as an *input* to the protocol. The protocol itself is assumed to be known to all the people that are involved in executing it. Their goal is to learn some feature of all the information that they collectively know.

Equality Suppose Alice and Bob are given two n-bit strings. Alice is given x and Bob is given y, and they want to know if $x = y$. There is a trivial solution: Alice can send her input x to Bob, and Bob can let her know if $x = y$. This is a *deterministic* protocol that takes $n + 1$ bits of communication. Interestingly, we shall prove that no deterministic protocol is more efficient. On the other hand, for every number k, there is a *randomized* protocol that uses only $k + 1$ bits of communication and errs with probability at most 2^{-k} – the parties can use randomness to hash their inputs and compare the hashes. More on this in Chapter 3.

The terms *deterministic* and *randomized* will be formally defined later in the book.

See Chapter 3.

Median Suppose Alice is given a list of numbers from $[n]$ and Bob is given a different list of numbers from $[n]$. They want to compute the median element of the list that is obtained by combining these lists. If t is the total number of elements in their lists, this is the $\lceil t/2 \rceil$th element after the lists are combined and sorted. There is a simple protocol that takes $O(\log n \cdot \log t)$ bits of communication. In the first step, Alice and Bob each announce the number of their elements that are at most $n/2$. This takes $O(\log t)$ bits of communication. If there

For example, the median of $(1, 2, 3)$ and $(2, 3, 4)$ is 2, the third element in the list $(1, 2, 2, 3, 3, 4)$.

are k elements that are at most $n/2$ and $k \geq \lceil t/2 \rceil$, then Alice and Bob can safely discard all the elements that are larger than $n/2$ and recurse on the numbers that remain. If $k < \lceil t/2 \rceil$, then Alice and Bob can recurse after throwing out all the numbers that are at most $n/2$, and replacing $\lceil t/2 \rceil$ by $\lceil t/2 \rceil - k$. There can be at most $O(\log n)$ steps before all of their elements must come from a set of size 1. This single number is the median.

Cliques and Independent Sets Here Alice and Bob are given a graph G on n vertices. In addition, Alice knows a clique C in the graph, and Bob knows an independent set I in the graph. They want to know whether C and I share a common vertex or not, and they want to determine this using a short conversation. Describing C or I takes about n bits, because in general the graph may have 2^n cliques or 2^n independent sets. So, if Alice and Bob try to tell each other what C or I is, that will lead to a very long conversation.

A clique is a set of vertices that are all connected to each other. An independent set is a set of vertices that contains no edges. The degree of a vertex is the number of its neighbors.

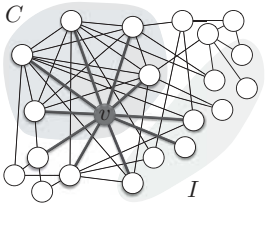

Figure I.3 The vertices that are not neighbors of v cannot be involved in any intersection between C and I.

Here we discuss a clever interactive protocol allowing Alice and Bob to have an extremely short conversation for this task. They will send at most $O(\log^2 n)$ bits. If C contains a vertex v with degree less than $n/2$, Alice sends Bob the name of v. This takes just $O(\log n)$ bits of communication. See Figure I.3 for an illustration. Now, either $v \in I$, or Alice and Bob can safely discard all the nonneighbors of v because these cannot be a part of C. This eliminates at least $n/2$ vertices from the graph. Similarly, if I contains a vertex v of degree at least $n/2$, Bob sends Alice the name of v. Again, either $v \in C$, or Alice and Bob can safely delete all the neighbors of v from the graph, which eliminates about $n/2$ vertices. If all the vertices in C have degree more than $n/2$, and all the vertices in I have degree less than $n/2$, then C and I do not share a vertex. The conversation can safely terminate. So, in each round of communication, either the parties know that $C \cap I = \emptyset$, or the number of vertices is reduced by a factor of 2. After k rounds, the number of vertices is at most $n/2^k$. If k exceeds $\log n$, the number of vertices left will be less than 1, and Alice and Bob will know if C and I share a vertex or not. This means that at most $\log n$ vertices can be announced before the protocol ends, proving that at most $O(\log^2 n)$ bits will be exchanged before Alice and Bob learn what they wanted to know.

One can show that if the conversation involves only one message from each party, then at least $\Omega(n)$ bits must be revealed for the parties to discover what they want to know. So, interaction is vital to bringing down the length of the conversation.

Disjointness with Sets of Size k Alice and Bob are given two sets $A, B \subseteq [n]$, each of size $k \ll n$, and want to know if the sets share a common element. Alice can send her set to Bob, which takes $\log \binom{n}{k} \approx k \log(n/k)$ bits of communication. There is a randomized

See Chapter 2.

protocol that uses only $O(k)$ bits of communication. Alice and Bob sample a random sequence of sets in the universe and Alice announces the name of the first set that contains A. If A and B are disjoint, this eliminates half of B. In Chapter 3, we prove that repeating this procedure gives a protocol with $O(k)$ bits of communication.

Disjointness with k Parties The input is k sets $A_1, \ldots, A_k \subseteq [n]$, and there are k parties. The ith party knows all the sets *except for* the ith one. The parties want to know if there is a common element in all sets. We know of a clever deterministic protocol with $O(n/2^k)$ bits of communication, and we know that $\Omega(n/4^k)$ bits of communication are required. We do not know of any randomized protocol with communication better than the deterministic protocol discussed eariler, but we do know every randomized protocol must have communication at least $\Omega(\sqrt{n}/2^k)$.

See Chapters 4 and 5. In Chapter 4, we prove that when $k > \log n$, it is enough for each party to announce the number of elements in $[n]$ that are in i of the sets visible to her for $i = 0, 1, \ldots, k$.

Summing Three Numbers The input is three numbers $x, y, z \in [n]$. Alice knows (x, y), Bob knows (y, z), and Charlie knows (x, z). The parties want to know whether or not $x + y + z = n$. Alice can tell Bob x, which would allow Bob to announce the answer. This takes $O(\log n)$ bits of communication. There is a clever deterministic protocol that communicates $\sqrt{\log n}$ bits, and one can show that the length of any deterministic conversation must increase with n. In contrast, there is a randomized protocol that solves the problem with a conversation whose length is a constant.

See Chapter 4.

Pointer Chasing The input consists of two functions $f, g : [n] \to [n]$, where Alice knows f and Bob knows g. Let $a_0, a_1, \ldots, a_k \in [n]$ be defined by setting $a_0 = 1$, and $a_i = f(g(a_{i-1}))$. The goal is to compute a_k. There is a simple k round protocol with communication $O(k \log n)$ that solves this problem, but any protocol with fewer than k rounds requires $\Omega(n)$ bits of communication.

See Chapter 6.