CS 761: Randomized Algorithms **University of Waterloo: Winter 2025**

Lecture 5

January 28, 2025

Instructor: Sepehr Assadi

Topics of this Lecture

In this lecture, we are going to look at two highly active area of research in Theoretical Computer Science (but from a very limited perspective to fit into a single lecture!), namely, communication complexity, and error correcting codes.

1 Communication Complexity

Suppose we have a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ (one can consider different domains for the function, different ranges, relations instead of Boolean functions, or even functions with more than two arguments corresponding to *multi-party* communication models; for brevity, we stick to the most basic setting in this lecture). Function f defines a *communication problem* as follows: there are two players, say Alice and Bob, who get inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$, respectively, and their goal is to compute $f(x,y)$. For instance, in the *equality problem*, Alice and Bob would like to evaluate $EQ(x,y)$ which is defined as:

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

namely, determine whether or not their inputs are equal.

Considering neither Alice nor Bob has the entire input on their own, the players need to *communicate* with each other to determine the value of $f(x,y)$. The communication happens according to a *protocol* and is done as follows: Alice sends a message to Bob based solely on her input; after receiving this message,

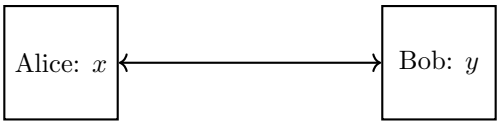

Figure 1: Alice and Bob computing $f(x, y)$

Bob responds back with his message which is a function of his input *and* the message of Alice; the players continue this throughout the protocol, until one of them outputs the answer, i.e., $f(x, y)$.

The main measure of efficiency in this model is the communication cost of the protocols, defined as:

Definition 1 (Communication cost). The communication cost of a protocol π, denoted by $\|\pi\|$, is the *worst-case* number of bits communicated between Alice and Bob in π over any choice of inputs x, y.

One-Way Communication Complexity. To make things easy for this lecture, we are going to focus on a restricted class of communication problems, named **one-way communication complexity**. Here, we only work with protocols wherein Alice sends a single message to Bob and Bob outputs the solution (as in, Bob never sends any message back to Alice even once).

Note that in this model, it is always possible for Alice to send her entire input to Bob in n bits and then Bob can solve the problem on his own. So, we are interested in protocols with communication cost (much) lower than the trivial n bits.

2 The Equality Problem

Let us recall the Equality problem again in the one-way communication model. Alice gets $x \in \{0,1\}^n$, Bob gets $y \in \{0,1\}^n$, and their goal is to output whether or not $x = y$ by Alice sending a single message to Bob.

2.1 Deterministic Protocols

We are going to prove that the trivial protocol wherein Alice sends all her input to Bob is effectively optimal for this problem.

Proposition 2. *In any deterministic protocol for Equality, Alice needs to send a message of size n bits.*

Proof. Suppose towards a contradiction that Alice's message has length less than n; this in particular implies that there are fewer than 2^n possible messages and as such, by pigeonhole principle, there are two strings x_1 and x_2 which are both mapped to the same message m. Given the message m and any input $y \in \{0,1\}^n$, the output of Bob is fixed. But, when Bob's input y is x_1 (or x_2), there is no right answer for Bob: if he says equal, then it is possible that the input to Alice was x_2 instead, and if says not equal, Alice's input could have been x_1, leading to a wrong answer in either case. Hence, Alice's message could have not been less than n bits, completing the proof. □

2.2 Randomized Protocols

The situation for randomized protocols for this problem however is quite different. We are going to show that there is a protocol that succeeds with probability at least $3/4$, while using $O(\log n)$ communication. There are multiple ways of achieving such a result and what we will do in this lecture is probably not the most direct way – however, this sets the stage for the other parts of this lecture as well.

Suppose Alice's protocol is to send a random index $i \in [n]$ together with the value x_i to Bob – then, Bob outputs equal if $x_i = y_i$ and not equal otherwise. Clearly, if $x = y$, this protocol always succeeds, but what if $x \neq y$? Let us consider two examples.

Good case. Suppose not only $x \neq y$, but in fact they are "quite different". More formally, suppose $\Delta(x, y) \geqslant n/3$ where we use $\Delta(x, y)$ to denote the number of indices $j \in [n]$ where $x_j \neq y_j$. In this case, the probability that Bob outputs equal is

$$1 - \frac{\Delta(x, y)}{n} \leqslant \frac{2}{3},$$

because Alice should have not send one of the differing indices.

At this point, to turn this protocol into a one that solves the problem with success at least $3/4$, we simply need to run it in parallel, say, 10, times (i.e., ask Alice to send 10 random indices chosen even with repetition), and then the probability that none of the indices is a differing one is at most

$$\left(1 - \frac{\Delta(x, y)}{n}\right)^{10} \leqslant \left(1 - \frac{1}{3}\right)^{10} \leqslant e^{-10/3} < 1/4.$$

Hence, in the good case, the protocol, with a standard success amplification trick, works very well.

Bad case. But now what if x and y differ "very minimally", say, in only 1 index. In this case, the probability that Bob outputs equal is

$$1 - \frac{\Delta(x, y)}{n} = 1 - \frac{1}{n},$$

which means, the protocols success probability is only $1/n$. This means, to get a constant probability of success (no matter how small), we need to sample $\Omega(n)$ random indices which is way too many (this will need to a protocol with $O(n \log n)$ communication which is way worse than even sending the entire input!).

So, can we avoid this bad case and ensure we are always in the good case somehow? This is the content of *error correcting codes* that we are going to explore next.

3 Error Correcting Codes

An **error correcting code** is a function $\Phi : \{0,1\}^n \to \{0,1\}^m$ such that

- if $x = y$, then so is $\Phi(x) = \Phi(y)$ (this comes for free from the definition of Φ being a function);
- and, if $x \neq y$, then $\Phi(x)$ is "very different" from $\Phi(y)$, namely, $\Delta(\Phi(x), \Phi(y)) \geqslant d \cdot m$ for some fixed $d > 0$ independent of n and m (although in certain cases, this requirement can be relaxed further).

We refer to d as the **distance** of the code and to the ratio of n/m as the **rate**. The rate now governs how much more communication we need in order to achieve the desired distance, since communicating a message in the image of Φ now needs m bits (as opposed to n of the original message).

When talking about error correcting codes, we are most interested in codes with constant rate and constant distance. Of course, there are also various other constraints, possibly most important among them being how fast we can encode and decode a code, but in this lecture, we are going to ignore those aspects.

Going back to our application to the equality problem, if we have a code with constant rate and distance, then Alice and Bob can first apply this code to their inputs, which immediately turn their input to the good case of the previous section, and then run the random sampling protocol over that. This way, they get an $O(\log n)$ communication protocol for the equality problem that succeeds with any arbitrarily large constant probability.

3.1 Constant Rate and Distance Codes

Let us start by proving that constant rate and distance codes do exist.

Theorem 3. *For any $\varepsilon \in (0, \frac{1}{2})$, there are error correcting codes with $(\frac{1}{2} - \varepsilon)$ distance and $\Omega(\varepsilon^2)$ rate.*

Proof. We show existence of a set of strings $\mathcal{C} \leqslant \{0,1\}^m$ of size 2^n for the parameter $m = 3n/\varepsilon^2$ such that for any two $x \neq y \in \mathcal{C}$,

$$\Delta(x, y) \geqslant (\frac{1}{2} - \varepsilon) \cdot m.$$

This immediately implies the existence of the desired code by picking any arbitrary injective mapping from $\{0,1\}^n$ to $\mathcal{C}$. We construct $\mathcal{C}$ randomly.

Suppose we pick two strings x, y uniformly at random and independent from $\{0,1\}^m$. Define m indicator random variables $Z_1, \ldots, Z_m$ such that $Z_i = 1$ iff $x_i \neq y_i$. Define $Z := \sum_{i=1}^m Z_i$ and note that $\Delta(x, y) = Z$. We have,

$$\mathbb{E}\left[\Delta(x, y)\right] = \mathbb{E}\left[Z\right] = \mathbb{E}\left[\sum_{i=1}^{m} Z_i\right] = \sum_{i=1}^{m} \mathbb{E}\left[Z_i\right] = \sum_{i=1}^{m} \Pr\left(x_i \neq y_i\right) = \frac{m}{2},$$

where here the only interesting steps are the third equality which is by linearity of expectation, and the final one which holds since each bit in x_i and y_i is chosen randomly and independently of the other string.

In addition, since Z is a sum of 0/1-independent random variables, we can apply Chernoff bound to Z and obtain that

$$\Pr\left(Z < (1 - 2\varepsilon) \cdot \mathbb{E}\left[Z\right]\right) \leqslant \exp\left(-\frac{4\varepsilon^2 \cdot m}{6}\right) = \exp\left(-\frac{4\varepsilon^2 \cdot 3n}{\varepsilon^2 \cdot 6}\right) = e^{-2n}.$$

This implies that

$$\Pr_{x,y}\left(\Delta(x, y) < (\frac{1}{2} - \varepsilon) \cdot m\right) < e^{-2n},$$

for two independently and uniformly chosen strings $x, y \in \{0,1\}^m$.

Now suppose we sample 2^n strings independently and uniformly from $\{0,1\}^m$ to form $\mathcal{C}$. By a union bound over the at most 2^{2n} pairs of different strings, we have,

$$\Pr_{\mathcal{C}}\left(\text{there are } x \neq y \in \mathcal{C} \text{ such that } \Delta(x, y) < (\frac{1}{2} - \varepsilon) \cdot m\right) < 2^{2n} \cdot e^{-2n} \ll 1.$$

Thus, $\mathcal{C}$ will satisfy our desired properties with a high probability. This in particular proves the existence of such a set (as otherwise this probability will be 1). □

We note that there is also another way of proving Theorem 3 as follows: greedily pick any arbitrary $x \in \{0,1\}^m$ and add it to $\mathcal{C}$; then, remove all strings from $\{0,1\}^m$ that are at distance less than $\ell = (1/2 - \varepsilon) \cdot m$ from x, and recurse. Since for any x we will be removing

$$\sum_{i=0}^{\ell-1} \binom{m}{i}$$

strings at most (since some of them might have already been removed), we obtain that

$$|\mathcal{C}| \geqslant \frac{2^m}{\sum_{i=0}^{\ell-1} \binom{m}{i}}.$$

Approximating the denominator appropriately then allows us to prove Theorem 3 with similar bounds. We note that the above inequality (in a more general form for any $[q]^m$) is often called the **Gilbert-Varshamov** bound in coding theory.

Let us also prove that the constant $1/2$ in Theorem 3 cannot be improved, namely, we cannot hope to have codes with even remotely interesting rate and distance strictly more than half.

Theorem 4. *For any $\varepsilon > 0$, there are at most $O(1/\varepsilon)$ strings in $\{0,1\}^m$ with pairwise distance $(1/2+\varepsilon)\cdot m$.*

Proof. We are going to prove this in two steps: first prove a continous version of this bound in $\mathbb{R}^m$ instead of $\{0,1\}^m$ and then translate this bound to the discrete setting as well.

Claim 5. *For any $\varepsilon \in (0,1)$, there are at most $t = O(1/\varepsilon)$ vectors $v_1, \ldots, v_t \in \mathbb{R}^m$ with $\|v_i\| = 1$ and $\langle v_i\,,\, v_j\rangle \leqslant -\varepsilon$ for all $i \neq j \in [t]$.*

Proof. We have

$$0 \leqslant \|\sum_{i=1}^{t} v_i\|^2 = \sum_{i=1}^{t}\|v_i\|^2 + 2\sum_{i<j}\langle v_i\,,\, v_j\rangle = t - 2\cdot\binom{t}{2}\cdot\varepsilon = t - t\cdot(t-1)\cdot\varepsilon;$$

rearranging the terms implies that

$$t \leqslant 1 + \frac{1}{\varepsilon},$$

concluding the proof. □

To extend to the discrete setting, suppose we have strings $x_1, \ldots, x_t$ with pairwise distance $(1/2+\varepsilon)\cdot m$. Then, for every $i \in [t]$, define the following vector

$$v_i := \frac{1}{\sqrt{m}} \cdot \left((-1)^{x_i[1]}, (-1)^{x_i[2]}, \ldots, (-1)^{x_i[m]}\right),$$

where $x_i[j]$ denotes the j-th bit of the string x_i.

Firstly,

$$\|v_i\|^2 = \sum_{j=1}^{m} v_i[j]^2 = \frac{1}{m}\cdot\sum_{j=1}^{m} 1 = 1,$$

and thus these vectors are unit-norm (as needed by our claim).

Secondly, for any $i \neq j$,

$$\begin{aligned}
\langle v_i\,,\, v_j\rangle &= \frac{1}{m}\cdot\sum_{k=1}^{m}(-1)^{x_i[k]}\cdot(-1)^{x_j[k]} && \text{(by the definition of the vectors)}\\
&= \frac{1}{m}\cdot((m-\Delta(x_i,x_j))\cdot 1 + (\Delta(x_i,x_j)\cdot(-1))) \\
&\qquad\qquad \text{(as equal-indices contribute } +1 \text{ while different-indices contribute } (-1))\\
&= \frac{1}{m}\left(m - 2\cdot\Delta(x_i,x_j)\right)\\
&\leqslant \frac{1}{m}\cdot(m - m - 2\varepsilon\cdot m) && \text{(by the lower bound on the distance of } x_i, x_j)\\
&= -2\varepsilon.
\end{aligned}$$

Thus, these vectors satisfy the premise of our claim above and hence we can conclude $t = O(1/\varepsilon)$ for the strings as well, concluding the proof. □

The bound in Theorem 4 is often referred to as the **Plotkin bound** in coding theory.

4 The Index Problem

Let us now see an entirely different application of error correcting codes, this time for proving a lower bound on the communication cost of another problem. In the *Index* problem, Alice receives a string $x \in \{0,1\}^n$ and Bob receives an index $i \in [n]$. The goal is for Alice to send a single message to Bob, and Bob should output x_i.

It is easy to see that any deterministic protocol for this problem requires n bits of communication (similar to the Equality problem) since Bob will be able to reconstruct x from the message sent by Alice by simply iterating over all choices of $i \in [n]$. Thus, the same argument as in Proposition 2 implies an n-bit lower bound for this problem. But, what about randomized protocols?

Unlike the Equality problem, the Index problem is hard even for randomized protocols in that they also need $\Omega(n)$ bits of communication to solve this problem with high constant probability. Formally,

Theorem 6. *Any randomized protocol for the Index problem on $\{0,1\}^n$ requires $\Omega(n)$ communication to succeed with probability at least 0.99.*

We will prove this theorem in the rest of this lecture.

4.1 Distributional Complexity

A general (and highly successful) way of proving lower bounds for randomized protocols (and in general algorithms) is to study them over a known *distribution* of inputs and turn them into deterministic protocols. Let us see this in action.

Let $\mathcal{C} \subseteq \{0,1\}^n$ be a set of error correcting codes (specifically their images) with distance $n/3$ and by Theorem 3 we know that the size of $\mathcal{C}$ can be $2^{\Omega(n)}$. Consider the following distribution μ over inputs to Alice and Bob:

- Sample x uniformly at random from $\mathcal{C}$ and let Alice's input be x;
- Sample i independently of x and uniformly from $[n]$ and let Bob's input be i.

Now, suppose π is a protocol for the Index problem with probability of success at least 0.99. Since π is randomized, we denote its randomness by r and write π_r to obtain the deterministic protocol obtained from π when its choice of random bits is r. By the definition of the protocol, for every (x,i) to the Index problem,

$$\Pr_r\left(\pi_r \text{ succeeds on } (x,i)\right) \geqslant \frac{99}{100}.$$

Naturally, this implies that for $(x,i) \sim \mu$ also, we have,

$$\mathop{\mathbb{E}}_{(x,i)\sim\mu} \Pr_r\left(\pi_r \text{ succeeds on } (x,i)\right) \geqslant \frac{99}{100}.$$

But, since r and (x,i) are independent of each other (the randomness of a protocol/algorithm is always chosen independent of the input), we can re-order the expectation and probability above and have,

$$\mathop{\mathbb{E}}_{r} \Pr_{(x,i)\sim\mu}\left(\pi_r \text{ succeeds on } (x,i)\right) \geqslant \frac{99}{100}.$$

But then this implies that, by an averaging argument, there is a fixed choice r^* of the random bits such that

$$\Pr_{(x,i)\sim\mu}\left(\pi_{r^*} \text{ succeeds on } (x,i)\right) \geqslant \frac{99}{100}.$$

In other words, we also have a *deterministic* protocol that on inputs sampled from μ, achieves a 99/100 probability of success. In addition, since communication cost is a worst-case measure, we get that π_{r^*} has communication cost as that of π.

The conclusion of this is that we can simply focus on proving the lower bound for deterministic algorithms, against a distribution of inputs, and obtain the randomized lower bound as well. This is often called **the easy direction of Yao's minimax principle**. As an aside, the hard direction of Yao's minimax principle shows that this approach is without loss of generality, meaning that, there always exist a distribution of inputs on which, the performance of deterministic algorithms match the original randomized algorithm exactly.

4.2 The Lower Bound

Equipped with the previous step, from now on, we can assume we have a deterministic protocol π that on inputs sampled from μ succeeds with probability at least 99/100, and our goal is to prove a lower bound on its communication cost.

Let $m = m(x)$ denote the message sent by Alice and $f(m, i)$ denote the function used by Bob to output the answer. In the following, all randomness is with respect to the distribution μ. We have,

$$\mathop{\mathbb{E}}_{(x,i)} \Pr\left(f(m(x), i) \neq x_i\right) \leqslant \frac{1}{100}.$$

Now, given each x let $B(x)$ denote the "bad" indices $i \in [n]$ such that the answer to the protocol on the input (x, i) is wrong, i.e., $f(m(x), i) \neq x_i$. Since i is chosen independent of x to be uniform over $[n]$, by the above equation, we have that

$$\mathop{\mathbb{E}}_{(x,i)} \Pr\left(f(m(x), i) \neq x_i\right) = \mathop{\mathbb{E}}_{x}\left[\frac{1}{n}\sum_{i=1}^{n} \Pr\left(f(m(x), i) \neq x_i\right)\right] = \frac{1}{n}\mathop{\mathbb{E}}_{x}\left[\sum_{i=1}^{n} \Pr(i \in B(x))\right] = \frac{1}{n} \cdot \mathop{\mathbb{E}}_{x}|B(x)|,$$

which together with the earlier equation, implies

$$\mathop{\mathbb{E}}_{x}|B(x)| \leqslant \frac{n}{100}.$$

By Markov's inequality, this implies that

$$\Pr\left(|B(x)| \geqslant \frac{n}{50}\right) \leqslant \frac{1}{2}.$$

This means that for at least half the strings $x \in \mathcal{C}$, given the message $m(x)$, we can recover all but $n/50$ (unknown) indices of x correctly and the remaining indices will be wrong. Let $\mathcal{C}' \subseteq \mathcal{C}$ be the set of such strings and for each $x \in \mathcal{C}'$ use $\tilde{x}$ to denote the recovered string, i.e.,

$$\tilde{x} = f(m(x), 1)\ ,\ f(m(x), 2)\ ,\ \ldots\ , f(m(x), n).$$

We claim that we can recover x from $\tilde{x}$ uniquely given that $\mathcal{C}$ was an error correcting code. Specifically, we have

$$\Delta(x, \tilde{x}) \leqslant \frac{n}{50},$$

as established above, but on the other hand, for any $y \neq x \in \mathcal{C}$, using triangle's inequality,

$$\Delta(y, \tilde{x}) \geqslant \Delta(y, x) - \Delta(x, \tilde{x}) \geqslant \frac{n}{3} - \frac{n}{50} > \frac{n}{50} \geqslant \Delta(x, \tilde{x});$$

As such, by mapping each $\tilde{x}$, which we can recover solely from the message m, to the closest point in $\mathcal{C}'$, we can recover $x \in \mathcal{C}'$ as well. In other words, the message of the protocol can recover half the strings in $\mathcal{C}$ uniquely and thus on those strings, it should be a one-to-one mapping. Given the size of $\mathcal{C}$, this implies that we need $2^{\Omega(n)}$ different messages and hence communication cost has to be $\Omega(n)$, concluding the proof of Theorem 6.