Interactive Coding

Aditya Sriram, Keshav Patel Keval

Indian Institute of Technology, Bombay

November 26, 2022

Summary

Motivation

Imagine a scenario where two players, Aditi and Bharat are playing chess over the phone. There is a third party, Eesha, who has access to their telephone line and can corrupt their messages. Suppose Bharat declares the move, "B2 to B4". As an adversary, Eesha tampers with the message and Aditi hears the move, "E2 to E4". A few days later, Bharat declares victory but Aditi rejects his claim because her board paints a different picture!

One method to communicate safely, involves using classical error correcting schemes to encode each message separately. However, such schemes would only be able to tolerate a very small amount of noise - noise sufficient to corrupt a single message. In this talk, we describe schemes that succeed even when a constant fraction of messages are corrupted, and do not have a significant overhead.

Formal Setting (Yao)

Let X, Y, Z be arbitrary finite sets and let $f : X \times Y \to Z$ be an arbitrary function. There are two players, A and B who wish to evaluate $f(x, y)$, for some inputs $x \in X$ and $y \in Y$. However, x is known only to A and y is known only to B. Hence, the two parties must communicate with each other. The communication is carried out according to a protocol, $\mathcal{P}$ (which depends only on f).

Formal Setting (Yao)

Definition 1.1

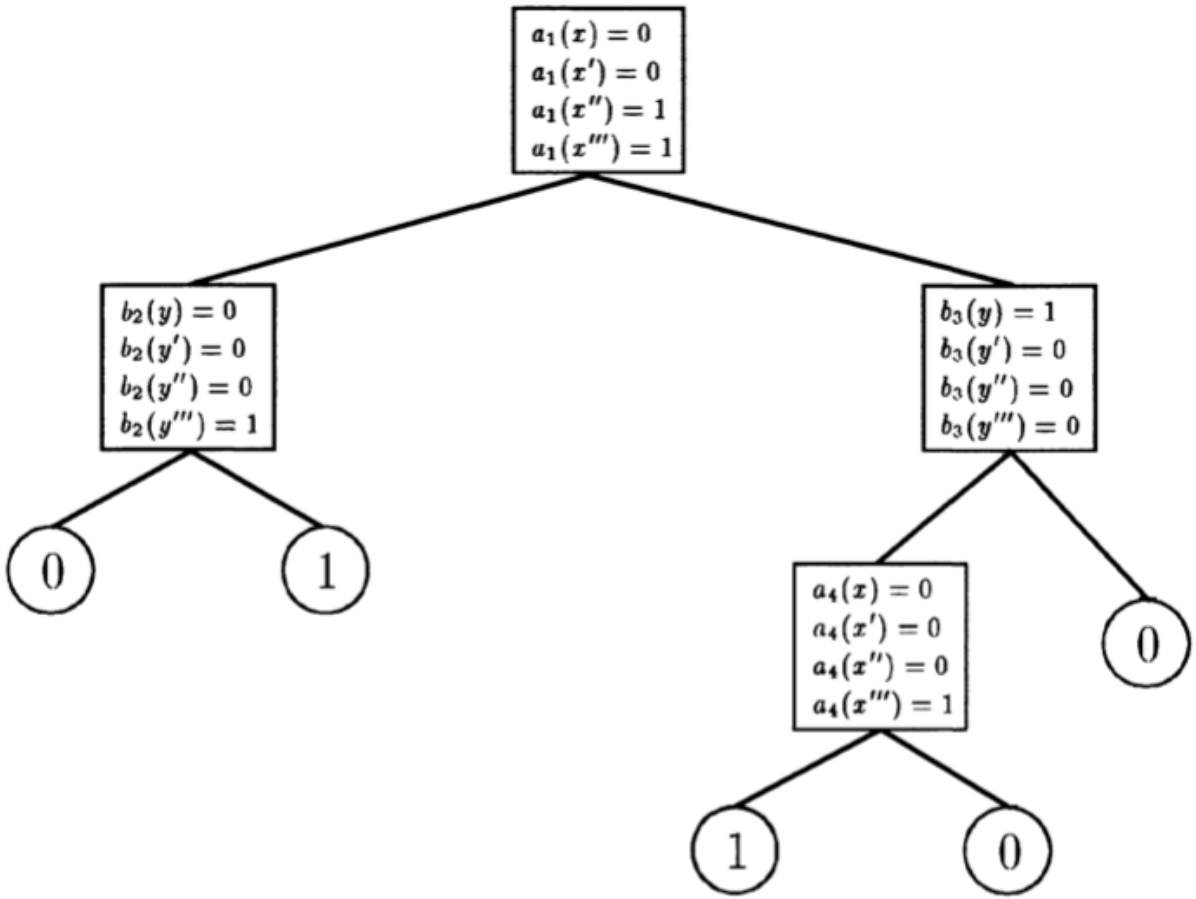
A protocol $\mathcal{P}$ over domain $X \times Y$ and range Z is a binary tree where each internal node ν is labeled either by a function $a_\nu : X \to \{0,1\}$ or by a function $b_\nu : Y \to \{0,1\}$, and each leaf is labeled with an element $z \in Z$. The value of the protocol $\mathcal{P}$ on input (x, y) is the label of the leaf reached by starting from the root, and traversing the tree. At each internal node ν labeled by a_ν (b_ν), the edge taken depends on the value of $a_\nu(x)$ ($b_\nu(y)$). The cost of $\mathcal{P}$ is the height of the tree.

Definition 1.2

For a function $f : X \times Y \to Z$, the (deterministic) communication complexity of f is the minimum cost of $\mathcal{P}$, over all protocols $\mathcal{P}$ that compute f. We denote the (deterministic) communication complexity of f by $\mathcal{D}(f)$.

Note: We assume that the two parties are computationally unbounded.

Protocol Tree



Source: Communication Complexity - Kushilevitz, Nisan

A Trivial Upper Bound

Proposition 1.1

For every function $f : X \times Y \to Z$,

$$\mathcal{D}(f) \leq \min\left(\log_2 |X|, \log_2 |Y|\right) + \log_2 |Z|$$

Proof

Consider the following protocol where party A sends all her inputs to party B. This requires sending $\log_2 |X|$ bits. B computes the function privately and sends the result of his computation back to party A. This requires sending $\log_2 |Z|$ bits. A similar argument holds for party B, sending all his inputs to A and the result follows.

Noisy Setting

The protocol is a pair of algorithms, $\pi \equiv (\pi_A, \pi_B)$ run by the parties. In each round, the message sent by a party is a function of the its input, the round number and the set of symbols it has received (this is called a transcript). Note that in general, the symbols received by one party will not be the ones sent to them by the other. The protocol concludes after a fixed number of rounds, n and each party outputs a value. The output of party A is $\pi_A(x, n+1, \text{trans}_A)$ and the output of party B is $\pi_B(y, n+1, \text{trans}_B)$.

The round complexity of the protocol is its length, $|\pi| = n$. The messages sent by each party are taken from an alphabet, Σ and the communication complexity of the protocol is defined to be, $CC(\pi) = n \log |\Sigma|$. We will compare the performance of the noise-resilient protocol, π to the noiseless protocol, π_0 for the same function. The noiseless protocol is defined over a binary alphabet and has a communication complexity, $CC(\pi_0) = n_0$.

Performance Parameters

- **Maximal Noise Ratio**: We will look at coding schemes that can tolerate a constant fraction of noise, i.e. when $\epsilon = O(1)$.
- **Code Rate**: The rate of the coding scheme π with respect to the noiseless π_0, defined as follow:

$$r = \frac{CC(\pi_0)}{CC(\pi)}$$

  indicates the amount of redundancy added. We will focus on schemes that have a constant rate. If a scheme does not have a constant rate, that is, when $\lim_{n_0 \to \infty} r = 0$, we say the coding scheme has a vanishing rate.
- **Success Probability**: The probability that both parties output the correct value. The probability is computed over the randomness of the protocol (if randomized) and the noise (if randomized). We aim to obtain coding schemes that succeed with exponentially high probability in the length of the noiseless protocol, $1 - 2^{-\Omega(n_0)}$.

Tree Code: Definitions

Let $\mathcal{T}$ be a rooted d-ary tree. $[d]$ denotes the set $\{1, 2, ..., d\}$. A node is identified with the rooted path leading up to it in an inductive fashion. The root is identified with the empty path. The path $(e_1, e_2, ..., e_n)$ where $e_i \in [d]$, is identified with the e_n-th child of the node at the end of the rooted path $(e_1, e_2, ..., e_{n-1})$.

Definition 2.1

A d-ary tree code over an alphabet Σ with distance parameter α is an infinite rooted d-ary tree in which each edge e is marked with a label $w_e \in \Sigma$. For any two rooted paths, p_1 and p_2 of the same length, that diverge at the j-th level, it holds that

$$\Delta(label(p_1), label(p_2)) \geq \alpha(n - j + 1)$$

where $label(p)$ denotes the concatenation of the labels along the path p and $\Delta(x, y)$ denotes the Hamming distance between strings x and y.

Properties

Definition 2.2 (Suffix Distance)

For any two strings $x, y \in \Sigma^n$, the relative suffix distance is defined as,

$$\Delta_{sfx}(x, y) = \max_{i=1}^{n} \frac{\Delta(x_i \cdots x_n, y_i \cdots y_n)}{n - i + 1}$$

Proposition 2.1

A tree $\mathcal{T}$ is a tree code with distance α iff for any two different rooted paths p_1, p_2 of the same length in $\mathcal{T}$, $\Delta_{sfx}(x, y) \geq \alpha$

Suffix Distance

Proof

Suppose $\mathcal{T}$ is a tree code. We will show that the suffix distance condition holds. Consider any two rooted paths, $p_1 = (e_1, ..., e_{j-1}, e_j, ..., e_n)$ and $p_2 = \left(e_1, ..., e_{j-1}, e'_j, ..., e'_n\right)$ with $e_j \neq e'_j$. Let $label(p_1) = l_1^1 \cdots l_n^1$ and $label(p_2) = l_1^2 \cdots l_n^2$,

$$\begin{aligned}
\Delta_{sfx}(label(p_1), label(p_2)) &= \max_{i=1}^{n} \frac{\Delta(l_i^1 \cdots l_n^1, l_i^2 \cdots l_n^2)}{n-i+1} \\
&\geq \frac{\Delta(l_j^1 \cdots l_n^1, l_j^2 \cdots l_n^2)}{n-j+1} \\
&\geq \frac{\alpha\,(n-j+1)}{n-j+1} \\
&\geq \alpha
\end{aligned}$$

Suffix Distance

Proof

Now assume that the suffix distance condition holds for any two different rooted paths, p_1 and p_2 in the tree. We want to show that,

$$\Delta\left(\mathit{label}(p_1), \mathit{label}(p_2)\right) \geq \alpha\left(n - j + 1\right)$$

For $i \geq 1$, define $\delta_i = \Delta(l_1^1 \cdots l_{j-1+i}^1, l_1^2 \cdots l_{j-1+i}^2)$ and $s_i = \Delta_{sfx}(l_1^1 \cdots l_{j-1+i}^1, l_1^2 \cdots l_{j-1+i}^2)$. We also define, $\delta_0 = 0$ (the rooted paths do not diverge until the j-th level). We will show that,

$$\delta_i \geq \alpha i$$

which will prove the claim. For $i = 1$, $\delta_1 = s_1 \geq \alpha$ ($l_1^1 \cdots l_j^1$ and $l_1^2 \cdots l_j^2$ are part of p_1 and p_2 respectively and are rooted paths themselves). For $i > 1$, assume that $\delta' \geq \alpha i'$ for $i' < i$.

Suffix Distance

Proof

Let $k \in [1, i]$ be the offset (beyond level $j-1$) that maximises s_i, i.e. for which $\Delta(l^1_{j-1+k} \cdots l^1_{j-1+i}, l^2_{j-1+k} \cdots l^2_{j-1+i}) \geq \alpha(i-k+1)$ (the paths diverge at the j-th level, so the maximum for suffix distance can only occur after this point). Using the inductive hypothesis for $i' = k-1$,

$$\begin{aligned}\delta_i &= \Delta(l^1_1 \cdots l^1_{j-1+k-1}, l^2_1 \cdots l^2_{j-1+k-1}) + \Delta(l^1_{j-1+k} \cdots l^1_{j-1+i}, l^2_{j-1+k} \cdots l^2_{j-1+i}) \\ &\geq \alpha(k-1) + \alpha(i-k+1) \\ &\geq \alpha \cdot i\end{aligned}$$

Encoding

Tree codes can be used to encode messages in an online fashion. A d-ary code has the following encoding,

$$TC_{enc_{\mathcal{T}}}(m) = w_1 w_2 \cdots w_n$$

where w_i is the i-th label along the rooted path defined by the message $m = m_1 m_2 \cdots m_n$. Observe that the tree code is prefix code, i.e. the encoding of $m_1 m_2 \cdots m_i$ only depends on $m_1 m_2 \cdots m_i$ and is independent of m_j for $j > i$.
For any message m and symbol $\sigma \in [d]$, it holds that

$$TC_{enc_{\mathcal{T}}}(m\sigma) = TC_{enc_{\mathcal{T}}}(m) \circ w_\sigma$$

where w_σ is the label of edge at the end of the rooted path, $m\sigma$. This online property is crucial for the use of a tree code in an interactive communication scheme. By round i, the parties would have communicated symbols $TC_{enc}(m_1 \cdots m_{i-1})$ and to extend their messages to $m_1 \cdots m_i$, they only need to send one additional symbol.

Decoding

Just like classical error correcting codes, tree codes can be decoded using nearest-neighbour decoding. Decoding a received word, $r \in \Sigma^n$ amounts to returning the message $m \in [d]^n$ whose encoding lies closest to r in terms of Hamming distance.

$$TC_{dec}(r) = \text{argmin}_{m \in [d]^n} \Delta(TC_{enc_{\mathcal{T}}}(m), r)$$

Lemma 2.1

Assume $\mathcal{T}$ is a tree code with distance α over alphabet Σ. For any $r \in \Sigma^n$, such that,

$$\Delta_{sfx}(r, TC_{enc_{\mathcal{T}}}(m)) < \frac{\alpha}{2}$$

it holds that $TC_{dec_{\mathcal{T}}}(r) = m$.

Decoding

Proof

The proof proceeds by contradiction. Assume that $\Delta_{sfx}(r, TC_{enc_{\mathcal{T}}}(m)) < \frac{\alpha}{2}$ but $TC_{dec_{\mathcal{T}}}(r) = m'$ where m' and m differ from the j-th index. Let $w = TC_{enc_{\mathcal{T}}}(m)$ and $w' = TC_{enc_{\mathcal{T}}}(m')$.
$\Delta_{sfx}(r, w) < \frac{\alpha}{2} \implies \Delta(r_j \cdots r_n, w_j \cdots w_n) < \alpha(n - j + 1)/2$.
$\because m' = \text{argmin}_{m \in [d]^n} \Delta(TC_{enc_{\mathcal{T}}}(m), r)$, $\Delta(w', r) < \Delta(w, r) \implies$
$\Delta(w'_j \cdots w_n, r_j \cdots r_n) < \Delta(w_j \cdots w_n, r_j \cdots r_n) < \alpha(n - j + 1)/2$. By the Δ-inequality, $\Delta(w, w') \leq \Delta(w', r) + \Delta(w, r) < \alpha(n - j + 1)$ which contradicts the tree code property.

Lemma 2.2

Assume $\mathcal{T}$ is a d-ary tree code with distance α, over alphabet Σ. Let $m \in [d]^n$ be some message, and let $s = TC_{enc_{\mathcal{T}}}(m)$ be its encoding using the tree code $\mathcal{T}$. For any $r \in \Sigma^n$ such that $TC_{dec_{\mathcal{T}}}(r)$ differs from m starting from the j-th index, it holds that, $\Delta(s_j \cdots s_n, r_j \cdots r_n) \geq \frac{\alpha(n-j+1)}{2}$

Existence (Schulman)

Theorem 2.1

For any $\alpha < 1$, there exists a d-ary tree code of infinite depth and distance α over alphabet of size $|\Sigma| = (cd)^{1/(1-\alpha)}$, for some constant, $c < 6$.

Proof

Let $g = g_1 g_2 \cdots$ be an infinite sequence with $g_i \in \mathbb{F}$, where $|\mathbb{F}|$ is to be determined. The string g is used to generate the edge labels in the following way: the edge at the end of the rooted path $p = (e_1, e_2, ..., e_n)$ is labelled by

$$w_p = \sum_{i=1}^{n} e_i g_{n-i+1}$$

g_i is picked independently and uniformly at random to generate g. We show that there is a positive probability of satisfying the tree code property.

Existence (Schulman)

Proof

Suppose two different paths $p_1 = (e_1, ..., e_n)$ and $p_2 = (e'_1, ..., e'_n)$ of the same length diverge at the j-th level, i.e. $e_j \neq e'_j$. label(p_1) and label(p_2) are identical until the $(j-1)$-th index and differ at the j-th index. After that point, the strings differ at some index $k \geq j$ iff

$$\sum_{i=j}^{k} e_i g_{k-i+1} \neq \sum_{i=j}^{k} e'_i g_{k-i+1}$$

or equivalently,

$$\sum_{i=j}^{k} (e_i - e'_i) g_{k-i+1} \neq 0$$

Existence (Schulman)

Proof

Define the indicator variable, X_k which takes the value 1 when the k-th labels of the two paths differ. From the construction, $X_k = 0$ for $k < j$. For $k \geq j$, $\mathbb{E}[X_k] = 1 - 1/|\mathbb{F}|$ because by fixing $g_1, ..., g_{k-j}$, there is exactly one g_{k-j+1} that violates the condition. Also, observe that X_k and $X_{k'}$ are independent as long as $k, k' \geq j$. Finally, X_j, ..., X_n only depend on the difference of the labels of the suffixes, $(e_1 - e_1'), ..., (e_n - e_n')$.
We call a specific g bad if for some (p_1, p_2) of length n that differ at their j-th edge, $\sum_{i=j}^{n} X_i < \alpha(n - j + 1)$. We bound the probability of g being bad using Chernoff's bound,

$$\mathbb{P}[g \text{ is bad}] \leq \mathbb{P}\left[\sum_{i=j}^{n} X_i < \alpha(n - j + 1)\right] \leq e^{-(n-j+1)D(\alpha||1-1/|\mathbb{F}|)}$$

where $D(x||y) = x \ln \frac{x}{y} + (1 - x) \ln \frac{1-x}{1-y}$.

Existence (Schulman)

Proof

Now, we take a union bound over all possible (p_1, p_2) of depth n. This only depends on $(e_j - e'_j), ..., (e_n - e'_n)$ which can take atmost d^{n-j+1} values. Thus, the probability that the string g is bad for some (p_1, p_2) of any depth, is bounded by

$$\sum_{n-j=0}^{\infty} d^{n-j+1} e^{-(n-j+1)D(\alpha||1-1/|\mathbb{F}|)} \leq \sum_{n-j=0}^{\infty} e^{-(n-j+1)(\ln d - D(\alpha||1-1/|\mathbb{F}|))}$$

which is strictly less than 1 if $|\mathbb{F}| > (ed)^{1/(1-\alpha)}$.

Interactive Coding using Tree Codes

We will assume that the noiseless protocol, π_0 is an alternating binary protocol. Recall that any protocol for interactive communication can be visualised as a protocol tree, $\mathcal{T}_0$, a binary tree where party A occupies the odd levels and party B occupies the even.
For any input x, π_0 defines a set of edges, E_x that correspond to the possible replies by A and, respectively, E_y defines a set of edges that correspond to the possible replies by B. For any input (x, y), the edges of $E_x \cup E_y$ define a unique rooted path P_0 which corresponds to the correct transcript of $\pi_0(x, y)$.
This task, where A is given only E_x and B is given only E_y is called the *pointer chasing problem*. An algorithm that solves the pointer chasing problem in the noisy setting, is a coding scheme that works for interactive communication as well (we say that the pointer chasing problem is a **complete** problem for interactive coding).

Interactive Coding using Tree Codes

We describe a protocol that uses a tree code with $d = 2^n$, where n is the round complexity of the protocol ($\mathcal{T}_0$ is of depth n so there are $O(2^n)$ edges). Every node in the tree code at a depth k is going to represent a set of atmost k edges in $\mathcal{T}_0$ (the root corresponds to the empty set, ϕ).
We describe the algorithm from the perspective of party A (the protocol is symmetric for B).
Party A is going to maintain two sets of edges, E_x and $\tilde{S}_B$. E_x is the set of edges in $\mathcal{T}_0$ that correspond to input x and $\tilde{S}_B$ is the set of edges received from B, after decoding. In any round,

1. If $E_x \cup \tilde{S}_B$ contains a unique rooted path, A communicates the edge in E_x, which has not been communicated till now and extends the rooted path
2. If it does not, A communicates a default symbol, $\perp$.

The Braverman-Rao Simulation

Algorithm 1 The Braverman-Rao simulation [18]

Input: a protocol π_0 of length $n_0 = |\pi_0|$ and an input x; a noise-resilience parameter $1/4 - \varepsilon$.
Let $\mathcal{T}$ be a d-ary tree code with distance $\alpha = 1 - \varepsilon$, depth $n = n_0/\varepsilon$, and $d = O(n)$.
Let E_x be the set of edges in $\mathcal{T}_0$ that correspond to Alice's possible replies, assuming she holds the input x.

1: $recv \leftarrow \emptyset$; $S_A \leftarrow \emptyset$
2: **for** $i = 1$ to $n = n_0/\varepsilon$ **do**
3: $\quad \tilde{S}_B \leftarrow \mathsf{TCdec}_{\mathcal{T}}(recv)$ ▷ Interpret each symbol as an edge
▷ (discard invalid/inconsistent edges)
4: $\quad$ **if** $E_x \cup \tilde{S}_B$ has a unique rooted path P **then**
5: $\quad\quad e_i \leftarrow$ the edge of $E_x \cap P$ with the lowest depth that is not in S_A
6: $\quad\quad S_A \leftarrow e_i$
7: $\quad$ **else**
8: $\quad\quad e_i \leftarrow \perp$
9: $\quad$ send the last symbol of $\mathsf{TCenc}_{\mathcal{T}}(e_1 \cdots e_i)$
10: $\quad$ receive a symbol r from Bob
11: $\quad recv \leftarrow recv \circ r$
12: output the unique rooted path (of length n_0) defined by $E_x \cup \tilde{S}_B$

Source: Coding for Interactive Communication: A Survey - Ran Gelles

Analysis

Theorem 2.2

For any $\epsilon > 0$, there exists an interactive coding scheme that simulates any protocol, π_0 in $O_\epsilon(n_0)$ rounds, using a constant-size alphabet, and is resilient to a fraction $1/4 - \epsilon$ corrupted symbols.

We start by proving a lemma that shows that if the noise is bounded, there are many rounds where the suffix distance between the sent and received codeword is small. By lemma 2.1, this implies that the party will be able to decode the received codeword correctly.

Analysis

Lemma 2.3

For any $r, s \in \Sigma^n$, if $\Delta(r, s) = \beta n$, then there exists a set of indices $I \subseteq [n]$ of size $|I| \geq (1 - \beta/\alpha)n$ such that for any $i \in I$,

$$\Delta_{sfx}(r_1 \cdots r_i, s_1 \cdots s_i) < \alpha$$

Proof

Consider the following algorithm to construct I recursively. We begin with $I = \phi$ and set $i = n$.

1. If $\Delta_{sfx}(r_1 \cdots r_i, s_1 \cdots s_i) < \alpha$, add i to I and set $i \leftarrow i - 1$
2. If not, find the largest $j < i$ such that $\Delta_{sfx}(r_1 \cdots r_j, s_1 \cdots s_j) < \alpha$, and set $i \leftarrow j$

For any iteration where we find j and discard the remaining indices, $\Delta(r_{j+1} \cdots r_i, s_{j+1} \cdots s_i) \geq \alpha(i - j)$. However, since the maximal distance is bounded and $n - |I|$ indices are removed, $\alpha(n - |I|) \leq \beta n$.

Analysis

Proof (Braverman-Rao Simulation)

Let s_A, s_B denote the n symbols sent by A and B in the protocol and r_A, r_B denote the n symbols corresponding symbols received. Let, $\beta_A = \frac{\Delta(s_B, r_A)}{n}$ and $\beta_B = \frac{\Delta(s_A, r_B)}{n}$ denote the fraction of noise seen by both parties respectively. For the given noise bound,

$$\frac{\Delta(s_A, r_B) + \Delta(s_B, r_A)}{2n} = \frac{\beta_A + \beta_B}{2} \leq \frac{1}{4} - \epsilon$$

(Factor of two because for a protocol of length n, each party sends n symbols which implies that $2n$ symbols are communicated in total) Since the parties are using a tree code with $\alpha = 1 - \epsilon$, from the lemma we just proved, there will be $|I_A| \geq (1 - \frac{2\beta_A}{1-\epsilon})n$ iterations in which $\Delta(r_A[1 \cdots i], s_B[1 \cdots i]) < \frac{\alpha}{2}$ and $|I_B| \geq (1 - \frac{2\beta_B}{1-\epsilon})n$ iterations in which $\Delta(r_B[1 \cdots i], s_A[1 \cdots i]) < \frac{\alpha}{2}$. In these rounds, the parties will be able to correctly decode the entire message sent to them.

Analysis

Proof

$\because |I_A \cap I_B| \geq |I_A| + |I_B| - n,$

$$\begin{aligned} |I_A \cap I_B| &\geq (2 - \frac{2\beta_A + 2\beta_B}{1-\epsilon})n - n \\ &= 2n - \frac{1-4\epsilon}{1-\epsilon}n - n \\ &\geq 2n - (1-3\epsilon)n - n \\ &= 3n\epsilon \\ &\geq n\epsilon \end{aligned}$$

Since $n = n_0/\epsilon$, $|I_A \cap I_B| \geq n_0$, at each such iteration, the correct path is extended by at least one edge so we are guaranteed that by round n, each of the parties has sent all its edges on the correct path.

Analysis

We just showed that both parties will have sent all the correct edges by the end of the protocol. Now, we establish that they are able to decode sufficiently long prefixes of the sent message to recover the correct path. Let i be the first round for which $s_A[1\cdots i] \cup s_B[1\cdots i]$ contains the entire correct path. If at round n, A is able to decode a word m such that $m[1\cdots i] = s_B[1\cdots i]$, he would have succeeded. However, the fact that he was not able to decode the prefix in earlier rounds implies that there were too many errors. To this end, we have the following lemma,

Lemma 2.4

Let i is the first round for which $s_A[1\cdots i] \cup s_B[1\cdots i]$ contains the entire correct path. Denote $\beta'_A(i-1) = \Delta(s_B[1\cdots(i-1)], r_A[1\cdots(i-1)])$ and $\beta'_B(i-1) = \Delta(s_A[1\cdots(i-1)], r_B[1\cdots(i-1)])$. Then,

$$\beta'_A + \beta'_B > \frac{\alpha}{2}\left(1 - \frac{n_0}{2(i-1)}\right)$$

Analysis

Proof

WLOG, let $\beta'_A \geq \beta'_B$ (A experiences more noise). From the way i is defined, it holds that up to round $i-1$, there were less than $n_0/2 = \epsilon n/2$ rounds where A extended the path by one edge. Lemma 2.3 suggests that $(1 - \frac{2\beta'_A}{\alpha})(i-1) < \epsilon n/2 \implies \beta'_A > \frac{\alpha}{2}\left(1 - \frac{\epsilon n}{2(i-1)}\right)$. Therefore, there were at least $\beta'_A(i-1) > \frac{\alpha}{2}(i - 1 - \frac{n_0}{2})$ corruptions before round i. For rounds $[i \cdots n]$, we are left with a corruption budget of,

$$2n\left(\frac{1}{4} - \epsilon\right) - \frac{\alpha}{2}\left(i - 1 - \frac{\epsilon n}{2}\right) = n\left(\frac{1-\epsilon}{2} - \frac{3\epsilon}{2}\right) - \frac{\alpha}{2}(i-1) + \frac{\alpha\epsilon n}{4}$$
$$< \frac{\alpha(n - i + 1)}{2}$$

which is too small to prevent A from correctly decoding a prefix of length i of s_B. Therefore, A will be able to output the correct prefix.

References


Communication Complexity - Kushilevitz, Nisan


Coding for Interactive Communication: A Survey - Ran Gelles


Tutorial on Coding for Interactive Communication: Introduction and Overview : https://www.youtube.com/watch?v=PhqE5Rc8mFY