List and Unique Coding for Interactive Communication in the Presence of Adversarial Noise

Mark Braverman∗and Klim Efremenko†

Abstract

In this paper we extend the notion of list-decoding to the setting of interactive communication and study its limits. In particular, we show that any protocol can be encoded, with a constant rate, into a list-decodable protocol which is resilient to a noise rate of up to $\frac{1}{2} - \varepsilon$, and that this is tight.

Using our list-decodable construction, we study a more nuanced model of noise where the adversary can corrupt up to a fraction $\alpha$ Alice’s communication and up to a fraction $\beta$ of Bob’s communication. We use list-decoding in order to fully characterize the region $\mathcal{R}_U$ of pairs $(\alpha, \beta)$ for which unique decoding with a constant rate is possible. The region $\mathcal{R}_U$ turns out to be quite unusual in its shape. In particular, it is bounded by a piecewise-differentiable curve with infinitely many pieces. We show that outside this region, the rate must be exponential. This suggests that in some error regimes, list-decoding is necessary for optimal unique decoding. We also consider the setting where only one party of the communication must output the correct answer. We precisely characterize the region of all pairs $(\alpha, \beta)$ for which one-sided unique decoding is possible in a way that Alice will output the correct answer.

1 Introduction

We consider the problem of interactive computation in the presence of adversarial errors. In this setting Alice and Bob want to perform a computation over a channel utilizing an alphabet $\Sigma$, which is affected by an adversarial noise of rate $\eta$. In other words, if the total number of symbols transmitted by Alice and Bob is $N$ (which is known a priori to all the participants), then the adversary is allowed to corrupt at most $\eta N$ symbols of the transmission. The goal is to provide a scheme that can simulate any communication protocol in an error-resilient way.

∗Department of Computer Science, Princeton University. Research supported in part by an Alfred P. Sloan Fellowship, an NSF CAREER award (CCF-1149888), NSF CCF-0832797, a Turing Centenary Fellowship, and a Packard Fellowship in Science and Engineering (mbraverm@cs.princeton.edu).

†Department of Computer Science, University of Chicago. Research supported by a Simons Fellowship in Theoretical Computer Science. Part of this work was done while KE was a member of the Institute for Advanced Studies and funded by NSF CCF-0832797 and DMS-0835373 (klinefrem@gmail.com).
The noninteractive version of the problem is the well-studied problem of encoding a message $M \in \Sigma^N$ with an error-correcting code $C : \Sigma^N \rightarrow \Sigma_2^{N'}$ resilient to adversarial errors. To be resilient to errors of rate $\eta$, we need the Hamming distance between each two codewords $C(M_1)$ and $C(M_2)$ to be sufficiently well spaced, so that corrupt versions of these words can be recovered correctly. Specifically, we need $d_H(C(M_1), C(M_2)) > 2\eta N'$ for all $M_1, M_2 \in \Sigma^N$. A code $C$ is said to be good if it has a constant rate: $N' = \tilde{O}(N)$ and $\log |\Sigma_2| = O(\log |\Sigma|)$; in other words, a good code stretches the input only by a constant factor. The most interesting case studied is when $|\Sigma| = O(1)$. For any $\eta = 1/2 - \varepsilon < 1/2$, a simple probabilistic argument shows that there exist good codes against adversarial errors of rate $\eta$. There are several well-known constructions of good codes.

Once the error rate $\eta$ exceeds 1/2, there is no hope of recovering from a fraction $\eta$ of errors, since for any $M_1, M_2$ there is a message $\tilde{M}$ such that $d_H(C(M_1), \tilde{M}) \leq \eta N'$ and $d_H(C(M_2), \tilde{M}) \leq \eta N'$, which means that $\tilde{M}$ could be a $\eta N'$-corrupted encoding of either $M_1$ and $M_2$. Nonetheless, using list-decoding, it is possible to recover from error rates exceeding 1/2. A code is said to be $\eta$-list-decodable with list of size $L$ if for every word $\tilde{M} \in \Sigma_2^{N'}$, the number of codewords within relative distance $\eta$ from that word is at most $L$. The notion of list-decoding dates back to works by Elias [Eli57] and Wozencraft [Woz58] in the 1950s. Once again, good list-decodable codes for any $\eta = 1 - \varepsilon < 1$ can be shown to exist probabilistically. Moreover, efficient constructions of list-decodable codes exist, and have numerous applications [Gur04].

In the interactive setting, it is not at all obvious that good error correction is possible against adversarial substitution errors of any rate. Note that any attempt to apply standard error-correcting codes round by round are bound to fail, since all the adversary has to do to derail the execution of the protocol is to completely corrupt a single round. Therefore, a subconstant error rate of $1/r$ suffices to foil an $r$-round protocol protected with a round-by-round code. In a striking work, Schulman [Sch96] showed that there exist good error-correcting codes against errors of rate $\eta < 1/240$. This work introduced the tree code primitive, variations on which have been used in follow-up works, including the present one. Informally, the tree code combines two desirable properties: (1) It is an “online” code, so the $i$th symbol of the encoding can be computed from the first $i$ symbols of the original word, thus allowing the encoding of the conversation to be computed as it progresses; (2) its error-correction properties are such that two messages are encoded to two codewords, which are far from each other. How far apart these two codewords are depends on the first place where two messages are different.

Interest in interactive error correction has been renewed recently, with Braverman and Rao [BR11] showing that the error rate that can be tolerated can be improved to $\eta < 1/4$. The constructions of both [Sch96] and [BR11] are not computationally efficient. A series of recent works made significant progress toward making interactive error correction computationally efficient [GMS11, Bra12, BK12, FGOS12, BN13], in some cases by restricting the error parameter or augmenting or restricting the model. In particular, the work of Brakerski and Kalai [BK12] shows how to implement interactive error correction efficiently for $\eta < 1/16$ with a beautiful scheme that uses private (nonshared) randomness.
In this paper we will consider only robust protocols: protocols in which, at all times, the person whose turn it is to speak is known to both Alice and Bob, regardless of the number of errors introduced into the communication so far (even if this amount exceeds the adversary’s budget). It is not hard to see \cite{BR11} that robustness is equivalent to the property that the identity of the speaker at round \( i \) at any execution of the protocol depends only on \( i \). Nonrobust protocols have recently received some attention. In particular, very recently \cite{AGS13} and \cite{GHS13} considered nonrobust models and concluded that, indeed nonadaptive protocols can withstand a higher error rate. It would be interesting to combine these results, particularly those from \cite{GHS13}, with ours to calculate the tight error-rate region for the nonrobust case.

In this work we investigate the limits of error rates that can be corrected in the interactive setting. Further, we develop the notion of interactive list decoding, which is the list analogue of interactive error correction. Its definition is quite straightforward: after the execution of a protocol, each party will output a constant-size list of possible original conversations. If the fraction of errors has not exceeded \( \eta \), each list will contain the intended conversation. We show that constant-rate interactive list-decodable coding is possible for all error rates \( \eta < \frac{1}{2} \), which we show to be the best error tolerance we can hope for.

Moreover, interactive list-decoding turns out to be the right tool for giving tight bounds on the error rates we can tolerate in the unique decoding setting. In the interactive setting, it is natural to consider pairs \((\alpha, \beta)\) of error rates, with \( \alpha \) representing the fraction of Alice’s communication that may be corrupted and \( \beta \) representing the fraction of Bob’s communication that may be corrupted. Using our list-decoding results, we are able to give a precise characterization of the region \( R_U \) of pairs \((\alpha, \beta)\) of error rates for which constant-rate unique decoding is possible. Previously, by the construction of Braverman and Rao \cite{BR11}, unique decoding has been known to be possible when \( \alpha + \beta < \frac{1}{2} \). At the same time, it is easy to see that unique decoding is impossible when \( \alpha \geq \frac{1}{2} \) or \( \beta \geq \frac{1}{2} \). The region \( R_U \) turns out to be quite unusual in its shape. In particular, it is bounded by a piecewise-differentiable curve with infinitely many pieces. When Alice and Bob are affected by an equal amount of error, the intersection of \( R_U \) with the line \( \alpha = \beta \) is the region \( \{(\alpha, \alpha) : \alpha < \frac{1}{3}\} \). Thus we can handle error of up to \( \frac{1}{3} - \varepsilon \) affecting each of Alice and Bob. Previously, only a lower bound of \( \frac{1}{4} - \varepsilon \) and an upper bound of \( \frac{1}{2} \) were known \cite{BR11}.

**Recent Related Works**

We would like to mention very recent related works of Ghaffari, Haeupler, and Sudan \cite{GHS13}, \cite{GH13}, which were developed independently of the present paper. Although these works are closely related to ours in subject matter, the results in these works are almost completely orthogonal. Whereas we study the error model where two parties have different amounts of noise, the main question of \cite{GHS13} was to find the maximal amount of total noise it is possible to tolerate if we allow for players to decide who is speaking next adaptively based on previous communication. The common part of this work and the works \cite{GHS13}, \cite{GH13} is the notion of list-decodable interactive codes, which has been developed independently. While the key definitions are, as expected, similar, there are some differences between the two.
lines of work. All the codes we consider in our work are constant rate, that is, an \( n \)-symbol interactive protocol is always encoded into an \( O(n) \)-symbol protocol over a constant-size alphabet. The codes in [GHS13] convert \( n \) symbols into \( n^2 \) symbols, and the improved construction in [GH13] converts \( n \) symbols into \( n \cdot 2^{O\left(\log^* n \cdot \log \log^* n\right)} \). Our analysis is more detailed in terms of the error region (we analyze the pairs of error rates that we can tolerate, not only their sum). On the other hand, our scheme is not computationally efficient, whereas the scheme of [GH13] is. More excitingly, it appears that by combining [GH13] with the present paper, one obtains efficient, constant-rate schemes with optimal error dependence.\(^1\) The question of finding the optimal-rate interactive error correcting schemes remains wide open.

Main Results

list-decoding. Our first set of results deals with list-decoding interactive protocols. We say that a protocol can handle \((\alpha, \beta)\) adversarial noise if an adversary can corrupt up to \(\alpha\) fraction of Alice’s messages and up to \(\beta\) fraction of Bob’s messages. For a formal definition, see Section 4.

Theorem 1. For each \(\varepsilon > 0\) and for every protocol \(\pi\), there exists another protocol \(\pi'\), with \(CC(\pi') = O_\varepsilon(CC(\pi))\), that is resilient \((\alpha, \beta)\) adversarial noise for all \(\alpha + \beta < 1 - \varepsilon\). The protocol \(\pi'(x, y)\) outputs a list of size \(O_\varepsilon(1)\) of transcripts such that \(\pi(x, y)\) is on the list.

On the other hand, we show that for each pair \((\alpha, \beta)\) with \(\alpha + \beta \geq 1\), list-decoding is impossible.

Theorem 2. For every \(\alpha, \beta\) such that \(\alpha + \beta \geq 1\), let \(\pi\) be a protocol that is resilient to \((\alpha, \beta)\) adversarial noise and that solves list-decoding problem of the Pointer Jumping Problem of depth \(T\) with list of size \(\exp(o(T))\). Then \(CC(\pi) = \exp(\Omega(T))\).

Note that the special case of Theorem 2, where \((\alpha, \beta) = (1, 0)\), is trivial, since in this case no useful communication is transmitted by Alice.

We can give an even tighter result by considering a slightly generalized error notion in Theorem 1. Let us consider only standard protocols \(\pi'\), where Alice and Bob send one message at a time in an alternating fashion. We can partition such a \(\pi'\) into blocks of two symbols, the first one being sent by Alice and the second by Bob. We say that a block is corrupted if the transmission of either symbol in the block is corrupted. Let \(\eta\) be the fraction of blocks the adversary corrupts. Note that it is always the case that \(\max(\alpha, \beta) \leq \eta \leq \alpha + \beta\).

We show that we can handle \(\eta\)-symmetric noise (for an exact definition of symmetric noise, see Section 4) up to \(1 - \varepsilon\), matching the one-way list-decoding bounds.

Theorem 3. For each \(\eta < 1\) and for every protocol \(\pi\), there exists another protocol \(\pi'\) with \(CC(\pi') = O_\eta(CC(\pi))\) which is resilient \(\eta\)-symmetric noise. The protocol \(\pi'(x, y)\) outputs a list of size \(O(\frac{1}{1-\eta})\) of transcripts such that \(\pi(x, y)\) is in the list.

Note that since \(\eta \leq \alpha + \beta\), Theorem 3 implies Theorem 1.

\(^1\)Haeupler, personal communication.
Unique decoding. Next, we turn our attention to the problem of unique decoding. In the unique decoding setting, at the end of the execution of $\pi'$, Alice and Bob need to be able to correctly recover the original protocol transcript $\pi$. In Section 13 we also consider the asymmetric case, where only Alice needs to recover $\pi$ uniquely.

We study the set $\mathcal{R}_U$ of pairs $(\alpha, \beta)$ for which unique decoding is possible. The set $\mathcal{R}_U$ is rather peculiar, and is defined as follows. Let

$$L_2(\alpha) := \frac{1}{2} \left( 1 - \frac{1}{(1 + \{\frac{1}{\alpha}\}) \cdot 2^{\lfloor \frac{1}{\alpha} \rfloor - 1} - 1} \right),$$

where $\{x\} := x - \lfloor x \rfloor$ is the fractional part of $x$. Then the unique decoding region is defined by

$$\mathcal{R}_U := \left\{ (\alpha, \beta) : (\alpha \leq \frac{1}{3} \text{ and } \beta < L_2(\alpha)) \text{ or } (\beta \leq \frac{1}{3} \text{ and } \alpha < L_2(\beta)) \right\}. \tag{1}$$

It is plotted in Figure 1. Note that $L_2$ is continuous and piecewise differentiable with infinitely many pieces. Also, $L_2(\frac{1}{3}) = \frac{1}{3}$, with $(\frac{1}{3}, \frac{1}{3})$ being the intersection point between the boundary $\partial \mathcal{R}_U$ and the line $(\alpha, \alpha)$.

**Theorem 4.** For each $(\alpha, \beta) \in \mathcal{R}_U$ and for every protocol $\pi$, there exists another protocol $\pi'$ with $CC(\pi') = O_{\alpha, \beta}(CC(\pi))$, which is resilient $(\alpha, \beta)$ adversarial noise such that $\pi'(x, y)$ outputs transcript of $\pi(x, y)$.

Theorem 4 is stronger than the main upper bound of [BR11], which shows only how to deal with $(\alpha, \beta)$ in the subregion $\alpha + \beta < 1/2$. More importantly, Theorem 4 turns out to be essentially tight, as shown in the following theorem. Let $\overline{\mathcal{R}}_U$ be the closure of $\mathcal{R}_U$, that is the union of $\mathcal{R}_U$ and its boundary. The Pointer Jumping Problem can be viewed as the generic communication complexity problem and is formally defined in Section 3.

**Theorem 5.** For every $(\alpha, \beta) \notin \overline{\mathcal{R}}_U$ and for every $T$, the following holds. Let $\pi'$ be a protocol resilient to $(\alpha, \beta)$ adversarial noise that solves the Pointer Jumping Problem of depth $T$. Then $CC(\pi') = 2^{\Omega_{\alpha, \beta}(T)}$.

We note that in a noiseless regime, one can solve Pointer Jumping Problem of depth $T$ with communication complexity $T$. Therefore, outside of $\overline{\mathcal{R}}_U$ one cannot perform unique decoding with constant stretch in communication.

**Remark 6.** The lower-bound statement of Theorem 4 is the best we can hope for in the following sense: as long as $\alpha < 1/2$ and $\beta < 1/2$, Alice and Bob can achieve unique-decodable communication with exponential stretch. Alice can use a (noninteractive) good code to send Bob her messages on all potential protocol transcripts, and Bob can do the same. This guarantees correct execution, but causes an exponential stretch in communication.

While Theorem 4 shows that unique decoding is possible in the interior of $\mathcal{R}_U$ and Theorem 5 shows that it is impossible in the interior of its complement, unlike the list-decoding case, we do not establish whether unique decoding is possible for error rates on
the boundary $\partial R_U$. We conjecture that similarly to the list-decoding case, the boundary belongs to the region where decoding is not possible.

**Acknowledgment**

We thank Ran Gelles for for the many insightful comments on an earlier draft of this paper.

## 2 Main Techniques and Discussion

### 2.1 List-Decoding

We start with a discussion of the proof of the positive results on interactive list decoding. First we note that any protocol $\pi$ that sends $T$ bits could be reduced to a Pointer Jumping Problem (PJP) of depth $2T$. PJP is an interactive problem, where Alice receives one edge per node at the odd levels of a binary tree of depth $2T$ and Bob receives one edge per node from the even-level nodes. The task of Alice and Bob is to find a unique path from the root to the leaf using their edges. The overall strategy is similar to that of other recent works: make progress as long as the error is not too high. From the viewpoint of, say, Bob, this means the following: at each step, Bob will try to decode, from Alice’s messages he has received so far, what her (noiseless) messages have been, and sends a response that makes progress on the overall protocol. Thus we make progress as long as Bob decodes Alice’s messages correctly.
It turns out that if Alice and Bob encode their messages using tree codes, this strategy works as long as $\alpha + \beta < 1/2 - \varepsilon$ \cite{BR11}: in other words, in sufficiently many rounds, Alice and Bob will correctly reconstruct each other’s transmissions so far.

What goes wrong when $1/2 < \alpha + \beta < 1$? In this case— for example if $\alpha > 1/2$, as in the case of one-way communication— there is no hope for Bob to be able to ever unambiguously reconstruct Alice’s message. As in the case of one-way list decoding, Bob can still hope to be able to recover a constant-size list $L$ of potential Alice’s communications so far. To make progress, Bob will send $\ell = |L|$ responses to simultaneously make progress on all these communications. Therefore, a list size $\ell$ causes an $\ell$-fold explosion in communication, and $\ell$ needs to be kept constant at (almost) all times. Unlike one-way communication, this approach has a major problem: if Alice sends $\ell$ responses, Bob will decode them and will need to send $\ell$ responses for each of Alice’s responses; thus Bob will need to send $\ell^2$ responses, and this will exponentially blow up communication. In order to overcome this problem, we will follow an approach of Braverman and Rao: instead of sending the next bit and trying to synchronize, Alice and Bob will send edges from a tree. This way, if Alice responds to the wrong message, she will send her correct edge, which will be irrelevant; Bob will know this since he knows that it is impossible to get to this edge using his edges. The main problem in this approach is that in order to encode a single edge, we require $O(T)$ (where $T$ is the depth of the tree) bits. Thus we will add an additional layer of coding. We will first compress our communication and then encode it. Since we are in the list-decoding regime, the adversary can control almost all the communication so basic approach of Braverman and Rao will not work here. A substantial amount of technical work is required to keep the the compression at the rate of $O(1)$ bits per edge.

In order for Alice and Bob to be able to perform list-decoding in interactive settings, we define the analogue of tree codes: a list-tree code. The actual definition is somewhat technical, but informally, a list tree code is a prefix code (i.e., it encodes symbols online) such that for any received word $w$, for almost all rounds, there are at most $\ell$ proper codewords that are $(1 - \varepsilon)$-close to $w$. With correctly selected parameters, a random prefix code is a list-tree code except with an exponentially small probability. This contrasts with the case of ordinary tree codes, for which a random prefix code is unlikely to be a tree code. list-tree codes resemble the potent tree codes of Gelles, Moitra, and Sahai \cite{GMS11}, which can informally be viewed as list-tree codes with list size $\ell = 1 + \varepsilon$.

To summarize, in this paper we are going to have two levels of coding: the first level is compression, which will allow us to take a list of edges from the PJP and to compress them so that, on average, we will use $O(1)$ bits per edge. We want to note that it is easy to see this approach cannot work for any set of edges; thus we will need to chose edges that we are sending very carefully. The second level of coding will be coding resilient to noise; that is, we are going to encode the compressed set of edges with a list-tree code, which will allow us to assert that if the noise rate is not too high, then Bob will get the correct answer in his list. We mention here that we need to carefully adjust the definition of list-tree code in order to make the compression to work.
2.2 Unique Decoding

Next we turn our attention to the unique-decoding regime. It appears that understanding (and being able to carry out) interactive list decoding is needed to achieve optimal unique decoding. To illustrate this point, consider an attempt to break the \( (\frac{1}{4}, \frac{1}{4}) \)-barrier from previous (nonlist decoding) works. To be specific, let \((\alpha, \beta) = (\frac{1}{4}, \frac{1}{4})\). The adversary can corrupt half of Alice’s messages in the first half of the protocol, and half of Bob’s messages in the second half of the protocol. It is easy to see that in the first half of the execution, unique decoding by Bob is not possible, since there is \( \frac{1}{2} \)-error on Alice’s messages. Therefore, if the parties wait until they can decode uniquely, they will not make any progress in the first half of the protocol. Similarly, they will also not make progress in the second half. On the other hand, with list-decoding, in this scenario we can achieve that by the end of the first part of the protocol Alice has decoded the transcript \( \pi \), and Bob has at most two candidate transcripts, \( \pi_1 \) and \( \pi_2 \). He can then use the second (noncorrupt) part of Alice’s transmission to decide whether to output \( \pi_1 \) or \( \pi_2 \).

By just using the list-decodable interactive scheme, and having each party output the answer closest to the received transcript, we can already overcome the \( \alpha + \beta < \frac{1}{2} \) barrier, and get an error-correcting scheme that works for \((\alpha, \beta)\) in the region:

\[
\mathcal{R}_2 := \left\{ (\alpha, \beta) : \alpha + 2\beta < 1 \text{ and } 2\alpha + \beta < 1 \right\}.
\]

In particular, \( \mathcal{R}_2 \) covers all \((\alpha, \alpha)\) for \( \alpha < \frac{1}{3} \).

In our main list-decodable scheme, Alice and Bob speak at the same rate throughout the protocol (i.e., by the time Alice communicates a \( p \)-fraction of her messages, Bob communicates a \( p \)-fraction of his messages). It turns out that for some values of \((\alpha, \beta)\) outside of \( \mathcal{R}_2 \), we can still achieve unique decoding by having Alice and Bob alter the relative rates at which they speak throughout the execution of the scheme. Note that our scheme is still robust; therefore, these rates will be predetermined by \((\alpha, \beta)\). For example, if we consider the point \( (\frac{1}{4}, \frac{3}{7} - \varepsilon) \in \mathcal{R}_U \setminus \mathcal{R}_2 \), the unique-decodable protocol will look as follows: by the time Alice communicates \( \frac{1}{4} \) of her messages, Bob will communicate approximately \( \frac{1}{7} \) of his; after that, for the next \( \frac{3}{4} \) of Alice’s messages, Bob will send the remaining \( \frac{6}{7} \) of his communication in a uniform fashion. The most striking feature of this general regime is the complicated shape of the unique-decoding region \( \mathcal{R}_U \), which is the result of the complicated way in which altering relative transmission rates achieves decodability.

2.3 Lower Bounds

Finally, we discuss our matching lower bounds. Going back to the list-decoding setting, consider the case when \( \alpha + \beta \geq 1 \). If Alice and Bob speak at the same rate, the adversary can erase the first \( \alpha \)-fraction of Alice’s communication and the last \( \beta \) fraction of Bob’s communication. This way there is no overlap between the part where Alice speaks and the part where Bob speaks. Therefore, the encoded protocol is equivalent to a two-round protocol between Alice and Bob. There are communication problems that require more than two rounds to execute efficiently (even with a small probability of success). By starting
with such a problem, we see that a list-decodable encoding that can withstand $(\alpha, \beta)$-error must result in a significant communication blowup. The preceding argument fails if Alice and Bob speak at different rates throughout the protocol— for example, if most of Alice’s communication is concentrated early in the execution and most of Bob’s communication is concentrated late. Using a slightly more complicated argument, we can show that any protocol resilient to $(\alpha, \beta)$-noise with $\alpha + \beta \geq 1$ can be simulated by a three round protocol, leading to a similar contradiction.

Now let us consider the problem that Alice and Bob speak at the same rate. Then adversary can chose either Alice or Bob and try to corrupt his output. Say adversary chose that he wants to corrupt output of Alice. In this case first $\alpha$ fraction of communication he can just erase all the communication of Alice. If protocol can not be solved in one round then after first $\alpha$ fraction of communication there are still two possible inputs for Bob which are consistent with this communication. Therefore adversary can every round toss a coin and play according to one of this two inputs for Bob. This way Alice will not be able to distinguish between this two inputs and adversary corrupts only $\frac{1-\alpha}{2}$ of communication of Bob. Thus we see that $\alpha + 2\beta > 1$ then no unique decoding is possible. When Alice and Bob does not speak at the same rate this argument is not valid any more. The main technical challenge is to adjust the above argument when Alice and Bob speak at different rates. By adjusting this argument to different rates we will get the region $R_U$.

Let us now try to give a brief idea of the region $R_U$ from point of view of upper bounds. Consider the simpler scheme when only Alice is interested in the correct answer. Let us assume that adversary can corrupt up-to $\alpha$ fraction of Alice’s communication. In this case in order to corrupt Alice’s answer, the adversary can (and we proof that this is essentially the only thing it can do) pick some point of the communication $s$. Before $s$, the adversary will corrupt enough communication of Alice and Bob so that Alice and Bob will not be able to perform any interactive communication at all. After $s$, the adversary will corrupt half of the communication of Bob such that Alice will output wrong answer (i.e. will be left with two potential answers without a way of determining which one of the two to output). From this one can see that adversary will always be interested in corrupting Alice’s communication before before point $s$. By writing linear inequalities we obtain that optimal scheme for Bob to talk is to increase his rate relative to Alice’s by a factor of 2 each $\alpha$ fraction of communication of Alice. Note that when $\alpha^{-1}$ is not an integer the number of $\alpha$-fraction intervals is not an integer, and the last interval gets truncated. This is where the non-smoothness of $R_u$ comes from.

3 Communication protocols, pointer jumping and errors

For sake of completeness we put here section from [BR11].

Given inputs $x, y$ from some domain, a deterministic protocol with alphabet $\Sigma$ is a rooted tree where every internal vertex has $|\Sigma|$ children, each corresponding to an element of the
alphabet. Every non-leaf vertex in the tree \( v \) is associated with a function \( f_v(x) \) (or \( f_v(y) \)) that maps one of the inputs to an element of \( \Sigma \). The outcome of the protocol on inputs \( x, y \) is the unique leaf that is reached by first evaluating the function associated with the root, then evaluating the function associated with the child obtained by the first evaluation, and so on. The depth of the tree, \( T \) is called the communication complexity of the protocol. Two parties who each have access to just one of the inputs can compute the outcome of the protocol by communicating at most \( T \) symbols to each other, in the obvious way.

In this paper, we only consider deterministic communication protocols, since our results easily extend to the case of randomized protocols by viewing a randomized protocol as a distribution over deterministic protocols.

Let \( T \) be a rooted \( s \)-ary tree of depth \( T \). Let \( \mathcal{X} \) denote the set of edges leaving vertices at even depth, and \( \mathcal{Y} \) denote the set of edges leaving vertices at odd depth. Given any set \( A \) of edges in the tree, we say that \( A \) is consistent if \( A \) contains at most one edge leaving every vertex of the tree. We write \( v(A) \) to denote the unique vertex of maximal depth that is reachable from the root using the edges of \( A \) and \( e(A) \) will be the last edge in this path. If \( v(A) \) is the root then \( e(A) \) will be dummy edge connected to the root.

Let \( X \subset \mathcal{X}, Y \subset \mathcal{Y} \) be consistent subsets. Then observe that \( X \cup Y \) is also consistent. In the pointer jumping problem, the two parties are given such sets \( X, Y \) and the goal is compute \( v(X \cup Y) \). Since every communication protocol with communication complexity \( t \) bits can be reduced to solving pointer jumping on a tree of depth \( 2t \), it will suffice for us to find a protocol that can compute \( v(X \cup Y) \) even if there are transmission errors.

In this paper, we define communication protocols that are resilient to transmission errors. In our protocols, every vertex at odd depth will be labeled by a function of the first party’s input, and every vertex at even depth will be labeled by a function of the second party’s input. Given such a protocol, each party runs it as above, using her transmissions and the messages she receives to determine her current position in the protocol tree. In general, each party will conclude the protocol at a different leaf. We are going to consider in this paper 3 problems: First one is: pointer jumping problem where both Alice and Bob have to compute \( v(X \cup Y) \). Second one is: one-way pointer jumping problem where at the end of the protocol only Alice has to know \( v(X \cup Y) \) and the last one is: list-decodable pointer jumping problem where at the end of the protocol both Alice and Bob has to output small list of nodes such that \( v(X \cup Y) \) is in the list.

Notation 7. We will denote by \( PJP(T, s) \) the pointer jumping problem of depth \( T \) and alphabet size \( s \).

**Theorem 8.** \([NW93]\) For every \( k < T \) there exists a function \( f(x, y) \) which is solvable in \( T \) communication complexity, but every \( k \) round protocol \( \pi(x, y) \) which computes \( f(x, y) \) with probability at least \( \frac{2}{3} \) has communication complexity at least \( \Omega(\exp(\Omega(T/k^2))) \).

**Proof.** Set \( T = O(\log n/k) \) and use Theorems 2.3, 2.6 from \([NW93]\). \( \square \)

**Corollary 9.** For every \( \varepsilon, k < T \), if protocol \( \pi \) solves one way \( PJP(2T, 2) \) in \( k - 4 \) rounds with probability \( \varepsilon \) then the communication complexity of \( \pi \) is at least \( \Omega(\varepsilon \cdot \exp(\Omega_k(T))) \)
Proof. Let us assume that we have protocol $\pi$ which solves one way $PJP(2T, 2)$ in $k - 4$ rounds with probability $\varepsilon$. Let us show that we can solve $PJP(2T, 2)$ in $k$ rounds with probability $2^{-3}$ and with communication complexity $O(CC(\pi) + T \varepsilon)$. Alice and Bob can execute protocol $\pi$ in parallel $t = 10 / \varepsilon$ times. Let us assume that $v_1, v_2, \ldots, v_t$ be the leaves that Alice got on these executions. Then with probability at least 0.9 one of these leaves is the correct answer. If one of the $v_i$ is the correct answer we can find it in 4 rounds and with $O(T \varepsilon)$ communication. It is easy to see that if $v$ consistent with Alice’s input and with Bob’s input then $v$ is the correct output. Then Alice and Bob will send all his leaves consistent with his input and out of these leaves they will see which one is the correct answer and tell it to an other party.

Since $PJP(2T, 2)$ is complete for communication complexity our protocol can compute function $f(x, y)$ in $k$ rounds with probability $2^{-3}$. Therefore from Theorem 8 it follows that $O(CC(\pi) + T \varepsilon) \geq \Omega((\exp(\Omega(T)) k^2)$. Therefore, $CC(\pi) \geq \Omega(\varepsilon \exp(\Omega(T)))$. □

4 Definition of an adversarial error protocol

Let us denote by $D(\Sigma)$ to be the set of probability distributions over $\Sigma$. For $x, y \in D(\Sigma)$ we will define $d(x, y) = \frac{1}{2} \sum_{\sigma \in \Sigma} |x(\sigma) - y(\sigma)|$. In this paper we will consider the scenario when the adversary can corrupt fraction of symbol i.e., each party sends symbol, but the receiver may receive a distribution over symbols. The amount of error accrued to the adversary if the sent symbol was $a$ and the received distribution $b$ is: $1 - b(a)$. For simplicity of notation we will identify $\Sigma$ with probability distributions supported on single symbol.

Remark 10. Note that this model is stronger than model when adversary can only replace symbol by another symbol. Since replacing symbol by another symbol in this model is equivalent to just sending distribution supported on this symbol (observe that for $a \neq b$, $d(a, b) = 1$). This model is closely related to the concept of soft-decoding in coding theory. The main reason we want to work with this notion, however, is not because it is more general, but because it actually streamlines notation and proofs, making them simpler.

For $x, y \in D(\Sigma)^m$ we can define $d(x, y) = \sum_{i=1}^{m} d(x[i], y[i])$ we will also denote by $agr(x, y) = m - d(x, y)$. Note that if $x, y \in \Sigma^m$ then $d$ is exactly hamming distance between $x, y$.

Definition 11. We say that a protocol $\pi'$ list-decodes $\pi$ with list of size $L$, if for every input $x, y$ both parties of the protocol of $\pi'(x, y)$ output list of size at most $L$ with $\pi(x, y)$ in the list.

Definition 12. We say that a protocol $\pi$ one sided protocol if its correctness depends only on the output of one party.

In this paper we will assume that all our protocols are oblivious i.e., who is speaking at round $i$ depends only on $i$ and not on randomness or input of the protocol $x, y$. Alphabet of our protocols will be $\Sigma_{out}$. We will say that communication complexity of the protocol is
the maximal over all inputs of bits send by our protocol. We will also have alphabet $\Sigma_{out}$ over which our protocol is executed. Alice(Bob) communication complexity $n_A(n_B)$ is the maximal number of symbols(from $\Sigma_{out}$) sent by Alice(Bob).

In this section we would like to discuss several variants of definitions of protocols resilient to adversarial error. We will assume in this paper that all our protocols are are deterministic.

Informally Communication Protocol Resilient to $(\alpha, \beta)$ noise is protocol which output correct answer when adversary can change $\alpha$ fraction of symbols sent by Alice and $\beta$ fraction of symbols sent by Bob.

**Definition 13** (Communication Protocols Resilient to $(\alpha, \beta)$ noise). Assume that we have deterministic protocol $\pi(x, y)$. For $w_A \in \Sigma_{out}^n$ let $\pi_A(x)(w_A)$ be a messages sent by Alice when she received $w_A$ from Bob. We say that protocol computing $f(x, y)$ is resilient to $(\alpha, \beta)$ noise if for every input $x, y$ and for every $w_A \in D(\Sigma_{out})^n_B, w_B \in D(\Sigma_{out})^n_A$ such that

$$d(\pi_A(x)(w_A), w_B) \leq \alpha n_A,$$

i.e., at most $\alpha$ fraction of messages sent by Alice is corrupted. And the same for Bob:

$$d(\pi_B(y)(w_B), w_A) \leq \beta n_B.$$

It holds that both $\pi_A(x)(w_A), \pi_B(y)(w_B)$ computes $f(x, y)$.

In case that we allow more than $\frac{1}{2}$ noise of total communication we have a problem since Eve can corrupt all the communication of one of the parties. Therefore we will limit Eve’s power to prevent this scenario. We will divide our communication to blocks of two so that in each block both both Alice and Bob exchange messages. We will say that some block is corrupted if message of one of the parties is corrupted. That is in this case if Eve decides to corrupt the communication of Alice in some block she can corrupt all the communication of Bob in this block at the same cost. Let us first give formal definitions:

**Definition 14** (Communication Protocols Resilinet to $(\alpha)$-symmetric noise). In this definition we will consider only protocols where at each even round Alice speak and at odd rounds Bob speaks. The protocol runs for $2n$ rounds.

We say that protocol performing task $f(x, y)$ is resilient to $\alpha$-symmetric noise if for every input $x, y$ and for every $w_A \in D(\Sigma_{out})^n, w_B \in D(\Sigma_{out})^n$, let us define $c_A = \pi_A(x)(w_A)$ be an output of Alice on input $w_A$ and $c_B = \pi_B(y)(w_B)$ be an output of Bob.

$$\sum_{i=1}^{n} \max(d(w_B[i], c_A[i]), d(w_A[i], c_B[i])) \leq \alpha n ,$$

Then Alice and Bob will correctly compute $f(x, y)$.
Part I  
List Decoding

5  List Tree Codes

Definition 15. A prefix code $C : \Sigma^n_{in} \rightarrow \Sigma^m_{out}$ is a code such that $C(x)_i$ depends only on $x[1..i]$. By abuse of notation we will also write for $i < n, x \in \Sigma^n_{in}, C(x) \in \Sigma^m_{out}$.

Definition 16. The suffix distance between two strings $x, y \in D(\Sigma)^m$ is defined as

$$\delta_s(x, y) = \max_{i=1..m} d(x[i..m], y[i..m]) / (m - i + 1).$$

In other words suffix distance is the largest relative distance between corresponding suffixes of two words.

Now we will define analogue of the list in one-way communication.

Definition 17. For every $w \in D(\Sigma)^n_{out}$ the $i$-th level $\epsilon$-list of $w$ under $C$ is given by

$$\text{List}_i(w, C, \epsilon) := \{ x \in \Sigma^n_{in} : \delta_s(C(x), w[1..i]) \leq 1 - \epsilon \}.$$ 

The $\epsilon$-list of $w$ under $C$ is given by

$$\text{List}(w, C, \epsilon) := \bigcup_{i=1}^{\vert w \vert} \text{List}_i(w, C, \epsilon).$$

We also define

$$\text{PrefixList}(w, C, \epsilon) := \{ y \in \Sigma^*_{in} : y = x[1..k] \text{ for some } k \text{ and some } x \in \text{List}(w, C, \epsilon) \}.$$ 

We will identify all codewords $\Sigma^*_n$ with a full $\Sigma_{in}$-ary tree. Where codewords of length $d$ will be at depth $d$. We will identify words $x \in \Sigma^n_{in}$ with node in a tree, where $x \in \Sigma^n_{in}$ will be a father of $x \circ c \in \Sigma^{n+1}_{in}$. In this case we will write $c$ on the edge going from $x$ to $x \circ c$. The tree $\text{PrefixList}$ is the rooted subtree that spans all the nodes in $\text{List}$. In the Figure 1 we give an example of $\text{PrefixList}$. Note that in this example although $\bigcup \text{List}_i = 5$. The size of PrefixList is 9.

For any rooted subtree $PL$ of full $\Sigma_{in}$-ary tree we will denote by $w(PL)$ the tree where we write $w[i]$ on all edges at depth $i$ and $C(PL)$ just a restriction of $C$ to $PL$. We will always consider trees in this paper as one-way graphs directed from root downwards. If we have some tree $PL$ and two different labelings of nodes $w(PL), C(PL)$ than $d(w(PL), C(PL)) = \sum_{e \in PL} d(w(e), C(e))$ and $\text{agr}(w(PL), C(PL)) = \vert PL \vert - d(w(PL), C(PL))$. Note that if $P$ is some path in the tree ending at $v$. Then $C(P)$ is just labeling of this path. The string that we get from this labeling is suffix of $C(v)$ since $C(v)$ is in fact labeling of path from root $v$. The following lemma is a good excise on above definitions.
Lemma 18. Let $v$ be a leaf in the the PrefixList($w, C, \varepsilon$) then for every path $P$ ending at $v$ we have $agr(C(P), w(P)) > \varepsilon |P|$

Proof. By definition of PrefixList all its leaves $v$ are in List($w, C, \varepsilon$). Thus for some $i, v \in List_i(w, C, \varepsilon)$. By definition of List$_i$ we have $\delta_s(C(v), w) \leq 1 - \varepsilon$. If $|P| = k$ then also by definition $C(P) = C(v)[i - k - 1, i]$. Thus from definition of $\delta_s$ the lemma follows.

We will define the size of the tree to be number of its vertexes. Now we are ready to define the main object of this paper.

Definition 19. An $(\varepsilon, L)$-list tree code $C : \Sigma^n_{in} \to \Sigma^n_{out}$ with average list size $L$ and decoding distance $1 - \varepsilon$ is a prefix code such that for all $w \in D(\Sigma_{out}^n)$ let $PL(w) = \text{PrefixList}(w, C, \varepsilon)$ then

1. $|PL(w)| \leq L \cdot n$.

2. $agr(C(PL), w(PL)) \leq \varepsilon Ln$

As we will see from the next lemma first property is included in the second property. We stated here first property since we are going to use mainly this property of the code. The only reason why we require $w$ to be string of distribution rather than just string of symbols is because this way we can perform soft decoding.

Remark 20. We also want to note that the only reason for defining PrefixList is in order to allow compression for the case the alphabet is of size $O(1)$. For a large alphabet it is enough to require that $|List_i| < \varepsilon'L$ for almost all $i$. 

Figure 1: PrefixTree with $List_2 = \{”hi”, ”by”\}, List_3 = ”bye”, List_4 = ”hell”, List_5 = ”hello”$
Lemma 21. Let \( PL \triangleq \text{PrefixList}(w, C, \varepsilon) \), then \( \text{agr}(C(PL), w(PL)) > \varepsilon|PL| \).

Proof. We will prove this by induction on number of leaves. If \( PL \) has only one leaf \( v \) then \( PL \) is a path from root to \( v \) and the claim follows from the Lemma 18.

Induction step: Let \( PL \) be tree with \( i \) leaves. Let \( P \) be a branch of \( v \) (i.e. path from \( w \) to \( v \) where \( w \) is a first predecessor of \( v \) which has more than one child.) Then \( PL \setminus P \) has one leaf less than \( PL \). Thus from induction we get that

\[
\text{agr}(C(PL \setminus P), w(PL \setminus P)) > \varepsilon (|PL| - |P|).
\]

From Lemma 18 we know that \( \text{agr}(C(P), w(P)) > \varepsilon |P| \). Thus

\[
\text{agr}(C(PL), w(PL)) = \text{agr}(C(PL \setminus P), w(PL \setminus P)) + \text{agr}(C(P), w(P)) > \varepsilon |PL|.
\]

\[\square\]

Theorem 22. For all \( 0 < \varepsilon < 1 \), let \( |\Sigma_{\text{out}}| > (2|\Sigma_{\text{in}}|)^{\frac{3}{2}} \). Then a random prefix code \( C : \Sigma^n_{\text{in}} \to \Sigma^n_{\text{out}} \) is a \((\varepsilon, L)\) list-tree code with \( L = \frac{1}{\varepsilon} + 1 \) with probability at least \( 1 - 2^{-n} \).

Before proving this we will need the following lemma.

Lemma 23. Let \( PL \) be full \( d \)-ary tree then there exists at most \((d + 1)^2s\) rooted subtrees of \( PL \) of size \( s \).

Proof. First we will write our tree as a path where each symbol says to what child to go and \( d + 1 \) symbol will say to go up. Next note if we make DFS on subtree of size \( s \) then we will pass on every edge twice once when we go down and once when we go up. Thus we can write every subtree of size \( s \) by \( 2s \) symbols.

Proof of Theorem 22. First let us note that from Lemma 21 it follows that \( \text{agr}(C(PL), w(PL)) > \varepsilon |PL| \) thus if \( \text{agr}(C(PL), w(PL)) \leq \varepsilon Ln \) it follows that \( |PL| \leq Ln \). Therefore second property of the list tree codes imply the first property.

Let us assume that \( w \) is a word that violates the conditions of the theorem. Then Lemma 21 also imply that if \( PL = PL(w) \) then \( \text{agr}(C(PL), w(PL)) > \varepsilon \max\{|PL|, Ln\} \). Now we are going to prove that w.h.p. \( \text{agr}(C(PL), w(PL)) < \varepsilon \max\{|PL|, Ln\} \) for every subtree \( PL \) and for every \( w \). We can assume w.l.g. \( |PL| \geq Ln \). Let us first fix some subtree \( PL \) of complete \( \Sigma_{\text{in}} \)-ary tree of size \( s \geq Ln \) and \( w \in \Sigma^n_{\text{out}} \) then the next claim together with Lemma 21 gives us bound on probability that the conditions of the list tree code broken for this specific \( w \) and \( PL \).

Claim 24. For every \( w \in \Sigma^n_{\text{out}} \) and tree \( PL \) of size at most \( s \) it holds that

\[
P[\text{agr}(C(PL), w(PL)) \geq \varepsilon s] \leq |\Sigma_{\text{out}}|^{-\varepsilon s} \left(\frac{\varepsilon s}{\varepsilon}\right)^{s} \leq |\Sigma_{\text{out}}|^{-\varepsilon s} 2^{\varepsilon s},
\]

where randomness is taken over the random choice of code \( C \).
Proof. The first inequality follows from union bound where we take union over all possible locations where \( C(PL), w(PL) \) have an agreement. The second inequality follows from the fact that 

\[
2^s = (1 + 1)^s \geq \binom{s}{\varepsilon s}.
\]

Thus using union bound over all possible trees and words \( w \) we get that probability that \( C \) is not list tree code is bounded by

\[
\sum_{s=Ln}^{\infty} |\Sigma_{out}|^{-\varepsilon s} 2^s \# \{ \text{rooted subtrees of size } s \} |\Sigma_{out}|^n
\]

Lemma \[23\] gives us a bound on the number of rooted subtrees of size \( s \). Therefore probability that \( C \) is not list tree code is bounded by:

\[
\sum_{s=Ln}^{\infty} |\Sigma_{out}|^{-\varepsilon s} 2^s (|\Sigma_{in}| + 1) 2^s |\Sigma_{out}|^n
\]

Since \( |\Sigma_{out}| > (2|\Sigma_{in}|)^{1/2} \) above quantity is bounded by:

\[
(2|\Sigma_{in}|)^{1/2} \sum_{s=Ln}^{\infty} (2|\Sigma_{in}|)^{-1/2} 2^s (|\Sigma_{in}| + 1) 2^s \leq 2(2|\Sigma_{in}|)^{1/2} (2|\Sigma_{in}|)^{-1/2} 2^{Ln} (|\Sigma_{in}| + 1) 2^{Ln}
\]

Since \( (2|\Sigma_{in}|)^{-1/2} 2(|\Sigma_{in}| + 1)^2 < 1 \) Recalling that \( L = \frac{1+\varepsilon}{\varepsilon} \) we can bound above as:

\[
2(2|\Sigma_{in}|)^{-1/2} 2^{Ln} (|\Sigma_{in}| + 1) 2^{2s} \leq 2^{-n/\varepsilon} \leq 2^{-n}. 
\]

To see that the theorem holds for all \( w \in D(\Sigma_{out})^n \) rather than \( w \in \Sigma_{out}^n \) observe that for every subtree \( PL \) and for every code \( C(PL) \) the closest codeword \( w \in D(\Sigma_{out})^n \) to \( C(PL) \) is \( w \), where \( w[i] \) is the most frequent symbol of \( C(PL) \) at level \( i \), thus it is in \( \Sigma_{out}^n \). \qed

6 Communication transcript

During our protocol Alice and Bob are going to send edges from the tree \( T \) of the original protocol. In order to describe some specific edge from \( T \) it may take \( O(T) \) bits. Also as in Braverman Rao [BR11] we will need one additional level of encoding that will compress our communication. First idea of the compression is that we are not sending random edges, we never send any edge before we send his grandparent. Therefore we can describe an edge by a link to grandparent and a path from grandparent to an edge. This will solve us the problem for the case that we have alphabet polynomial in the length of our communication since in this case we do not care of the length of the links. In case of constant size alphabet the size of the link is important and therefore we sometimes instead of sending link to grandparent we will send link to cousin who may be much closer to the edge we are sending.

16
Since links to a cousin’s looks little bit mysterious let us try to give brief intuition why it is necessary. Let us consider tree of size $t$. Let us assume that we have subset of nodes of this tree which we call special nodes. Let us assume that every special node has a link pointing to a closest special node predecessor of him. In this case we have no control on the lengths of links. However if size of link is difference in depth between nodes and we allow links to cousin’s then it is not very hard to see that sum of links is bounded by the size of the tree(for more details see Claim 46).

Each entry of our transcript will correspond to some edge in $T$. Every entry $a_i = (r_i, b_i, s_i)$ of the transcript will consist of integer $r_i \in \mathbb{N}$ which will be a pointer to another edge that appeared $r_i$ entries earlier in our transcript. $b_i \in \{0, 1\}$ bit will indicate whether the reference edge is grandparent or cousin$^2$ and $s_i \in \Sigma_T^{\leq 2}$ which will indicate path from grandparent to the edge. We will assume that an integer $k$ takes at most $2 \log k + 2$ bits to encode.

**Procedure of decoding $E(A)$:** Now let us describe formally how we will decode our transcript for $i = 1, \ldots, k$, we will do the following to decode $e_i$ from $a_i = (r_i, b_i, s_i)$:

1. if $r_i = 0$ set $e_i$ to be an edge at depth at most 2 specified by bits $s_i$.
2. if $r_i \geq i$ return error.
3. if $b_i = 0$ set $p_i = e_{i-r_i}$
4. if $b_i = 1$ set $p_i$ to be grandparent of $e_{i-r_i}$
5. set $e_i$ be the edge specified by starting at the child vertex of the edge $p_i$ and then taking (at most) two steps in the tree using the bits $s_i$.

**Procedure Encoding $Add(A, e)$:** Now assume that we want to add an edge $e_i$ to our transcript $a_1, a_2, \ldots, a_{i-1}$. If $e_i$ is at depth at most two we will set $r_i = 0$ and $s_i$ to correspond to path from root to $e_i$. Else we will decode edges $e_1, \ldots e_{i-1}$ from $a_1, \ldots a_{i-1}$. Next we are going to find maximal index $j$ such that $e_j$ is a cousin or grandparent of $e_i$ and then we are going to set $r_i = j - i$ and we will set $b_i = 0$ if $e_j$ is grandparent of $e_i$ and $b_i = 1$ if $e_j$ is cousin of $e_i$. We will set $s_i$ to correspond to path from grandparent of $e_i$ to $e_i$. For the purposes of this procedure, an edge is its own cousin.

Thus for $A = (a_1, a_2, \ldots, a_k)$ we have procedure $E(A)$ which returns edges encoded by $A$ and $Add(A, e)$ which adds edge $e$ to transcript $A$.

**Remark 25.** Note that Procedure Add does not works for any edge $e$, but only for edges at depth at most 2 from already added edges.

We think of the transcript as a stream of bits and we will also have the following functions:

- **Procedure size$(A)$**: which will return the size of the transcript.

$^2$In fact we will need cousins only for a small alphabet
• Procedure **End** − **round**(*A*, *i*): which will set the size of the transcript to be maximum between \(\log \Sigma_{in}^i\) and \(\text{size}(A)\) and padding if necessary transcript to this size by adding some 0 (which will represent no edge).

7 Recovering from errors using a polynomial size alphabet

The goal of this section is to prove the following theorem:

**Theorem 26.** For every \(\varepsilon, c > 0\) there exists a protocol \(\pi\) which is resilient to \(1 - \varepsilon\)-symmetric noise and which solves the problem of list-decoding of \(PJP(T, \Sigma_T)\), where \(|\Sigma_T| = T^c\) for some constant \(c > 0\) with list of size \(O(\frac{1}{\varepsilon})\). Moreover the protocol \(\pi\) runs \(O(\frac{T}{\varepsilon})\) rounds and in each round sends \(O(\varepsilon, c) (\log T)\) bits.

Let \(\varepsilon' = c_1 \varepsilon\) for some small constant \(0 < c_1 < 1\) to be defined later. Define also \(L = \frac{1}{\varepsilon} + 1\). During the protocol we are going to send \(O(T)\) edges and encode them with the transcript defined in previous section. The links in the transcript will be of size at most \(O(T)\) therefore every entry of transcript takes at most \(O(\log T)\) bits. Let us set \(\Sigma_{in}\) to be large enough to hold \(\text{Ent} = \frac{T}{\varepsilon} = O(\frac{1}{\varepsilon})\) entries of the transcript. Thus \(\log |\Sigma_{in}| = O_{\varepsilon, c}(\log T)\). Let \(n = \frac{T}{\varepsilon} = O(\frac{T}{\varepsilon})\) be the number of rounds of the protocol. By Theorem 22 there exist \(C : \Sigma^n_{in} \to \Sigma^n_{out}\) which is \((L, \varepsilon') = (O(\frac{1}{\varepsilon}), O(\varepsilon))\) list tree code with \(\log |\Sigma_{out}| = O_{\varepsilon}(\log |\Sigma_{in}|) = O_{\varepsilon, c}(\log T)\).

**Remark 27.** Note that we have here three alphabets:

- First is the alphabet of the original protocol \(\Sigma_T\) of size \(T^c\).
- Second is \(\Sigma_{in}\) which corresponds to a non-encoded single message that we are going to send during each round.
- The third one is \(\Sigma_{out}\) which corresponds to an alphabet which we are going to send over noisy channel.

Let \(C\) be encoding and for \(w \in \Sigma_{out}\), \(D(w)\) will return \(\text{List}_i(w, C, \varepsilon)\). At every round \(i\) Alice will decode received codeword \(w \in D(\Sigma_{out})^i\) and will get list of possible transcripts of Bob \(B_1, B_2, \ldots B_k\) (Bob will similarly decode Alice’s message). For every such transcript Alice will calculate \(E(B_i)\) edges that Bob sent and send next edge from \(X \subset \mathcal{X}\) by first adding this edge to transcript and then encoding transcript with list tree code.

The protocols of Alice and Bob will be symmetric so we will introduce only protocol for Alice. We will denote by \(w_A, w_B\) received codewords by Alice and Bob correspondently. Alice will maintain transcript \(A\) which will be initialized by empty set. Formally at block \(i\) Alice will,

**Input:** \(w_A \in D(\Sigma_{out})^i\)

1. Calculate \(\text{Decode}(w_A) = \text{List}_i(w_A, C, \varepsilon) = \{B_1, B_2, \ldots, B_k\}\)
2. For \(j = 1, \ldots, \min\{k, \text{Ent}\}\) do
(a) If $E(B_j) \cup E(A)$ has unique path from root and let $e$ be a last edge on this path then $Add(A, e)$

3. $End \cdot round(A, i + 1)$.

4. Send $C(A)[i + 1]$ to Bob.

We will run the protocol for $n$ steps. At the end Alice will construct $PrefixList(w_A, C, \varepsilon)$.

**Definition 28.** For a node $v \in PrefixList(w_A, C, \varepsilon)$ we will call it output node if $E(v) \cup X$ has unique path from root and this path ends at a leaf $\ell$ and none of predecessors of $v$ has this property. The output of the output node is the leaf $\ell$.

In other words, the output node $v$ is the node at which corresponds to some possible output. Let $v, v_s \in PrefixList(w_A, C, \varepsilon)$ such that $v_s$ is a successor of $v$ Let $v$ be node at depth $i$ and $v_s$ at depth $j$ and let $s \in \sum_{out}^{i+1}$ to be string corresponding to path from $v$ to $v_s$. Then $agr(v \rightarrow v_s, w) \equiv agr(s, w[i, \ldots, j])$. For output Alice will do the following:

1. Construct $PrefixList(w_A, C, \varepsilon)$.

2. For every output node $v \in PrefixList(w_A, C, \varepsilon)$

   (a) If exists successor of $v$, $v_s \in PrefixList(w_A, C, \varepsilon)$ such that $agr(v \rightarrow v_s, w) > T$ then output $v$.

In other words Alice will output all answers which corresponds to communication and has at least $T$ places of agreement with $w$ after output node.

**Lemma 29.** The protocol above solves the task of list decoding pointer jumping problem with list size $O(\frac{1}{\varepsilon})$

The rest of the section will prove this lemma. Let us denote by $PL_A = PrefixList(w_A, C, \varepsilon)$ and by $PL_B = PrefixList(w_B, C, \varepsilon)$

**Claim 30.** Output list size of the protocol is at most $O(\frac{1}{\varepsilon})$.

**Proof.** By definition of list tree codes agreement between $C(PL_A)$ and $w_A$ is less than $\varepsilon'L_n$. In order for some node $v$ to be in the output list we require that there will be an agreement of at least $T = \varepsilon'n$, which appears after we reached output node. Note that by definition output nodes can not be successors of each other. Thus each different node from the output list corresponds to a disjoint agreement of at least $T$ between $C(PL_A)$ and $w_A$. Thus the size of the output list can not be larger than $\frac{\varepsilon'L_n}{T} = L = O(\frac{1}{\varepsilon})$. $\square$

Now the rest of the section we will prove that if there is at most $1 - \varepsilon$ error then the correct answer is in the list.

**Definition 31.** Let $P$ be the unique path from the root in $X \cup Y$. We say that we advance in the block $i$ if at this block Alice or Bob adds a new edge from $P$ to his transcript.
Definition 32. We say that the block $i$ is good if:

1. At block $i$ both Alice and Bob decoded correctly each other’s messages i.e., $B_{[1,\ldots,i-1]} \in \text{List}_{i-1}(w_A,C,\varepsilon)$ and $A_{[1,\ldots,i-1]} \in \text{List}_{i-1}(w_B,C,\varepsilon)$.

2. The list size of both Alice and Bob at block $i$ was less than $\text{Ent}$.

Claim 33. At every good block we advance.

Proof. By definition at the good block both Alice has transcript $B$ of Bob and Bob has transcript $A$ of Alice. Moreover, the list in this block is small, and thus Alice will add last edge from $E(B) \cup X$ and Bob will add $E(A) \cup Y$. Let $e$ be a first edge from $P$ which does not appear in $E(A) \cup E(B)$. Let us assume w.l.o.g. that $e \in X$. Then $e$ is a last edge from $E(B) \cup X$. Thus we will advance in this block. \hfill $\square$

Thus now we will need to prove that there are many good blocks. We first we will note that there are not too many blocks which do not satisfy the second condition and then we will prove that there are many blocks which do satisfy the first condition.

Claim 34. There are at most $O(\varepsilon'n)$ blocks which do not satisfy the second condition of being good.

Proof. Note that by definition of $\text{PrefixList}$, $\sum_{i=1}^{n} \text{List}_i(w_A,C,\varepsilon) \leq \text{PrefixList}(w_A,C,\varepsilon)$. Since at every bad block we have that $\text{List}_i(w_A,C,\varepsilon) \geq \text{Ent} = \frac{L}{\varepsilon}$ (or same for Bob.) We have at most

$$\frac{|PL_A| + |PL_B|}{\text{Ent}} \leq \frac{2Ln}{\text{Ent}} = 2\varepsilon'n \leq O(\varepsilon'n)$$

such blocks. \hfill $\square$

Lemma 35. Let $c_A = C(A)$, $c_B = C(B)$ be a messages that Alice (Bob) has sent and $w_A, w_B$ is what they received. Let us assume that:

$$\sum_{i=1}^{n} \max(d(w_B[i], c_A[i]), d(w_A[i], c_B[i])) \leq (1-\varepsilon)n,$$

then there are at least $(\varepsilon - \varepsilon')n$ blocks $i$ at which first condition of being good is satisfied.

The lemma will follow from the following claim.

Claim 36. Let $w \in D(\Sigma)^n, c \in \Sigma^n \; d(w,c) < (1-\beta)n$. Define the set

$$S_\alpha(n) \triangleq \{i \leq n : \delta_s(w[1,\ldots,i],c[1,\ldots,i]) \leq 1-\alpha\}$$

Then $|S_\alpha(n)| \geq (\beta - \alpha)n$. 
Proof of Lemma 35. Recall that by definition of List, we have that $B[1, \ldots, i]$ in the $List_i(w_A, C, \varepsilon)$ if
\[ \delta_s(w_A[1, \ldots, i], c_B[1, \ldots, i]) \leq 1 - \varepsilon', \]
and the same for Bob. For each $i$ let us define $w[i], c[i]$ to be either $w_B[i], c_A[i]$ or $w_A[i], c_B[i]$ such that $d(w[i], c[i]) = \max(d(w_B[i], c_A[i]), d(w_A[i], c_B[i]))$. Then we are given that $d(w, c) \leq (1 - \varepsilon)n$. Then using Claim 36 we will get that there are at least $(\varepsilon - \varepsilon')n$ indexes $i$ such that $\delta_s(w[1, \ldots, i], c[1, \ldots, i]) \leq 1 - \varepsilon'$. For every such $i$ the first condition holds since by definition of $w, c$ it holds that
\[ \delta_s(w_A[1, \ldots, i], c_B[1, \ldots, i]) \leq \delta_s(w[1, \ldots, i], c[1, \ldots, i]) \leq 1 - \varepsilon', \]
And the same for Bob, i.e.,
\[ \delta_s(w_B[1, \ldots, i], c_A[1, \ldots, i]) \leq \delta_s(w[1, \ldots, i], c[1, \ldots, i]) \leq 1 - \varepsilon'. \]

Proof of Claim 36. Let us first give intuitive proof of the claim. By definition of suffix distance $\delta_s(w, c) < 1 - \alpha$ if there exists a suffix with agreement less than $\alpha$. Thus let us do the following if $\delta_s(w, c) < 1 - \alpha$ just add last index to $S_\alpha$ and proceed to previous index. If $\delta_s(w, c) \geq 1 - \alpha$ then cut suffix with agreement at most $\alpha$. Note that by this procedure we will cut at most $\alpha$ agreement between $w, c$ thus since agreement between $w, c$ is at least $\beta$ it must be that we added at least $(\beta - \alpha)n$ indexes to the list $S_\alpha$.

We will prove the claim by induction. For $n = 1$ the claim trivially follows since in this case if $\beta > \alpha$, $d(w, c) < 1 - \beta < 1 - \alpha$.

If we know claim for all strings up-to $n - 1$ then we separate in two cases if $n \in S_\alpha$ then We have that $|S_\alpha(n)| = |S_\alpha(n - 1)| + 1$, by induction on $w[1, \ldots, n-1], c[1, \ldots, n-1]$ we know that
\[ d(w[1, \ldots, n-1], c[1, \ldots, n-1]) \leq (1 - \beta)n = (1 - \frac{\beta n - 1}{n - 1})(n - 1) \]
Thus $|S_\alpha(n - 1)| \geq \beta n - 1 - \alpha(n - 1) \geq (\beta - \alpha)n - 1$. Therefore $S_\alpha(n) \geq (\beta - \alpha)n$.

If $n \notin S_\alpha$ then by definition there exists an $n'$ such that $d(c[n', \ldots, n], w[n', \ldots, n]) > (n - n' + 1)(1 - \alpha)$ thus
\[ d(w[1, \ldots, n'-1], c[1, n'-1]) < (1 - \beta)n - (n - n' + 1)(1 - \alpha) = (n' - 1) - (\alpha(n' - 1) + (\beta - \alpha)n). \]
By induction on $n' - 1$ we get that
\[ |S_\alpha(n)| \geq |S_\alpha(n' - 1)| \geq \alpha(n' - 1) + (\beta - \alpha)n = (\beta - \alpha)n \]

Be Claim 34 and Lemma 35 we get:
Corollary 37. If our protocol corrupted in at most \((1 - \varepsilon)n\) blocks then there are at least \(\varepsilon n - O(\varepsilon' n)\) good blocks.

Now we are ready to prove that \(v(X \cup Y)\) is in the list.

Claim 38. \(v(X \cup Y)\) is in the list.

Proof. Let \(P \subset PL_A\) be a path in PrefixList that corresponds to the correct path (i.e. it is actually the message that Bob has sent). Let \(e_1\) be a length of \(P\). Then in order to prove that the correct output is in the list we have to show there exists output node on \(P\) at some place \(e_0\) and that agreement of \(P\) and \(w_A\) between \(e_0\) and \(e_1\) is at least \(T\).

If \(c_1 < \frac{1}{2}\) then \(\varepsilon n - T - \varepsilon' n > 0\). Let \(i_0\) be a place such that we have agreement \(\varepsilon n - T - \varepsilon' n\) before \(i_0\) and agreement \(T + \varepsilon' n\) after \(i_0\) i.e.,

\[
\sum_{i=1}^{i_0} \max(d(w_B[i], c_A[i]), d(w_A[i], c_B[i])) = i_0 - \varepsilon n + T + \varepsilon' n ,
\]

There are at least \(\varepsilon n - T - \varepsilon' n - O(\varepsilon' n)\) good blocks before \(i_0\). By definition of \(i_0\) there is at least \(T + \varepsilon' n\) agreement after \(i_0\).

Thus by taking \(c_1\) small enough we will get at least \(T\) good blocks before \(i_0\). Since at every good block we advance we will get to output node after \(T\) good blocks in particular output node with correct answer is before \(i_0\).

Let us show that there exists \(i_1\) such that \(\delta_s(w_A[1, \ldots i_1], c_B[1, \ldots i_1]) \leq 1 - \varepsilon'\) and such that agreement before \(i_1\) is at least \(\varepsilon n - \varepsilon' n\). If \(\delta_s(w_A[1, \ldots n], c_B[1, \ldots n]) \leq 1 - \varepsilon'\) then take \(i_1 = n\) and we are done. Else by definition of \(\delta_s\) there exists an \(i\) such that

\[
d(w_A[i, \ldots n], c_B[i, \ldots n]) > (1 - \varepsilon')(n - i + 1)
\]

Let us take \(i_1 = i_2 - 1\) where \(i_2 
eq 1\) is a minimal value \(i\) satisfying equation above. We claim that \(\delta_s(w_A[1, \ldots i_1], c_B[1, \ldots i_1]) \leq 1 - \varepsilon'\) else there exists \(i_3 < i_2\) such that

\[
d(w_A[i_3, \ldots i_2 - 1], c_B[i_3, \ldots i_2 - 1]) > (1 - \varepsilon')(i_2 - i_3)
\]

Therefore

\[
d(w_A[i_3, \ldots n], c_B[i_3, \ldots n]) =
\]

\[
d(w_A[i_3, \ldots i_2 - 1], c_B[i_3, \ldots i_2 - 1]) + d(w_A[i_2, \ldots n], c_B[i_2, \ldots n]) >
\]

\[
(1 - \varepsilon')(i_2 - i_3 + n - i_2 + 1) = (1 - \varepsilon')(n - i_3 + 1)
\]

Contradiction to minimality of \(i_2\). Note also that by definition of \(i_1\) agreement from \(i_1\) to \(n\) is at most \(\varepsilon' n\). Thus agreement between \(i_0\) which is after output node and \(i_1\) point in PrefixList is at least \(T\). In other words \(agr(w_A[i_0, \ldots i_1], c_B[i_0, \ldots i_1]) \geq T\) and as just proved \(c_A[1, \ldots i_1]\) is in \(PrefixList(w_A, C, \varepsilon)\). Thus the correct codeword should be in the list.
8 Constant alphabet

In this section we want to show interactive coding theorem for constant size alphabet. The main difficulty in extending results from the previous section is that in general link to grandfather may be of size $\log n$. In order we are going to prove that sum of all link is at most $O(n)$ thus on average we are going to send link of size $O(1)$. The goal of this section is to prove the following theorem:

**Theorem 39.** For every $\varepsilon, c > 0$ there exists a protocol $\pi$ is resilient to $1 - \varepsilon$-symmetric noise and which solves the problem of list decoding of $PJP(T, 2)$ with list of size $O(\frac{1}{\varepsilon})$. Moreover the protocol $\pi$ runs in $O(\frac{T}{\varepsilon})$ rounds and in each round sends $O_\varepsilon(1)$ bits.

We start by observing that Theorem 3 is a corollary of this theorem.

**Proof of Theorem 39.** Since Pointer Jumping Problem is complete for communication complexity any protocol $\pi$ with $CC(\pi) = T$ can be converted to Pointer Jumping Problem of depth $2T$. Then from Theorem 39 it follows that there exists a protocol $\pi'$ which solves list decoding Pointer Jumping Problem and resilient to $\eta < 1$ noise with $CC(\pi') = O_\eta(T)$ and with list size $O(\frac{1}{1-\eta})$.

As in previous section define $\varepsilon' = c_1 \varepsilon$ for some small constant $c_1$ to be defined later. Define also $L = \frac{1}{\varepsilon'} + 1$.

In case of constant alphabet we will have $\Sigma_{in}$ to be large enough to contain $Ent = \frac{T}{\varepsilon'}$ transcript entries with link of size $MLSize = \frac{T^2}{\varepsilon^2} = O(\frac{1}{\varepsilon})$. Note that the size of $\Sigma_{in}$ is a constant depending on $\varepsilon$. By Theorem 22 there exist $C : \Sigma_{in}^n \rightarrow \Sigma_{out}^n$ which is $(L, \varepsilon') = (O(\frac{1}{\varepsilon'}), O(\varepsilon'))$ list tree code with $\log |\Sigma_{out}| = O_\varepsilon(\log |\Sigma_{in}|) = O_\varepsilon(1)$.

Let $n = \frac{T}{\varepsilon'} = O(\frac{T}{\varepsilon})$ be the number of rounds of the protocol.

In case we will send the link of size larger than $MLSize$ we will say that we send a long link. In general it may happen that we will need to send a long link which will not fit in the space we dedicated for it. In this case we will send it anyway at cost of not sending other edges we wanted to send in this and maybe several next rounds (up to $\log n$ rounds). We are going to modify our protocol a little bit in order to be able to prove that the sum of all links in our protocol is at most the size of $PrefixList < Ln$. At last we going to conclude that there are at most $O(\varepsilon'n)$ blocks at which we are sending a long link and thus those links will not disturb our protocol too much.

Note that in list decoding regime adversary can corrupt up-to $1 - \varepsilon$ fraction of communication therefore we need to be very careful when we are sending a long links since most of the time we will respond to an adversary communication. We are going to do the following adjustment to our protocol. Note that in Section 7 every edge Alice (or Bob) sent was a response on the decoding of some transcript which on his side corresponds to some node in $PrefixList(w_A, C, \varepsilon)$. In this protocol we are going to remember for every edge in our transcript for which node in $PrefixList$ it corresponds and we will allow only to send edge $e$ at node $v \in PrefixList$ only in case that grandparent of $e$ was sent at predecessor of $v$. As we will see later this way we will prevent from Alice to send too many long links. As we
will see the sum of all links send in this way will approximately the size of the PrefixList.

It may happen that some node $v$ has many successors which are far a way who are trying to send link to his grandfather at node $v$. In order that the sum of links will be not too large we need a cousin trick\footnote{Recall that we send a link to the closest either grandfather or cousin}.

Now we let us write pseudo-code for Alice's communication on the $i$’th block. Alice will maintain her PrefixList tree $PL_A$ and for some nodes of the tree she will associate edges from her set of edges $X$.

input: $w_A \in D(\Sigma_{out})^i$

1. Calculate $Decode(w_A) = List_i(w_A, C, \varepsilon) = \{B_1, B_2, \ldots, B_k\}$

2. Update tree $PL_A$ from $B_1, \ldots B_k$.

3. For $j = 1 \ldots \min\{k, Ent\}$ do:
   
   (a) If $Size(A) > (i + 1) \log \Sigma_{in}$ then Send $C(A)[i+1]$ to Bob and continue to next round.
   
   (b) Set $v \in PL_A$ to be node corresponding to $B_j$.
   
   (c) Let $u \in PL_A$ be the last predecessor (i.e. closest to $v$) of $v$ associated with some edge $e_{\text{grandparent}} \in \mathcal{T}$.
   
   (d) If $E(B_j)$ contains exactly one son of $e_{\text{grandparent}}$ (let us call it $e_{\text{father}}$ ) then:
      
      i. Let $e_{\text{son}} \in X$ to be son of $e_{\text{father}}$.
      
      ii. Add $(A, e_{\text{son}})$.
      
      iii. Associate $e_{\text{son}}$ with $v$.

4. End − round($A, i + 1$)

5. Send $C(A)[i + 1]$ to Bob and continue to next round.

Also as with case of large alphabet we will run the protocol $n = \frac{T}{\varepsilon} = O(\frac{n}{\varepsilon})$ rounds. Output will be the same

1. Construct $PrefixList(w_A, C, \varepsilon)$.

2. For every output node $v \in PrefixList(w_A, C, \varepsilon)$

3. If exists successor of $v$, $v_s \in PrefixList(w_A, C, \varepsilon)$ such that $agr(v \to v_s, w) > T$ then output $v$. 

\footnote{Recall that we send a link to the closest either grandfather or cousin}
9 Analysis

The goal of this section is to prove

**Theorem 40.** The protocol from Section 8 is resilient $(1-\varepsilon)$-symmetric noise and it solves the task of list decoding Pointer Jumping Problem of depth $T$ with list size $O(\frac{1}{\varepsilon})$. The communication complexity of this protocol is $O(\varepsilon(T))$.

The main technical difficulty for constant alphabet comes from the fact that we need to prove that sum of links is not too large. We prove it in Claim 46. All the rest analysis in this section is similar to the one we had for large alphabet with small adaptations. All the bound on the list size is exactly the same. If some node $v$ from $PL_A$ or $PL_B$ has an edge associated to it we will say that we speak at node $v$. We will have here several differences. First we say that we advance at block $i$ if Alice or Bob spoke at node associated with a correct transcript. Let $P_A \subset PL_A$ and $P_B \subset PL_B$ be the paths from the root which corresponds to transcripts of Bob and Alice, respectively.

**Remark 41.** Note that we have here three trees $T, PL_A, PL_B$ as well three ”correct” paths $P \subset T, P_A \subset PL_A, P_B \subset PL_B$.

Then we are saying that we advance at block $i$ if $P_A[i]$ or $P_B[i]$ is associated with an edge in other words its mean that at round $i$ either Alice or Bob got the correct transcript in his list and send an edge as a respond to this transcript. Let $X_i$ be a set of edges associated with $P_A[1, \ldots i]$ and $Y_i$ be the set of edges associated with $P_B[1, \ldots i]$. As we will see $X_i \cup Y_i$ is a rooted sub-path of $P$ thus if we advanced $T$ times then we will reach the end of $P$ thus if we will advance $T$ times the correct codeword will be in the list. The following lemma is essentially says that if we will look on the ”correct paths” of Alice and Bob then we will see that at this path the correct communication.

**Lemma 42.** The set $X_i \cup Y_i$ is a rooted sub-path of $P$.

**Proof.** Informally the claim follows since Alice can send a new edge at node $v$ only after she will receive a respond from the last edge she sent. Now let us try to write it more formally. Let us prove this by induction. At round 0 the statement trivially follows. If at round $i$ nobody spoke at nodes $P_A[i], P_B[i]$ then $X_i = X_{i-1}, Y_i = Y_{i-1}$ and again it trivially follows. If at block $i$ Alice spoke at $P_A[i]$ (or the same proof for Bob $P_B[i]$), let $j < i$ be last time that Alice spoke at $P_A[j]$ let $e_j \in T$ be the corresponding edge associated with this node. Then according to our protocol $e_j$ will be set to be $e_{\text{grandparent}}$. Note that by definition of $P_A[i]$ it corresponds to the correct transcript. Thus $E(P_A[i]) \subset Y$ therefore $e_{\text{father}} \in Y$. By induction we assume that $e_j$ is the last node of $X_j \cup Y_j$ and it is on the path $P$. Therefore $e_{\text{father}}$ next node on $P$ and $e_{\text{son}}$ is the node after it.

Note that if the following three conditions are satisfied, we advance at block $i$.

1. At block $i$ both Alice and Bob decoded correctly each other’s messages i.e., $B[1, \ldots i-1] \in \text{List}_{i-1}(w_A, C, \varepsilon)$ and $A[1, \ldots i-1] \in \text{List}_{i-1}(w_B, C, \varepsilon)$. 25
2. The list size at block $i$ was less than $\text{Ent}$.

3. We did not send long link at block $i$ and we did not send long link at previous blocks which we have not finished sending by block $i$.

From Lemma 35 it follows that there are at least $\varepsilon n - O(\varepsilon' n)$ blocks that satisfy the first condition. As in case with large alphabet we know that there are at most $O(\varepsilon' n)$ blocks which do not satisfy second condition. All that is left for us is to bound the number of blocks that do not satisfy the third condition.

The following lemma takes care of the third issue.

**Lemma 43.** The number of blocks at which long links where being sent is at most $O(\varepsilon' n)$.

We will dedicate the rest of the section to proving this lemma.

Every edge that Alice has sent is associated with some node from $PL_A$. For every node $v \in PL_A$ for which we associated an edge we let $\text{link}(v)$ be a offset in transcript of this edge.

**Definition 44.** Let us assume that the $i$-th edge $e_i$ in the transcript was sent at node $v$ and that it was sent at round $j$. Suppose $e_i$ in the transcript has a link to $e_{i-t}$. Let us assume that $e_{i-t}$ was sent at round $k$; then we define length of the link by $j - k$. We denote it by $|\text{link}(v)|$.

Note that the length of the link is not much different than the link itself.

**Claim 45.** $\text{link}(v) \leq \text{Ent} \cdot (|\text{link}(v)| + 1)$.

We denote by $PL_A(v)$ to be a subtree of $PL_A$ rooted at $v$.

**Claim 46.** Let $v \in PL_A$ be a node at which we have sent an edge then
\[
\sum_{u \in PL_A(v) \setminus \{v\}} |\text{link}(u)| \leq |PL_A(v)|.
\]

**Proof.** We are going to prove the lemma by induction. For leaves it is trivially true. Denote by
\[
S(v) = \sum_{u \in PL_A(v) \setminus \{v\}} |\text{link}(u)|.
\]

Next assume that at node $v$ we have sent an edge $e \in M$. Let us consider all nodes in $PL_A(v)$: $u_1, u_2, \ldots, u_k \in PL_A(v)$ at which we have sent grandchildren of $e$. First note that according to our protocol on the nodes between $v$ and $u_i$ we did not sent any edges. Therefore using this and induction assumption we get
\[
S(v) = \sum_{i=1}^{k} |\text{link}(u_i)| + \sum_{i=1}^{k} S(u_i) = \sum_{i=1}^{k} |\text{link}(u_i)| + \sum_{i=1}^{k} |PL_A(u_i)|.
\]

Let us also assume $u_i$ are sorted according the the order they appear in the transcript. Then $u_1$ will send a link to $v$ (or maybe somebody later, and set that this is his grandparent), $u_2$
will send a link to $u_1, \ldots u_k$ will send the link to $u_{k-1}$ (or maybe somebody later) and all these nodes have links to their cousins. Thus

$$\sum_{i=1}^{k} |\text{link}(u_i)| \leq \text{dist}_{PL_A}(v, u_k).$$

Therefore

$$S(v) \leq \sum_{i=1}^{k} |PL_A(u_i)| + \text{dist}_{PL_A}(v, u_k).$$

Note that since none of $u_i$’s are predecessor of another $u_j$ $PL_A(u_i)$ are disjoint subtrees of $PL_A(v)$. Also note that path from $v$ to any $u_i$ does not belong to any of these trees thus we get that

$$\sum_{i=1}^{k} |PL_A(u_i)| + \text{dist}_{PL_A}(v, u_k) \leq |PL_A(v)|.$$ 

And the claim follows.

Now we are ready to prove lemma Lemma 43.

**Proof of Lemma 43.** Recall that we assume that every link $l$ takes $2 \log l + 2$ space of storage. The number of blocks at which long link of size $l > \text{MLSize}$ is sent is at most $\frac{\log l + 2}{\log \text{MLSize}} + 2 < \frac{l}{\text{MLSize}} + 3$. From Claims 45, 46 it follows that:

$$\sum_{u \in PL_A} \text{link}(u) \leq \text{Ent} \cdot |PL_A| \leq \frac{L^2}{\varepsilon'} n.$$ 

And the same for $PL_B$. Therefore the number of long links is bounded by:

$$LL = \frac{2L^2}{\varepsilon' \text{MLSize}} n = O(\varepsilon'n).$$

The number of blocks at which long links where sent is bounded by

$$\sum_{u \in PL_A} \frac{\text{link}(u)}{\text{MLSize}} + \sum_{u \in PL_B,} \frac{\text{link}(u)}{\text{MLSize}} + 3LL \leq O\left(\frac{L^2}{\varepsilon' \text{MLSize}} n\right) + O(\varepsilon'n) = O(\varepsilon'n)$$

We will use the following theorem for unique decoding:

**Theorem 47.** If some integer $i_0 \leq n$ it holds that

$$\sum_{i=1}^{i_0} \min(\text{agr}(w_B[i], c_A[i]), \text{agr}(w_A[i], c_B[i])) \geq T + O(\varepsilon'n) = O(\varepsilon n),$$

Then there exists output node on both paths $P_A, P_B$ and this node is located before level $i_0$.

\footnote{Recall that this are the “correct” paths}
Proof. To prove this theorem it is enough to show that there are at least $T$ rounds before time $i_0$ at which we advance. This follows from Lemma 35, Lemma 43 and Claim 34.

We are going to use the following corollary of this theorem:

**Corollary 48.** If output communication point on $P_A$ or $P_B$ is located after $s$ then

$$d(w_B[1, \ldots, s], c_A[1, \ldots, s]) + d(w_A[1, \ldots, s], c_B[1, \ldots, s]) \geq s - O(\varepsilon n).$$

**Claim 49.** $v(X \cup Y)$ is in the list.

**Proof.** The proof here is almost the same as in Claim 38. First find $i_0$ such that we have agreement $\varepsilon n - T - \varepsilon' n$ before $i_0$ and agreement $T + \varepsilon' n$ after $i_0$. From Theorem 47 it holds that by taking $c_0$ small enough we will reach output communication node before $i_0$. Next proceed the proof exactly as in Claim 38.

**Proof of Theorem 3.** We are already know that the protocol from Section 8 solves correctly the task of list decoding pointer jumping problem. Since pointer jumping problem is complete for communication protocols we In order to prove the theorem we will execute the protocol above. Note that number of rounds is $O_\varepsilon(T)$. Also note that $\Sigma_{in}$ is of constant size (depending on $\varepsilon$, but not on $T$), thus $\Sigma_{out}$ by Theorem 22 is also of constant size.

## 10 Optimality of our results for list decoding

In this section we are going to show that for any $\alpha, \beta$ with $\alpha + \beta \geq 1$ there exists protocol such that any list decoding protocol resilient to $(\alpha, \beta)$ noise with linear communication complexity must have exponential list size.

We will show this by showing that if one can perform some task with $(\alpha, \beta)$ noise than one can perform the same noiseless task in 3 rounds.

**Theorem 50.** Let $\alpha, \beta \in [0, 1]$ such that $\alpha + \beta \geq 1$. Let us assume that there exists a protocol performing task $T$ which is resilient to $(\alpha, \beta)$ noise. Then there exists a 3-round noiseless protocol, (here at each round one can send arbitrary length messages) with the same communication complexity performing task $T$.

**Proof.** If Alice and Bob speaks alternatively then this theorem follows very easily since adversary can erase all the communication of Alice first $\alpha$ fraction of the communication and all communication of Bob $1 - \alpha < \beta$ fraction of communication at the end. This way we will in fact get two round protocol where at the beginning only Bob is speaking(since all Alice’s communication is erased) and at the end only Alice is speaking. But if Alice and Bob does not speaking alternatively then it may happen that at the beginning Alice speaks more and at the end Bob speaks more so this way at first $\alpha$ fraction of communication Alice speaks more than $\alpha$ fraction of her communication. Thus we need to choose more carefully where to erase Alice’s communication and where to erase Bob’s communication.
Let us assume that during the protocol Alice sent $n_A$ messages and Bob sent $n_B$ messages. Let us also define $A(i), B(i)$ be number of messages sent by Alice and by Bob before round $i$. Consider $f(i) = \frac{A(i)}{n_A} - \frac{B(i)}{n_B}$, let $i_{\text{max}}$ be a point at which this function is maximal. If $n_A - A(i_{\text{max}}) \leq \alpha n_A$ and $B(i_{\text{max}}) \leq \beta n_B$ then Eve can destroy communication of Bob before $i_{\text{max}}$ and communication of Alice after $i_{\text{max}}$ and thus we will get protocol equivalent to a two round protocol where Alice send all her information in rounds $1, \ldots, i_{\text{max}}$ and Bob sends all his information in rounds $i_{\text{max}} + 1, \ldots, n$.

Now assume that $B(i_{\text{max}}) > \beta n_B$. Let $i_0$ be a point such that $B(i_0) = B(i_{\text{max}}) - \beta n_B$. In this case Eve will corrupt the communication of Alice in the interval $[0, \ldots, i_0]$, the communication of Bob in the interval $[i_0, \ldots, i_{\text{max}}]$, and the communication of Alice in the interval $[i_{\text{max}}, n_A + n_B]$. This way we will obtain protocol which is equivalent to a 3 round protocol. By definition of $i_0$ in the interval $[i_0, i_{\text{max}}]$ there are exactly $\beta$ fraction of Bob's communication. By definition of $i_{\text{max}}$ in the interval $[i_{\text{max}}, i_1]$ and the communication of Bob everywhere else. In this case from definition of $i_{\text{max}}$ we will get

$$\alpha = \frac{A(i_1) - A(i_{\text{max}})}{n_A} \leq \frac{B(i_1) - B(i_{\text{max}})}{n_B}$$

Thus communication of Bob in the interval $[i_{\text{max}}, i_1]$ is at least $\alpha$ therefore his communication on other two intervals will be at most $1 - \alpha \leq \beta$.

Now we are ready to prove Theorem 2.

**Theorem 51 (Theorem 2).** For every $\alpha, \beta$ such that $\alpha + \beta \geq 1$. Let $\pi$ be a protocol which is resilient to $(\alpha, \beta)$ adversarial noise and which solves list decoding problem of Pointer Jumping Problem of depth $T$ with list of size $L = \exp(o(T))$. Then $\text{CC}(\pi) = \exp(\Omega(T))$.

**Proof.** If protocol $\pi$ is resilient to $(\alpha, \beta)$ noise with $\alpha + \beta \geq 1$ then by Theorem 50 we know that there exists a three round protocol which solves list decoding of Pointer Jumping Problem. Consider a protocol which at the end outputs a random codeword from the list. Then this protocol solves the Pointer Jumping Problem with probability at least $\frac{1}{L}$. From Corollary 9 it follows that this protocol must have $\exp(\Omega(T))$ communication complexity. \qed
Part II
Unique Decoding

In this part we are going to consider the following question: assume that Alice and Bob want to perform an interactive communication protocol $\pi$ and assume that Eve can adversary corrupt up-to $\alpha$ fraction of messages sent by Alice and $\beta$ fraction of coordinates sent by Bob. We want to find a region at which we still can perform interactive communication in the unique decoding regime. The answer to this question is not obvious and quite surprising.

11 Unique decoding up-to $\alpha + 2\beta < 1$.

In this section we are going to show how to perform unique decoding up-to $\alpha + 2\beta < 1$ and $2\alpha + \beta < 1$. Although, as we will see from the next sections, this is not optimal. The algorithm described here gives essential ideas for the next sections.

Let us assume that $\beta < \frac{1}{2}(1 - \alpha)$ we are going to show that in this case Alice will output the correct answer. Let $\varepsilon = 1 - 2\beta - \alpha$. Note that $\varepsilon > 0$. Let $\varepsilon' = c\varepsilon$ for some small constant $c$ to be defined later. The protocol is very simple we will perform list decoding protocol from previous section which is resilient to $(1 - \varepsilon')$ noise, but instead of outputting a list at the end we will output the closest answer i.e., we will find $c_{\text{output}} \in C$ such that

$$d(c_{\text{output}}, w) = \min\{d(c, w) : c \in C\}.$$  

Here $C$ is the set of all codewords. We will calculate $E(c_{\text{output}})$ and output $v(X \cup E(c_{\text{output}}))$.

Now let us show why this will be the correct answer. First we will need the following lemma about $(\varepsilon', \frac{1}{2} + 1)$ list decodable tree codes:

**Lemma 52.** Let $C$ be $(\varepsilon', \frac{1}{2} + 1)$ list decodable tree code. For every $x, y \in \Sigma_n$ let $s$ be first index where $x[s] \neq y[s]$. Then $d(C(x), C(y)) \geq n - s - 2\varepsilon'n$.

**Proof.** Let us take $w$ to be $C(x)$ on the first $n - \varepsilon'$ locations and to be $C(y)$ at the last $\varepsilon'n$ locations. Then $\delta_s(w, C(y)) \leq 1 - \varepsilon'$ and also $\delta_s(w[1, \ldots, n - \varepsilon'n], C(x)[1, \ldots, n - \varepsilon'n]) = 0 \leq 1 - \varepsilon'$. By definition of list tree code we know that agreement between $w$ and PrefixList is at most $\varepsilon'Ln = (1 + \varepsilon')n$. Note that agreement between $w$ and $C(x)$ on the first $n - \varepsilon'n$ locations is $n - \varepsilon'n$. Note also that starting level $s$, $C(x)$ and $C(y)$ represent different branches in the PrefixList. Agreement between $w$ and $C(y)$ starting level $s$ is at least the agreement between $C(x)[i, \ldots, n]$ and $C(y)[s, \ldots, n]$. Thus

$$n - \varepsilon'n + agr(C(y)[s, \ldots, n], C(x)[s, \ldots, n]) \leq \varepsilon'Ln = n + \varepsilon'n$$

Thus $agr(C(y)[s, \ldots, n], C(x)[s, \ldots, n]) \leq 2\varepsilon'n$.

Let us assume by contradiction that protocol outputs wrong answer. Let us assume that Alice outputs the wrong answer. Let $c_B \in \Sigma_{\text{out}}$ be a codeword which was sent by Bob. Let
us assume that \( w_A \) was the codeword received by Alice. Assume assume that \( c_{\text{output}} \) is the closest codeword to \( w_A \). There are two important points on \( c_B \) one is a “split” point \( s \) to be a first place where \( c_{\text{output}}[s] \neq c_B[s] \) and an other is \( e_{\text{end}} \) output communication point (we will show soon that exists one on the "correct" path \( c_B \)). Observe that if output node is located before split point \( s \) then we output the correct answer. Thus by contradiction assumption \( s < e_{\text{end}} \). The proof now follows from Lemma 52 which will give lower bounds on \( s \) and from Corollary 48 which will give upper bounds on \( e_{\text{end}} \) and thus also on \( s \).

Let us define \( B_1 = d(c_A[1, \ldots, s], w_B[1, \ldots, s]) \) and \( B_2 = d(c_A[s + 1, \ldots, n], w_B[s + 1, \ldots, n]) \) note that \( B_1 + B_2 \leq \beta n \). From the Lemma 52 we know that \( d(c_{\text{output}}, c_B) \geq n - s - O(\varepsilon'n) \); thus, since \( w_A \) is closer to \( c_{\text{output}} \) than to \( c_B \) we have that the number of errors in Bob’s messages in last \( n - s \) rounds was at least \( \frac{n - s - O(\varepsilon'n)}{2} \). Thus we have that

\[
B_2 \geq \frac{n - s - O(\varepsilon'n)}{2}
\]

Rewriting thus we get

\[
s \geq n - 2B_2 - O(\varepsilon'n)
\]

On other hand from Corollary 48 it follows that

\[
B_1 + \alpha n \geq d(w_B[1, \ldots, s], c_A[1, \ldots, s]) + d(w_A[1, \ldots, s], c_B[1, \ldots, s]) \geq s - O(\varepsilon'n) .
\]

Thus we got that

\[
B_1 + \alpha n \geq n - 2B_2 - O(\varepsilon'n)
\]

Rewrite this and we will get that

\[
2\beta n + \alpha n \geq B_1 + B_2 + \alpha n \geq n - O(\varepsilon'n) .
\]

Thus we have got that \( 2\beta + \alpha > 1 - O(\varepsilon') \) thus by taking \( c \) small enough we will get that \( 2\beta + \alpha > 1 - \varepsilon \) contradiction to the definition of \( \varepsilon \).

12 Repetition power of the protocol

In order to construct a protocol which can handle optimal fraction of errors we would like to give different to the cost of corruption of any symbol in the protocol. One way to it is by repetition power of a protocol. In the case of one way communication repetition power \( k \) just means that we will repeat each symbol \( k \) times. In case of interactive protocol with \( n \) rounds. Let \( a : [n] \rightarrow \mathbb{N}; \) we will do the same: we will just send the symbol of the \( i \)th round \( a(i) \) times. Unlike one-way communication here we need to explain how Alice (Bob) acts when at round \( i \) she receives \( a(i - 1) \) symbols. Since these are all these are soft symbols (i.e., elements from \( D(\Sigma) \)) Alice will just calculate the average of all received distributions. It will be more convenient for us to give a definition with \( a(i), b(i) \) where \( a(i) \) is the number of symbols which Alice send at her \( i \)’th round and \( b(i) \) the number of symbols Bob sent at his \( i \)-th round.
Let \( \pi \) be a protocol with \( 2n \) alternating rounds of Alice and Bob over alphabet \( \Sigma_{\text{out}} \). Let as assume that Alice talks at rounds \( 2i - 1 \) and Bob talks at rounds \( 2i \) for \( i \in [n] \). At every round \( i \) of the protocol \( \pi \) Alice or Bob receives symbol from \( D(\Sigma_{\text{out}}) \) and outputs symbol in \( \Sigma_{\text{out}} \).

**Definition 53.** Let \( a : [n] \to \mathbb{N}, \ b : [n] \to \mathbb{N} \) be any functions. The repetition power \( \pi^{a,b} \) of the protocol \( \pi \). Is the following protocol: In the protocol \( \pi^{a,b} \) Alice will receive at round \( 2i - 1 \) \( b(i - 1) \) symbols \( w_1, \ldots, w_{b(i-1)} \) \( \in D(\Sigma_{\text{out}}) \) calculate average \( w = \frac{1}{b(i-1)} \sum_{i=1}^{b(i-1)} w_i \). Run protocol \( \pi \) with input symbol \( w \). Get output \( c_A[i] \) and send \( a(i) \) copies of \( c_A[i] \) to Bob. Bob at round \( 2i \) will receive \( a(i) \) symbols from \( D(\Sigma_{\text{out}}) \) run \( \pi \) on their average and sent \( b(i) \) copies of output of the protocol \( \pi \).

First note the simple connection between communication costs of \( \pi \) and \( \pi^{a,b} \)

**Lemma 54.** Let \( \pi \) be a protocol with \( 2n \) alternating rounds of Alice and Bob over alphabet \( \Sigma_{\text{out}} \). Then communication cost of \( \pi^{a,b} \) is \( \sum_{i=1}^{n} a(i) + b(i) \).

**Proof.** Follows from the definition. \( \square \)

The following lemma makes a connection between amount of noise in protocol \( \pi \) and protocol \( \pi^{a,b} \).

**Lemma 55.** Let us assume that \( c'_A, c'_B \) what Alice and Bob sent in the protocol \( \pi^{a,b} \). Let \( w'_A,w'_B \) is what they have received in protocol \( \pi^{a,b} \). Let \( c_A, c_B, w_A, w_B \) be a corresponding values on which the protocol \( \pi \) was simulated the the following holds

\[
d(c'_A, w'_B) = \sum_{i=1}^{n} a(i) d(c_A[i], w_B[i]) ,
\]

And the same for Bob i.e.,

\[
d(c'_B, w'_A) = \sum_{i=1}^{n} b(i) d(c_B[i], w_A[i]) .
\]

This lemma essentially shows that using repetition we can give different weight to the symbols of our protocol.

**Proof.** Let us consider \( c'_A, w'_B \) as matrix vectors with \( c'_A[i,j] \) for \( 1 \leq i \leq n, 1 \leq j \leq a(i) \). Then \( c'_A[i,j] = c_A[i] \) for every \( j \). It also holds that \( w_B[i] = \frac{1}{a(i)} \sum_{j=1}^{a(i)} w'_B[i,j] \) (note here we are summing up probability distributions). Next note that for every \( \sigma \in \Sigma \) and \( k \) distributions \( \sigma_1, \ldots, \sigma_k \) it holds that \( d(\sigma, \frac{1}{k} \sum \sigma_i) = \frac{1}{k} \sum d(\sigma, \sigma_i) \). Therefore for every \( i \) it holds that

\[
\sum_{j=1}^{a(i)} d(c'_A[i,j], w'_B[i,j]) = \sum_{j=1}^{a(i)} d(c_A[i], w'_B[i,j]) = a(i) d(c_A[i], \frac{1}{a(i)} \sum_{j=1}^{a(i)} w'_B[i,j]) = a(i) d(c_A[i], w_B[i]) .
\]

By summing over all \( i \) we will get the first equation of the theorem. The proof of the second equation is the same when Alice is replaced by Bob. \( \square \)
13 Decoding of one sided protocols

In this section we will consider an easier task, we will require that only Alice will output the correct answer. Recall the analysis of the algorithm from the Section 11. Note that essentially it means that in order for Eve to corrupt output of Alice she needs to pick some $i \in [1,n]$ and corrupt at least $\alpha_1, \beta_1$ fraction of communication of Alice and Bob before $i$, such that $\alpha_1 + \beta_1 > 1 - O(\varepsilon)$ and at $n - i$ last rounds Eve needs to corrupt $1/2 - O(\varepsilon)$ fraction of the communication sent by Bob. In the scenario when Alice and Bob speak at the same speed it always optimal for Eve to corrupt Alice’s communication before $i$ and half of the communication of Bob after $i$. Now let us change protocol such that at the beginning Alice speaks more and at the end Bob speaks more. In this case Eve will need spent more error to corrupt Alice’s communication at the beginning and more error to corrupt Bob’s communication at the end. Essentially we are going to carefully choose functions $a,b$ and our protocol in this section will be $\pi^{a,b}$ where $\pi$ is the list decodable protocol from earlier in the paper. Let us consider the following example: let $\alpha = \frac{1}{3}$. Let us now divide our communication into three parts where in each part we will have $\frac{n}{3}$ blocks of communication. Let us consider the following functions $a(i) = 1$ for every $i \in [n], b(i) = 1$ for $i = 1, \ldots, \frac{n}{3}, b(i) = 2$ for $i = \frac{n}{3}, \ldots, \frac{2n}{3}$ and $b(i) = 4$ for $i = \frac{2n}{3}, \ldots, n$. Then in the first part of the communication block of communication will be one Bob’s and one Alice’s message. In the second part in each block Bob will send twice his (the same) message and Alice will send one message; in the last part in each block Bob will send his message four times and Alice will send one message. So in total Alice will send $n$ messages and Bob will send $\frac{7}{3}n$ messages.

In this scenario suppose we want to pick some $i$ and corrupt almost all (up to $O(\varepsilon n)$) blocks before $i$ and half of the messages of Bob in rounds after $i$. If we want to corrupt the minimum amount of communication of Bob, we will corrupt his messages in the first $i - \frac{n}{3}$ rounds then Alice’s messages in next $\frac{n}{3}$ rounds and half of Bob’s messages in the last $n - i$ rounds. It is easy to see that in this case we will corrupt $n$ Bob’s messages. Thus we can decode up-to $\beta < \frac{3}{7}$.

Now let us analyze such protocols for different functions $a(i)$. For every integrable function $f : [0,1] \to \mathbb{R}^+$ with $\int_0^1 f(t)dt = 1$. The function $f(t)$ will denote the number of messages Bob sends at round $tn$ per every Alice’s message. Let us now define for every $f$:

$$L_1(\alpha, f) = \inf \{ \int_0^s f(t)g(t)dt + \frac{1}{2} \int_s^1 f(t)dt : s \in [\alpha, 1], 0 \leq g(t) \leq 1, \int_0^s g(t)dt \geq s - \alpha \}$$

**Remark 56.** We want to mention that once we have picked $s$, the functions $g$ which minimizes an expression above is 1 on set $S, \mu(S) = s - \alpha$. Here $S$ is the set of form $\{x < s : f(x) < s\} \cup A_t$, where $A_t \subset \{x : f(x) = t\}$ for some $t$.

That is, $s$ in the expression above is exactly where correct and non correct codewords are split and $g(t)$ corresponds to the fraction of communication of Bob corrupted at round $tn$ for $t \leq s$. 

33
Now let us define non explicitly the function which will be the boundary till which we can perform one sided unique decoding. In the next subsection we will give an explicit formula to this function. Let us set

$$L_1(\alpha) = \sup_f (L_1(\alpha, f) : \int_0^1 f(t)dt = 1, f(t) \geq 0)$$

Thus $L_1(\alpha)$ is the maximal amount of error Bob can handle when Alice has $\alpha$ error and we want Alice to output the correct answer.

The following theorem says that we can to decode with one sided answer if $\beta < L_1(\alpha)$ and that if $\beta > L_1(\alpha)$ we cannot decode. We do not know what happens on the line $\beta = L_1(\alpha)$.

**Theorem 57.** For every $(\alpha, \beta)$ such that $\beta < L_1(\alpha)$. There exists a protocol which is resilient to $(\alpha, \beta)$ noise and solves the one way pointer jumping problem. The protocol uses $O_{\alpha, \beta}(T)$ communication.

**Proof.** Let us set $f$ be a function such that $L_1(\alpha) = L_1(\alpha, f)$ (the next section shows such an $f$ actually exists). Let $\varepsilon = O(L_1(\alpha) - \beta)$ be a small constant.

For any given function $f(t)$ and for any integer $n$ let us define a function

$$b(i) \triangleq \left\lfloor \varepsilon \int_{i/n}^{(i+1)/n} f(t)dt \right\rfloor.$$

Let $a(i) = \frac{1}{i}$ for every $i$. Let $\pi$ be a list decoding protocol from previous section with code $C$ which is $(\varepsilon, \frac{1}{\varepsilon} + 1)$ list decodable and with $n$ rounds. Our protocol will be $\pi^{a,b}$. Let us define $c'_A, c'_B, w'_A, w'_B$ to be messages sent by Alice and Bob and messages received by them. Let $c_A, c_B, w_A, w_B$ be a corresponding messages on which list decoding protocol $\pi$ was simulated.

We will see in the next section that supremum of $L_1$ is achieved at the functions with $\sup f(x) < \infty, \inf f(x) > 0$. Let us assume that $\varepsilon < \inf f(x)$. Let $R = \sup f(x)$. Then every round Bob send at least one and at most (constant) $R$ messages. Note also that from Lemma 54 it follows that in protocol $\pi^{a,b}$ Alice sends $\frac{n}{\varepsilon}$ messages and Bob sends $\frac{n}{\varepsilon} - t$ where $t < n$ messages. For the sake of simplicity let us assume that Bob sends $\frac{n}{\varepsilon}$ messages. (say by sending dummy messages at the end.)

Let $c_{out\text{put}}$ be the closest codeword to $w_A$, where here we give to symbol $i$ weight $b(i)$ i.e., $c_{out\text{put}} \in C$ we choose minimizes the expression

$$\sum_{i=1}^n b(i)d(c[i], w_A[i]).$$

Let us assume that first place where $c_B$ and $c_{out\text{put}}$ are different is $s \cdot n$. Let us assume by contradiction that Alice outputs a wrong answer then output node is located after $s \cdot n$.

Then by Lemma 52 we know that $d(c_B[s \cdot n, \ldots , n], c_{out\text{put}}[s \cdot n, \ldots , n]) \geq (n - s \cdot n) - O(\varepsilon n)$. Therefore

$$\sum_{i=s \cdot n}^n b(i)d(c_{out\text{put}}[i], c_B[i]) \geq \sum_{i=s \cdot n}^n b(i) - R \cdot O(\varepsilon n) = \sum_{i=s \cdot n}^n b(i) - O(\varepsilon n)$$

34
Therefore
\[ \sum_{i=s-n}^n b(i) d(w_A[i], c_B[i]) \geq \frac{1}{2} \sum_{i=s-n}^n b(i) - O(\varepsilon n) \geq \frac{n}{1} \frac{1}{2} \int_s^1 f(t) dt - O(\varepsilon n). \]

On other hand from Corollary 48 it follows that
\[ d(w_B[1, \ldots, s \cdot n], c_A[1, \ldots, s \cdot n]) + d(w_A[1, \ldots, s \cdot n], c_B[1, \ldots, s \cdot n]) \geq s \cdot n - O(\varepsilon' n). \tag{3} \]

Note that since \( a(i) = \frac{1}{\varepsilon} \) is constant from Lemma 55 it follows that
\[ d(w_B[1, \ldots, s \cdot n], c_A[1, \ldots, s \cdot n]) \leq \varepsilon d(w'_B[1, \ldots, \frac{s \cdot n}{\varepsilon}], c'_A[1, \ldots, \frac{s \cdot n}{\varepsilon}]). \tag{4} \]

We are assuming that at most \( \alpha \) fraction of Alice’s communication was corrupted thus
\[ d(w'_B[1, \ldots, \frac{s \cdot n}{\varepsilon}], c'_A[1, \ldots, \frac{s \cdot n}{\varepsilon}]) \leq \frac{s \cdot n}{\varepsilon}. \tag{5} \]

Therefore from Equations (3), (4), (5) it follows that,
\[ d(w'_A[1, \ldots, s \cdot n], c'_B[1, \ldots, s \cdot n]) \geq s \cdot n - \alpha n - O(\varepsilon n). \tag{6} \]

Let \( s' \cdot n = \sum_{i=1}^{s \cdot n} b(i) \) then from Lemma 55 it follows that
\[ d(w'_A[1, \ldots, s' \cdot n], c'_B[1, \ldots, s' \cdot n]) = \sum_{i=1}^{s \cdot n} b(i)d(w_A[i], c_B[i]). \]

Let us define \( g(t) = d(w_A[[tn]], c_B[[tn]]) \). Then we have
\[ d(w'_A[1, \ldots, s' \cdot n], c'_B[1, \ldots, s' \cdot n]) \geq \frac{n}{\varepsilon} \left( \int_0^s g(t) f(t) dt - O(\varepsilon) \right). \]

Thus we have that
\[ d(w'_A, c'_B) = \sum_{i=1}^n b(i)d(w_A[i], c_B[i]) \geq \frac{n}{\varepsilon} \left( \int_0^s g(t) f(t) dt + \frac{1}{2} \int_s^1 f(t) dt - O(\varepsilon) \right). \]

Note that from Equation (6) it follows that
\[ \int_0^s g(t) \geq s - \alpha - O(\varepsilon). \]

Therefore since \( f \) is bounded it follows that:
\[ d(w'_A, c'_B) \geq L_1(\alpha, f) - O(\varepsilon). \]

By taking \( \varepsilon \) small enough we will get a contradiction.
We also claim that the opposite direction is true:

**Theorem 5.8.** For every \( \alpha, \beta \) with \( \beta > L_1(\alpha) \), if a protocol \( \pi \) is resilient to \((\alpha, \beta)\)-noise, then there exists a constant \( O(\frac{1}{\beta - L_1(\alpha)}) \) rounds protocol with the same communication complexity in which Alice solves the same task.

**Proof.** Let us consider the protocol \( \pi \). Assume that during the protocol Alice sent \( n_A \) symbols and Bob sent \( n_B \) symbols. Let \( \varepsilon > 0 \) be a small constant to be chosen later. Let \( T(1) \) be the time when Alice sent her \( \varepsilon n_A \)'s symbol, and \( T(i) \) be the time when Alice sent her \( i\varepsilon n_A \)'s symbol. Let \( n_1 = T(1) - \varepsilon n_A \) be number of messages Bob sent by time Alice send \( \varepsilon n_A \) of her messages. Let \( n_i = T(i) - in \varepsilon \) be number of messages Bob sent in the time Alice sent \( i\varepsilon n_A \) of her communication. Let us set \( f(x) \) to be equal \( \frac{n_i - n_{i-1}}{\varepsilon n_B} \) in the interval \( [(i-1)\varepsilon, i\varepsilon] \). Note that \( \int_0^1 f(t)\,dt = 1 \) in other words \( f(x) \) is an approximation of the rate at which Bob speaks. Thus by definition of \( L_1(\alpha) \) there exists \( s \in [0,1] \) and \( 0 \leq g(t) \leq 1 \) such that \( \int_0^s g(t) \geq s - \alpha \) such that \( \int_0^s f(t)g(t)\,dt + \frac{1}{2}\int_0^1 f(t)\,dt \leq L_1(\alpha) < \beta \).

Let \( t = \lfloor \frac{s - \alpha}{\varepsilon} \rfloor \). Note that from Remark 55 it follows that we can assume w.l.o.g. that \( g = 1_s \) where \( S \) is union of \( t \) of intervals of form \([ (i-1)\varepsilon, i\varepsilon] \) plus a part of such an interval.

Let us define

\[
S = \bigcup_{k=1}^t [(i_k - 1)\varepsilon, i_k\varepsilon] \cup J \subset \bigcup_{k=1}^{t+1} [(i_k - 1)\varepsilon, i_k\varepsilon],
\]

where \( J \subset ([i_{t+1} - 1)\varepsilon, i_{t+1}\varepsilon] \). The minimum value is achieved when we take the intervals with minimal values of \( f(x) \). Since \( \int_0^1 f(t)\,dt = 1 \) value of \( f \) on \([ (i_{t+1} - 1)\varepsilon, i_{t+1}\varepsilon] \) is at most \( \frac{1}{\alpha - \varepsilon} \). Let us define \( S' = \bigcup_{k=1}^{t+1} [(i_k - 1)\varepsilon, i_k\varepsilon] \). Then for small enough \( \varepsilon \) it holds that

\[
\int_0^s f(t)1_{S'}\,dt + \frac{1}{2}\int_s^1 f(t)\,dt \leq \int_0^s f(t)1_{S}\,dt + \frac{1}{2}\int_s^1 f(t)\,dt + O(\varepsilon) \leq L_1(\alpha) + O(\varepsilon) < \beta
\]

Let us now assume that Eve has erased communication of Bob on the intervals \([T(i_k - 1), \ldots, T(i_k)] \) (from the representation of the set \( S \) above), and the communication of Alice in all the rest intervals before \( s \). Note that since \( \mu(S') \geq s - \alpha \) we have erased at most \( \alpha \) fraction of the communication of Alice and at exactly \( \int_0^s f(t)1_{S'} \) fraction of communication of Bob. Note that after time \( s \) Bob communicates \( \int_s^1 f(t) \) communication thus after time \( s \) Eve can corrupt at least \( \frac{1}{2} \) of communication of Bob, since \( \beta - \int_0^s f(t)1_{S'}\,dt \geq \frac{1}{2}\int_s^1 f(t)\,dt \).

We claim that by time \( s \) Alice knows the answer. Therefore we are done since then we have \( \frac{2}{\varepsilon} \) round protocol in which Alice solves the task. Assume by contradiction that there exists two inputs to Bob which are consistent with what Alice has saw by time \( s \) i.e., there exists two values \( y_1, y_2 \) such that for every non corrupted round \( j \) it holds that:

\[
\pi_B(x, y_1)(w_B[1, \ldots, s \cdot n])[j] = \pi_B(x, y_2)(w_B[1, \ldots, s \cdot n])[j]
\]

Where \( w_B \) is the corrupted codeword sent by Alice. Since Eve can corrupt \( \frac{1}{2} \) of the communication after round \( s \), Eve will construct a soft codeword which with probability \( \frac{1}{2} \) takes value sent by \( \pi_B(x, y_1) \) and with probability \( \frac{1}{2} \) takes value sent by \( \pi_B(x, y_2) \). Therefore we got codeword \( w_A \) which Alice could receive for both Bob’s inputs \( y_1, y_2 \). Therefore, since we
assume that Alice output on $x,y_1$ is different from output on input $x,y_2$, on one of these inputs Alice outputs the wrong answer.

\section{Calculating the functions $L_1$}

In this section for every $\alpha$ we want to calculate $\sup_f L_1(\alpha, f)$. At almost no more cost we can calculate more general function which we will need in the next section. For $\alpha < \gamma \leq 1$ let us define

\[ L_1(\alpha, \gamma, C) = \{ \sup_f L_1(\alpha, f) : \int_0^{\gamma} f(t)dt \geq C, \int_0^1 f(t)dt = 1 \}. \]

Note that $L_1(\alpha) = L_1(\alpha, 1, 1)$.

\textbf{Theorem 59.} Let us assume that function $f$ satisfies the following properties: Let $F(s) = \int_0^s f(t)dt$.

1. $f \geq 0$, $F(1) = 1$.
2. $F(\gamma) = C$
3. Denote by $l = \lfloor \frac{\gamma}{\alpha} \rfloor$. The function $f$ is constant on the interval $[(l-1)\alpha, l\alpha]$.
4. The function $T(s) = \frac{1}{2}(1 - F(s)) + F(s - \alpha)$ is constant in the interval $[\alpha, \gamma]$ and $L_1(\alpha, f) = T(\alpha)$.

Then $L_1(\alpha, \beta, C) = L_1(\alpha, f)$.

\textbf{Proof.} Let us now assume by contradiction that we have a function $h$ with $L_1(\alpha, h) \geq L_1(\alpha, f)$ and $\int_0^\gamma h(t)dt \geq C$. Define $H(s) = \int_0^s h(t)dt$. Then for every $s \in [\alpha, \gamma]$ let us take $g = 1_{[0, s-\alpha]}$ in the definition of the $L_1$. Then we will get that

\[ \frac{1}{2}(1 - H(s)) + H(s - \alpha) \geq L_1(\alpha, f) \geq \frac{1}{2}(1 - F(s)) + F(s - \alpha) = T(\alpha). \]  

\textbf{Claim 60.} For every $i = 1, \ldots, l$ it holds that $H(i\alpha) - H((i-1)\alpha) \leq F(i\alpha) - F((i-1)\alpha)$ in particular $H(i\alpha) \leq F(i\alpha)$.

\textbf{Proof.} Let us prove this by induction on $i$. For $i = 1$ this follows from Equation (7) for $s = \alpha$.

For $i$ we will get it from Equation (7) for $s = i\alpha$ and induction assumption on $i-1$.

\[ \frac{1}{2}(1 - H(i\alpha)) + H((i-1)\alpha) \geq \frac{1}{2}(1 - F(i\alpha)) + F((i-1)\alpha) \]

Rewriting the above inequality we get:

\[ H(i\alpha) - H((i-1)\alpha) \leq F(i\alpha) - F((i-1)\alpha) - (F((i-1)\alpha) - H((i-1)\alpha)). \]

Since by induction we assume that $F((i-1)\alpha) - H((i-1)\alpha) \geq 0$ the claim follows.
On other hand by using Equation (7) at \( s = \gamma \) we get that from second property of the theorem it follows that:

\[
T(\alpha) = \frac{1}{2}(1 - F(\gamma)) + F(\gamma - \alpha) = \frac{1 - C}{2} + F(\gamma - \alpha)
\]

Using third property of the theorem we will get that \( F(\gamma - \alpha) = F((l - 1)\alpha) + \left\{ \frac{\gamma}{\alpha} \right\} (F(\alpha) - F((l - 1)\alpha)) \) thus

\[
T(\alpha) = \frac{1 - C}{2} + F((l - 1)\alpha) + \left\{ \frac{\gamma}{\alpha} \right\} (F(\alpha) - F((l - 1)\alpha)). \tag{8}
\]

Now let us take in the definition of \( L_1 \) value of \( s = \gamma \) and \( g(x) \) to be 1 on \([0, (l - 1)\alpha]\) and \( \left\{ \frac{\gamma}{\alpha} \right\} \) on \([l - 1, \alpha, \alpha] \). Note that \( \int_{0}^{\gamma} g = \alpha(l - 1) + \alpha \left\{ \frac{\gamma}{\alpha} \right\} = (l + \left\{ \frac{\gamma}{\alpha} \right\}) \alpha - \alpha = \gamma - \alpha. \) Using this \( g \) we will get an equation

\[
\frac{1}{2}(1 - H(\gamma)) + H((l - 1)\alpha) + \left\{ \frac{\gamma}{\alpha} \right\} (H(\alpha) - H((l - 1)\alpha)) \geq T(\alpha)
\]

Since \( H(\gamma) \geq C \) it follows that

\[
H((l - 1)\alpha) + \left\{ \frac{\gamma}{\alpha} \right\} (H(\alpha) - H((l - 1)\alpha)) \geq F((l - 1)\alpha) + \left\{ \frac{\gamma}{\alpha} \right\} (F(\alpha) - F((l - 1)\alpha)) .
\]

Therefore from Claim 60 and above equation it follows that an equality should hold. Therefore:

\[
L_1(\alpha, g) \leq \frac{1}{2}(1 - H(\gamma)) + H((l - 1)\alpha) + \left\{ \frac{\gamma}{\alpha} \right\} (H(\alpha) - H((l - 1)\alpha)) = \\
= \frac{1}{2}(1 - H(\gamma)) - \frac{1 - C}{2} + T(\alpha) \leq T(\alpha) = L_1(\alpha, f).
\]

Thus \( L_1(\alpha, f) = L_1(\alpha, h). \)

Now in order to calculate \( L_1(\alpha) \) all we need is to construct the functions that admits all four conditions of Theorem \([59]\) with \( C = \gamma = 1. \)

Let us define functions \( f_\alpha \) as follows: \( \tilde{f}_\alpha(x) = 2^i \) in the interval \((i\alpha, \min\{(i + 1)\alpha, 1\}]. \) Define \( N_\alpha = \int_{0}^{1} \tilde{f}_\alpha(t)dt. \) Define \( f_\alpha(x) = \frac{\tilde{f}_\alpha}{N_\alpha}. \) Note that \( f_\alpha \) has a property that \( f_\alpha(x - \alpha) = \frac{1}{2}f(x) \) for \( x \in [\alpha, 1]. \) Let us define \( F_\alpha(s) = \int_{0}^{s} f_\alpha(t)dt. \) Properties 1-3 follow immediately. Property 4 follows from the following lemma:

**Claim 61.** The function:

\[
T(s) = \frac{1}{2}(1 - F_\alpha(s)) + F_\alpha(s - \alpha)
\]

is constant.
Proof. Let us differentiate $T(s)$.

$$T'(s) = f_\alpha(s - \alpha) - \frac{1}{2} f_\alpha(s)$$

But this is zero since $f_\alpha(s - \alpha) = 2 f(s)$. \qed

Thus we have that for every $s$ it holds that $C_\alpha = L_1(\alpha, f_\alpha) = \frac{1}{2} (1 - F_\alpha(s)) + F_\alpha(s - \alpha)$ for every $s$. We can now easily calculate $C_\alpha$.

**Lemma 62.**

$$C_\alpha = L_1(\alpha, f_\alpha) = \frac{1}{2} \left( 1 - \frac{1}{(1 + \{\frac{1}{\alpha}\})2^{|\frac{1}{\alpha}| - 1}} \right).$$

**Proof.** We can pick $s = \alpha$. In this case $T(s) = \frac{1}{2} (1 - F_\alpha(\alpha)) = \frac{1}{2} \left( 1 - \frac{1}{(1 + \{\frac{1}{\alpha}\})2^{|\frac{1}{\alpha}| - 1}} \right).$ \qed

Therefore we have proved the following lemma.

**Lemma 63.** For $\alpha = \frac{1}{k}$ for integer $k$ we have $L_1(\alpha) = \frac{1}{2} (1 - \frac{1}{2^{k-1}})$. For all other $\alpha$ it is $\frac{1}{2} \left( 1 - \frac{1}{(1 + \{\frac{1}{\alpha}\})2^{|\frac{1}{\alpha}| - 1}} \right)$.

15 Two sided communication

In this section we would like to answer the question in what is the maximal amount of error we can handle in case we want that both Alice and Bob will return the correct answer. As in previous section we can define the function $f(t)$ of the rate at which Bob speaks with respect to Alice’s rate. Let us define

$$L_2(\alpha, f) = \inf \left\{ \int_0^s g(t)f(t)dt : 0 \leq g(t) \leq 1, \int_0^s g(t)dt \geq \frac{s - 2\alpha + 1}{2}, s \in [1 - 2\alpha, 1] \right\}.$$

Let us define

$$L_2(\alpha) = \sup_f \{ \min(L_1(\alpha, f), L_2(\alpha, f)) : \int_0^1 f(t)dt = 1, f(t) \geq 0 \}.$$

The following theorems have almost the same proof as Theorems 57 and 58.

**Theorem 64.** For every $(\alpha, \beta)$ such that $\beta < L_2(\alpha)$. There exists a protocol $\pi$ which is resilient to $(\alpha, \beta)$ noise and which solves pointer jumping problem. The protocol $\pi$ uses $O_{\alpha,\beta}(T)$ communication.
Proof. Let us set $f$ to be a function such that $\beta < L_1(\alpha, f)$ and $\beta < L_2(\alpha, f)$. Let $\varepsilon = O(L_2(\alpha) - \beta)$ be a small constant. Define the function

$$b(i) \triangleq \left\lfloor \frac{n}{\varepsilon} \int_{i/n}^{(i+1)/n} f(t)dt \right\rfloor .$$

Let $a(i) = \frac{1}{i}$ for every $i$. Let $\pi$ be the list decoding protocol from previous section with code $C$ which is $(\varepsilon, \varepsilon) + 1$ list decodable and with $n$ rounds. Our protocol will be $\pi^{a,b}$. Let us define $c'_A, c'_B, w'_A, w'_B$ to be messages sent by Alice and Bob and messages received by them. Let $c_A, c_B, w_A, w_B$ be a corresponding messages on which the list decoding protocol $\pi$ was simulated. Alice output the codeword $c \in C$ which minimizes an expression

$$\sum_{i=1}^{n} b(i)d(c[i], w_A[i]) .$$

Bob outputs $c \in C$ which minimizes an expression

$$\sum_{i=1}^{n} a(i)d(c[i], w_B[i]) .$$

Since $a(i)$ is constant this is the same as to just output closest codeword to $w_B$. The output of Alice is correct by Theorem 58. So now we need to prove that output of Bob is correct. Let $c_{\text{output}}$ be the output of the Bob. Let $s \cdot n$ be a first place where $c_{\text{output}}$ is different from $c_A$; then we know from Lemma 52 that

$$d(c_{\text{output}}[s, \ldots, n], w_B[s, \ldots, n]) \geq \frac{n - s \cdot n}{2} - O(\varepsilon n) .$$

Let $s' = \sum_{i=1}^{s \cdot n} b(i)$ then from Lemma 55 it follows that

$$L_2(\alpha, f) > \beta \geq d(w'_A[1, \ldots, s'], c'_B[1, \ldots, s']) = \sum_{i=1}^{s \cdot n} b(i)d(w_A[i], c_B[i]).$$

Let us define $g(t) = d(w_A[[tn]], c_B[[tn]])$. By taking $\varepsilon$ small enough we have

$$\int_{0}^{s} g(t)f(t)dt < L_2(\alpha, f) .$$

Thus by definition of $L_2$ we get that

$$\int_{0}^{s} g(t)dt < \frac{s - 2\alpha + 1}{2} .$$

Note that the following holds

$$n \int_{0}^{s} g(t)dt = d(w_A[1, \ldots, s \cdot n], c_B[1, \ldots, s \cdot n]) .$$
From Corollary \[48\] it follows that
\[
d(w_B[1, \ldots s \cdot n], c_A[1, \ldots, s \cdot n]) + d(w_A[1, \ldots, s \cdot n], c_B[1, \ldots, s \cdot n]) \geq s \cdot n - O(\varepsilon n) .
\]
Thus we have that for small enough \(\varepsilon\)
\[
d(w_B[1, \ldots s \cdot n], c_A[1, \ldots, s \cdot n]) > \frac{s \cdot n + 2\alpha n - n}{2} .
\]
Thus total error we have is
\[
d(w_B[1, \ldots n], c_A[1, \ldots, n]) > \frac{n - s \cdot n}{2} + \frac{s \cdot n + 2\alpha n - n}{2} > \alpha n.
\]
Contradiction to the assumption that there is at most \(\alpha\) fraction of the communication is corrupted \(\square\)

**Theorem 65.** For every \(\alpha, \beta\) with \(\beta > L_2(\alpha)\), if protocol \(\pi\) resilient to \((\alpha, \beta)\) noise. Then there exist constant \(c(\alpha, \beta)\) rounds protocol with the same communication complexity in which Alice or Bob solves the task.

**Proof.** Let \(\alpha' < \alpha\) be a number such that \(\beta > L_2(\alpha')\). Let us take \(\varepsilon = O(\min\{\alpha - \alpha', \beta - L_2(\alpha)\})\).

Let us consider protocol \(\pi\). Let as assume that during the protocol Alice sent \(n_A\) symbols and Bob sent \(n_B\) symbols. Let \(T(1)\) be the time when Alice sent her \(\varepsilon n_A\)’s symbol, and \(T(i)\) be the time when Alice sent her \(i\varepsilon n_A\)’s symbol. Let \(n_1 = T(1) - \varepsilon n_A\) be the number of messages Bob sent by the time Alice sent \(\varepsilon n_A\) of her messages. Let \(n_i = T(i) - in_A\varepsilon\) be the number of messages Bob sent in the time when Alice sent \(i\varepsilon n_A\) of her communication. Let us set \(f(x)\) to be equal \(\frac{n_{i+1} - n_i}{\varepsilon n_B}\) in the interval \([i \varepsilon, (i + 1) \varepsilon]\). Note that \(\int_0^1 f(t)dt = 1\). In other words \(f(x)\) is an approximation of the rate at which Bob speaks.

By definition of \(L_2(\alpha')\) we have that \(\beta > L_1(\alpha', f)\) or \(\beta > L_2(\alpha', f)\). If \(\beta > L_1(\alpha', f)\) then from Theorem \[58\] we will get that there exists a constant round protocol in which Alice solves \(\pi\). Thus let us assume that \(\beta > L_2(\alpha', f)\).

Thus by definition of \(L_2(\alpha')\) there exists \(s \in [0, 1]\) and \(0 \leq g(t) \leq 1\) such that \(\int_0^s g(t)dt \geq \frac{s - 2\alpha' + 1}{2}\) such that \(\int_0^s f(t)g(t)dt < \beta\).

Let \(t = \left[\frac{s - 2\alpha' + 1}{2\varepsilon}\right]\) We can assume w.l.o.g. that \(g = 1_{S'}\) where \(S'\) is union of \(t\) of intervals of form \([i \varepsilon, (i + 1) \varepsilon]\) plus part of such interval. Let us define
\[
S = \cup_{k=1}^t [(i_k - 1)\varepsilon, i_k \varepsilon] .
\]
Then it holds that
\[
\int_0^s f(t)1_Sdt < \beta
\]
Let us now assume that Eve has erased communication of Bob on the intervals \([T(i_k - 1), T(i_k)]\) (from the definition of \(S\)) and the communication of Alice in all the rest intervals before \(s\). Note that since \(\mu(S) \geq \frac{s - 2\alpha' + 1}{2} - \varepsilon\) we have erased at most \(s - \left(\frac{s - 2\alpha' + 1}{2} - \varepsilon\right) \leq \frac{s + 2\alpha - 1}{2}\)
fraction of the communication of Alice before $s$ and at most $\beta$ fraction of communication of Bob. Note that after time $s$, Alice communicates $1-s$ fraction of her communication. Thus after time $s$ Eve can corrupt at least $\alpha - \frac{s + 2\alpha - 1}{2} = \frac{1-s}{2}$. In other words Eve can corrupt $\frac{1}{2}$ of communication of Alice, after time $s$.

We claim that by time $s$ Bob knows the answer. Therefore we are done since then we now have a $\frac{2}{3}$ round protocol in which Bob solves the task. The proof of this fact is the same as in Theorem 58.

\section{Calculating the Function $L_2(\alpha)$}

Assume that $\alpha \leq \frac{1}{3}$. Here we are going to use the following function. $\tilde{f}_\alpha(x) = 2^i$ in the interval $(i\alpha, \min\{(i+1)\alpha, 1 - 2\alpha\}]$ and in the interval $[1 - 2\alpha, 1]$ it will be $\left(1 + \frac{1}{2}\right)^{2\left\lfloor \frac{1}{\alpha} \right\rfloor - 2}$. Then we will define $f_\alpha$ to be normalization of $\tilde{f}_\alpha$. Let us also define $F_\alpha(s) = \int_0^s f_\alpha(t)dt$. The hardest part of this subsection is the following lemma

\begin{lemma}
$L_2(\alpha, f_\alpha) = L_1(\alpha, f_\alpha) = \frac{1}{2} \left(1 - \frac{1}{(1 + \{\frac{1}{\alpha}\})2\left\lfloor \frac{1}{\alpha} \right\rfloor - 1} - 1\right)$.
\end{lemma}

In particular we will get that $L_1(1/k) = L_2(1/k) = \frac{1}{2}(1 - \frac{1}{2k-1})$.

\begin{proof}
First let us calculate $L_1(\alpha, f_\alpha)$. To do so we need to calculate for every $s$ the value $\min\{\int_0^s f(t)g(t)dt + \frac{1}{2}(1 - F_\alpha(s)) : \int_0^s g(t)dt \geq s - \alpha\}$. For $s \leq 1 - \alpha(1 + \{\frac{1}{\alpha}\})$ the function which minimizes this expression is $g = 1_{[a,s-a]}$. Thus for $s < 1 - 2\alpha$ this expression is equal to $\frac{1}{2}(1 - F_\alpha(s)) + F(s - \alpha)$ Consider the function:

$$T(s) = \frac{1}{2}(1 - F_\alpha(s)) + F_\alpha(s - \alpha).$$

Note that $T'(s) = f_\alpha(s - \alpha) - \frac{1}{2} f_\alpha(s) \geq 0$. Thus $T(s)$ is non-decreasing. We have that $T(\alpha) \leq T(s)$ for every $s \geq \alpha$. Thus for $s \leq 1 - \alpha(1 + \{\frac{1}{\alpha}\})$ we can get at least $T(\alpha)$. For $s > 1 - \alpha(1 + \{\frac{1}{\alpha}\})$ the optimal $g$ is 1 on interval $[0, \ldots 1 - \alpha(2 + \{\frac{1}{\alpha}\})]$ and 1 on interval $[1 - \alpha(1 + \{\frac{1}{\alpha}\}), s]$. Therefore value of $L_1$ for $s > 1 - \alpha(1 + \{\frac{1}{\alpha}\})$ is

$$T(s) = F_\alpha(1 - \alpha(2 + \{\frac{1}{\alpha}\})) + F_\alpha(s) - F_\alpha(1 - \alpha(1 + \{\frac{1}{\alpha}\})) + \frac{1 - F(s)}{2}.$$

Again $T'(s) = \frac{1}{2} f(s) \geq 0$. Thus minimum is achieved at $s = 1 - \alpha(1 + \{\frac{1}{\alpha}\})$. Direct calculation shows that $T(\alpha) < T(1 - \alpha(1 + \{\frac{1}{\alpha}\}))$. Therefore

$$L_1(\alpha, f_\alpha) = \frac{1}{2}(1 - F_\alpha(\alpha)) = \frac{1}{2} \left(1 - \frac{1}{(1 + \{\frac{1}{\alpha}\})2\left\lfloor \frac{1}{\alpha} \right\rfloor - 1} - 1\right).$$

\end{proof}
To calculate \( L_2(\alpha, f_\alpha) \) note that for \( s \geq 1 - 2\alpha \) an optimal \( g \) is 1 on \([0, 1 - \alpha(2 + \{\frac{1}{\alpha}\})]\) and 1 on \([s-2\alpha(1+\{\frac{1}{\alpha}\})+1, s] \). Thus

\[
T_2(s) = F_\alpha(1 - \alpha(2 + \{\frac{1}{\alpha}\})) + F_\alpha(s) - F_\alpha\left(\frac{s - 2\alpha(1 + \{\frac{1}{\alpha}\}) + 1}{2}\right).
\]

Again differentiate \( T_2 \) we get that \( T_2'(s) = f(s) - \frac{1}{2}f\left(\frac{s - 2\alpha(1 + \{\frac{1}{\alpha}\}) + 1}{2}\right) \). It is easy to see that \( T'(s) > 0 \) for \( s > 1 - 2\alpha \). Thus minimal is achieved when \( s = 1 - 2\alpha \) Thus

\[
L_2(\alpha, f_\alpha) = F_\alpha(1 - 2\alpha) = \frac{1}{2}(1 - F_\alpha(\alpha)) = \frac{1}{2}\left(1 - \frac{1}{(1 + \{\frac{1}{\alpha}\})2^{\lfloor \frac{1}{\alpha}\rfloor - 1} - 1}\right).
\]

\[\blacksquare\]

**Theorem 67.**

\[ L_2(\alpha) = L_1(\alpha, f_\alpha) = L_2(\alpha, f_\alpha) .\]

**Proof.** From Lemma 66 we know that \( L_1(\alpha, f_\alpha) = L_2(\alpha, f_\alpha) \) let us denote this value by \( C \). Let us assume that we have function \( g \) which is better than \( f_\alpha \) i.e., \( \int_0^1 g(t)dt = 1 \) and

\[
\min\{L_1(\alpha, g), L_2(\alpha, g)\} \geq \min\{L_1(\alpha, f_\alpha), L_2(\alpha, f_\alpha)\} = L_1(\alpha, f_\alpha) = L_2(\alpha, f_\alpha) = C
\]

Denote by \( G(s) = \int_0^s g(t)dt \). Then by taking \( s = 1 - 2\alpha \) in definition of \( L_2 \) we have that \( G(1 - 2\alpha) \geq L_2(\alpha, g) \geq F_\alpha(1 - 2\alpha) = C \). Thus we have that \( L_1(\alpha, f) \leq L_1(\alpha, g) \leq L_1(\alpha, 1 - 2\alpha, C) \). Now to complete the proof we need to show that \( L_1(\alpha, f_\alpha) = L_1(\alpha, 1 - 2\alpha, C) \). This follows from Theorem 59. The conditions 1-3 follows immediately. To see the condition 4 note that for \( s \in [\alpha, \beta] \) it holds that \( f_\alpha(s) = 2f_\alpha(s - \alpha) \). \[\blacksquare\]

### 17 Proof of Theorems 4 and 5

Since from Theorems 64, 65 it follows that \( L_2 \) is the boundary of \((\alpha, \beta)\) we can handle \( L_2 \) must be symmetric in \( \alpha, \beta \) i.e., if \( L_2(\alpha) = \beta \) then \( L_2(\beta) = \alpha \). Also note that \( L_2 \) is decreasing function and that \( L_2(\frac{1}{3}) = \frac{1}{3} \). Therefore if \( \beta = L_2(\alpha) \) then \( \alpha \leq \frac{1}{3} \) or \( \beta \leq \frac{1}{3} \). Thus the region \( \mathcal{R}_a \) can be defined by

\[ \mathcal{R}_U = \{(\alpha, \beta) : \beta < L_2(\alpha)\} . \]

**Theorem 68.** For each \((\alpha, \beta) \in \mathcal{R}_U\) and for every protocol \( \pi \) there exists another protocol \( \pi' \), with \( CC(\pi') = O_{\alpha, \beta}(CC(\pi)) \) which is resilient \((\alpha, \beta)\) adversarial noise. Such that \( \pi'(x, y) \) outputs transcript of \( \pi(x, y) \).

**Proof.** This follows from the Theorem 64 and the fact that Pointer Jumping Problem is complete for communication complexity. \[\blacksquare\]

**Theorem 69.** For every \((\alpha, \beta) \notin \mathcal{R}_U\) and for every \( T \) the following holds. Let \( \pi' \) be a protocol resilient to \((\alpha, \beta)\) adversarial noise which solves the Pointer Jumping Problem of depth \( T \). Then \( CC(\pi') = 2^{O_{\alpha, \beta}(T)} \).
Proof. From Theorem 65 it follows that there exists constant $O_{\alpha,\beta}(1)$ round protocol $\pi''$ with the same communication complexity which solves one way pointer jumping problem. Then from Corollary it follows that communication complexity of $\pi''$ is $2^{\Omega_{\alpha,\beta}(T)}$. 

18 Conclusions

In this paper we completely answered the question of characterizing the maximal error rate we can handle in adversarial interactive communication with a constant-rate encoding. We have defined list decoding of interactive communication and showed that this primitive is also important for unique decoding of interactive communication. Our encodings are constant rate, i.e. only incur constant blow-up in communication. However we have not tried to optimize this constant, leaving such optimization as an open problem. Below we highlight this and additional outstanding open problems:

Open Problem 1. The big problem is to understand what is the best rate is possible for interactive communication. We want to note that this question is widely open even in the random noise scenario. Recently there was some progress on this question in random noise scenario with noise rate approaching 0 by Kol and Raz [KR13].

Open Problem 2. In this paper we made an encoding over a channel with a large constant-size alphabet. Without much effort one can modify all our protocols to work over a binary channel with a loss of a factor two in the error rates one can handle. However it is not clear at all that these error rates are best possible. Establishing the region of $(\alpha, \beta)$ for which unique decoding is possible over binary alphabet is an open problem.

Open Problem 3. Construct explicit list tree codes with computationally efficient encoding and decoding. It is plausible that construction list-decodable tree codes will be an easier problem than just a regular tree codes since, similarly to potent tree codes from [GMS11], a random prefix code is a list decodable tree code with high probability, while it is not a tree code with high probability.

References


Peter Elias. List decoding for noisy channels. 1957.


