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 α Alice's communication and up to a fraction β of Bob's communication. We use list-decoding to characterize the region fully $\mathcal{R}_U$ of pairs (α, β) 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 (α, β) 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 Σ, which is affected by an adversarial noise of rate η. In other words, if the total number of symbols transmitted by Alice and Bob is N (which is known *a priori* to

*An extended abstract of this paper appeared in the proceedings of FOCS'14

†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).

‡Tel-Aviv University. Part of this work was done while the author was Simons Institute at Berkeley at the Department of Computer Science, University of Chicago, supported by a Simons Fellowship in Theoretical Computer Science; and a member of the Institute for Advanced Studies and funded by NSF CCF-0832797 and DMS-0835373 European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.(klimefrem@gmail.com).

all the participants), then the adversary is allowed to corrupt at most ηN symbols of the transmission. The goal is to provide a scheme that can simulate any communication protocol in an error-resilient way.

The non-interactive version of the problem is the well-studied problem of encoding a message $M \in \Sigma^N$ with an error-correcting code $C : \Sigma^N \to \Sigma_2^{N'}$ resilient to adversarial errors. To be resilient to errors of rate η, 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 that hamming distance (we will denote it by d_H through this paper) between any two codewords will be at least $2\eta N'$ i.e., $d_H(C(M_1), C(M_2)) > 2\eta N'$ for all $M_1, M_2 \in \Sigma^N$. A code C is said to be good if it has a constant rate: $N' = O(N)$ 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 η. There are several well-known constructions of good codes.

Once the error rate η exceeds $1/2$, there is no hope of recovering from a fraction η of errors, since for any M_1, M_2 there is a message $\tilde{M}$ such that $d_H(C(M_1), \tilde{M}) \le \eta N'$ and $d_H(C(M_2), \tilde{M}) \le \eta N'$, 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 η-list-decodable with list of size L if for every word $\tilde{M} \in \Sigma_2^{N'}$, the number of codewords within relative distance η 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 corrupt a single round completely. 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 ith 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 depend 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 limiting 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 (non-shared) 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 [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. Non-robust protocols have recently received some attention. In particular, very recently [AGS13] and [GHS13] considered non-robust models and concluded that, indeed nonadaptive protocols can withstand a higher error rate. It would be interesting to combine these results, particularly those from [GHS13], with ours to calculate the tight error-rate region for the non-robust 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 analog 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 η, each list will contain the intended conversation. We show that constant-rate interactive list-decodable coding is possible for all error rates $\eta < 1/2$, 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 (α, β) of error rates, with α representing the fraction of Alice's communication that may be corrupted and β representing the fraction of Bob's communication that may be corrupted. Using our list-decoding results, we can give a precise characterisation of the region $\mathcal{R}_U$ of pairs (α, β) of error rates for which constant-rate unique decoding is possible. Previously, by the construction of Braverman and Rao [BR11], unique decoding has been known to be possible when $\alpha + \beta < 1/2$. At the same time, it is easy to see that unique decoding is impossible when $\alpha \geq 1/2$ or $\beta \geq 1/2$, therefore it is impossible to improve bound on $\alpha + \beta$, but if both parties know (α, β) we can perform unique decoding in a larger region. 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. When Alice and Bob are affected by an equal amount of error, the intersection of $\mathcal{R}_U$ with the line $\alpha = \beta$ is the region $\{(\alpha, \alpha) : \ \alpha < 1/3\}$. Thus we can handle error of up to $1/3 - \varepsilon$ affecting each of Alice and Bob. Previously, only a lower bound of $1/4 - \varepsilon$ and an upper bound of $1/2$ were known [BR11][1].

[1] In our work we assume that (α, β) is known to both parties while in [BR11] only bound on sum $\alpha + \beta$ was assumed

Recent Related Works

We would like to mention very recent related works of Ghaffari, Haeupler, and Sudan [GHS13, 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 central question of [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 [GHS13, 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 a 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 $n \cdot 2^{O(\log^* n \cdot \log\log^* n)}$. 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.[2] 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 (α, β) adversarial noise if an adversary can corrupt up to α fraction of Alice's messages and up to β fraction of Bob's messages. For a formal definition, see Section 4.

Theorem 1. *For each $\varepsilon > 0$ and for every protocol π, there exists another protocol π', with $CC(\pi') = O_\varepsilon(CC(\pi))$, that is resilient (α, β) 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 (α, β) with $\alpha + \beta \geq 1$, list-decoding is impossible. The Pointer Jumping Problem can be viewed as the generic communication complexity problem and is formally defined in Section 3.

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

We note that in a noiseless regime, one can solve Pointer Jumping Problem of depth T with communication complexity T. Therefore, if $\alpha + \beta \geq 1$ one cannot perform list decoding with constant stretch in communication.

[2] Haeupler, personal communication.

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

We can give an even tighter result by considering a slightly generalized error notion in Theorem 1. Let us consider the only standard protocols π', where Alice and Bob send one message at a time in an alternating fashion. We can partition such a π' 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 η be the fraction of blocks the adversary corrupts. Note that it is always the case that $\max(\alpha, \beta) \le \eta \le \alpha + \beta$. We show that we can handle η-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 π, there exists another protocol π' with $CC(\pi') = O_\eta(CC(\pi))$ which is resilient η-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 $\eta \le \alpha + \beta$, Theorem 3 implies Theorem 1.

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 π', Alice and Bob need to be able to recover the original protocol transcript π correctly. In Section 13 we also consider the asymmetric case, where only Alice needs to recover π uniquely.

We study the set $\mathcal{R}_U$ of pairs (α, β) 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

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

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 (α, α).

Theorem 4. *For each $(\alpha, \beta) \in \mathcal{R}_U$ and for every protocol π, there exists another protocol π' with $CC(\pi') = O_{\alpha,\beta}(CC(\pi))$, which is resilient (α, β) 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 (α, β) 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.

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

Recall that 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 5 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 (non-interactive) 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 demonstrates 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 $\partial\mathcal{R}_U$. We conjecture that similarly to the list-decoding case, the boundary belongs to the region where decoding is not feasible.

Acknowledgment

We thank Ran Gelles for the many insightful comments on an earlier draft of this paper. We thank the anonymous referees for their thorough reading of the paper, and for the numerous comments which helped significantly improve the presentation in the paper.

2 Main Techniques and Discussion

2.1 List-Decoding

We start with a discussion of the proof of the positive results of interactive list decoding. First, we note that any protocol π that sends T bits could be reduced to a Pointer Jumping Problem of depth $2T$. Pointer Jumping Problem 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 $\alpha + \beta < 1/2 - \varepsilon$ [BR11]: in other words, in sufficiently many rounds, Alice and Bob will correctly reconstruct each other's transmissions so far.

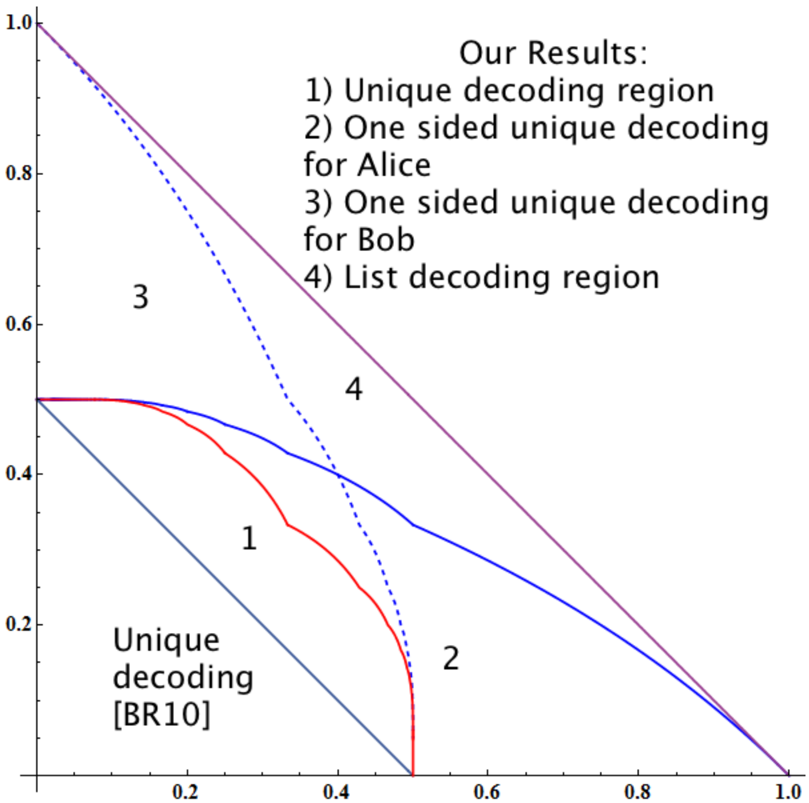

Figure 1: Our results: a diagram depicting the regions of error-rates which can be corrected in the different error models. Region 1 corresponds to error rates where unique decoding is possible; in regions 2 and 3 only Alice (resp. Bob) can unique decode; in region 4 both parties can list-decode. Note the inclusions $1 \subset 2 \subset 4$, $1 \subset 3 \subset 4$. The previously known unique decoding region of [BR10] is also shown.

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 simultaneously to make progress on all these communications. Therefore, a list size ℓ causes an ℓ-fold explosion in communication and ℓ needs to be kept constant at (almost) all times. Unlike one-way communication, this approach has a major problem: if Alice sends ℓ responses, Bob will decode them and will need to send ℓ responses for each of Alice's responses; thus Bob will need to send ℓ^2 responses, and this will exponentially blow up communication. 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 with this approach is that to encode a single edge we require $O(T)$ (where T is the depth of the tree) bits. Thus we will add a 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 the so basic approach of Braverman and Rao will not work here. A substantial amount of technical work is required to keep the compression at the rate of $O(1)$ bits per edge.

For Alice and Bob to be able to perform list-decoding in interactive settings, we define the analog 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 ℓ 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 [GMS11].

To summarize, in this paper, we are going to have two levels of coding. The first one is compression, which will allow us to take a list of edges from the Pointer Jumping Problem 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 choose 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 adjust the definition of list-tree code carefully to make the compression 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 (non-list 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 π, and Bob has at most two candidate transcripts, π_1 and π_2. He can then use the second (noncorrupt) part of Alice's transmission to decide whether to output π_1 or π_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 $\alpha + \beta < 1/2$ barrier, and get an error-correcting scheme that works for (α, β) in the region:

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

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

In our primary 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 (α, β) outside of $\mathcal{R}_2$, we can still achieve unique decoding by having Alice and Bob alter the relative rates of which they speak throughout the execution of the scheme. Note that our scheme is still robust; therefore, these rates will be predetermined by (α, β). 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 complex 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 α-fraction of Alice's communication and the last β 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 (α, β)-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 (α, β)-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 in which Alice and Bob speak at the same rate. Then the adversary can choose either Alice or Bob and try to corrupt her or his output. Suppose the adversary chooses to corrupt Alice's output. In this case first, α fraction of communication the adversary can just erase all Alice's communication. If the protocol cannot be solved in one round, then after the first α fraction of communication, there are still two possible inputs for Bob that are consistent with this communication. Therefore, the adversary can toss a coin every round and play according to one of these two inputs for Bob. This way Alice will not be able to distinguish between these two inputs, and the adversary corrupts only $\frac{1-\alpha}{2}$ of Bob's communication. Thus we see that if $\alpha + 2\beta > 1$, then no unique decoding is possible. When Alice and Bob do not speak at the same rate, this argument is no longer valid. The main technical challenge is to adjust the preceding argument when Alice and Bob speak at different rates. By adjusting this argument to different rates, we will get the region $\mathcal{R}_U$.

Let us now try to give a brief idea of the region $\mathcal{R}_U$ from the point of view of upper bounds. Consider the simpler scheme when only Alice is interested in the correct answer. Let us assume that the adversary can corrupt up to an α fraction of Alice's communication. In this case, to corrupt Alice's answer, the adversary can (and we prove that this is essentially the only thing it can do) pick some point of the communication, s. Before s, the adversary will corrupt enough Alice's and Bob's communication that Alice and Bob will not be able to perform any interactive communication at all. After s, the adversary will corrupt half of Bob's communication, so that Alice will output a 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, we can see that the adversary will always be interested in corrupting Alice's communication before point s. Let us assume that Bob's rate relative to Alice is an increasing function (we will proof that we can indeed assume that). In this case for every fixed s, a minimum number of errors adversary needs to invest on Bob's side is exactly Bob's communication first $s - \alpha$ rounds and $\frac{1}{2}(1 - s)$ last Bob's rounds. If Bob's rate is doubled every α fraction of Alice's communication then for every s we will get the same function. By writing linear inequalities we see that this is optimal scheme for Bob. Note that when α^{-1} is not an integer, the number of α intervals is not an integer, and the last interval is truncated. This is where the non-smoothness of $\mathcal{R}_u$ comes from.

Mathematically, the doubling of the rate is a consequence of correctly balancing the various constraints corresponding to the various strategies the adversary may employ. Intuitively, there will be a special point in the protocol when the full correct transcript will be list-decoded — the timing of this point depends on the adversary's strategy. Assume α is small (otherwise α^{-1} is small, and there is only one block). By our list-decoding result, to delay this point, the adversary needs to essentially turn the conversation into a bounded-round one by maintaining the sum of the error-rates $\alpha' + \beta' > 1$ — intermittently shutting down one party or another. To the extent the adversary would like to shut down Bob, it needs to corrupt *all* of his messages. After that point, and until the end of the protocol, to confuse Alice it's enough to only corrupt *half* of Bob's messages: the factor two here is the difference of the rate resilience of list vs. unique decoding. Thus, in some sense, it is "cheaper" to corrupt Bob later in the game — after the special point. At the optimal protocol, all the possible strategies of the adversary just barely fail, which means that moving the special point should not make a difference, thus later in the protocol Bob should have a higher transmission rate – so that corrupting half of Bob a bit later in the protocol costs as much as corrupting all of Bob a little sooner.

3 Communication Protocols, Pointer Jumping and Errors

For sake of completeness, we include section from [BR11].

Given inputs, x, y from some domain, a deterministic protocol with alphabet Σ 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 Σ. 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 whom 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 an obvious way.

In this paper, we consider only 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 $\mathcal{T}$ be a rooted s-ary tree of depth T. Let $\mathcal{X}$ denote the set of edges leaving vertices at the 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 to 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 before, 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 three problems in this paper: The first one is: the Pointer Jumping Problem where both Alice and Bob have to compute $v(X \cup Y)$. The second one is the 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 the list-decodable Pointer Jumping Problem, where at the end of the protocol, both Alice and Bob have to output a 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 $f(x, y)$ that is solvable in T communication complexity, but every k round protocol $\pi(x, y)$ that computes $f(x, y)$ with probability at least $\frac{2}{3}$ has communication complexity at least $\exp(\Omega(\frac{T}{k}))$.*

Proof. Set $T = O(\frac{\log n}{k})$ and use Theorems 2.3 and 2.6 from [NW93]. □

Corollary 9. *For every $\varepsilon, k < T$, if protocol π solves one way $PJP(2T, 2)$ in $k - 4$ rounds with probability ε, then the communication complexity of π is at least $\Omega(\varepsilon \cdot \exp(\Omega_k(T)))$.*

Proof. Let us assume that we have protocol π that solves the one-way $PJP(2T, 2)$ in $k - 4$ rounds with probability ε. Let us show that we can solve $PJP(2T, 2)$ in k rounds with probability $\frac{2}{3}$ and with communication complexity $O(\frac{CC(\pi)+T}{\varepsilon})$. Alice and Bob can execute protocol π in parallel $t = \frac{10}{\varepsilon}$ 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 right answer, we can find it in four rounds and with $O(\frac{T}{\varepsilon})$ 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 right answer and tell it to another party.

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

4 Definition of an Adversarial Error Protocol

Let us denote by $D(\Sigma)$ the set of probability distributions over Σ. 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 a fraction of a symbol; that is, each party sends a symbol, but the receiver may receive a distribution over symbols. The amount of error accrued to the adversary if the sent symbol is a and the received distribution is b is $1 - b(a)$. For simplicity of notation, we will identify Σ with probability distributions supported on a single symbol.

Remark 10. *Note that this model is stronger than the model where an adversary can replace only symbol by another symbol. Since replacing one symbol by another symbol in this model is equivalent to just sending a 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 streamlines notation and simplifies proofs in the section of unique decoding.

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 define agreement 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 π' list-decodes f with a list of size L if, for every input x, y, both parties of the protocol of $\pi'(x,y)$ output a list of size at most L with $f(x,y)$ in the list.*

Definition 12. *We say that a protocol π is a 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; that is, who is speaking at round i depends only on i and not on randomness or input of the protocol x, y. The alphabet of our protocols will be Σ_{out}. We will say that the communication complexity of the protocol is the maximal over all inputs of bits sent by our protocol. Alice's (Bob's) communication complexity $n_A(n_B)$ is the maximal number of symbols(from Σ_{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 deterministic.

Informally the communication protocol resilient to (α, β)-noise is a protocol that outputs the correct answer when an adversary can change an α fraction of symbols sent by Alice and a β fraction of symbols sent by Bob.

Definition 13 (Communication Protocols Resilient to (α, β)-noise)**.** *Assume that we have deterministic protocol $\pi(x,y)$. For $w_A \in D(\Sigma_{out})^{n_B}$, let $\pi_A(x)(w_A)$ be a messages sent by Alice when she received w_A from Bob and $\pi_B(y)(w_B)$ is the same for Bob. We say that the protocol computing $f(x,y)$ is resilient to (α, β)-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*

1. $$d(\pi_A(x)(w_A), w_B) \le \alpha n_A.$$

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

It holds that both Alice and Bob can compute $f(x,y)$ from $\{x, w_A\}$ and $\{y, w_B\}$ respectively.

Let us define even more restrictive definition of noise. We will divide our communication in to blocks of two so that in each block both Alice and Bob exchange messages. We will say that some block is corrupted if a message of one of the parties is corrupted. That is, in this case if adversary decides to corrupt the communication of Alice in some block, he can corrupt all the communication of Bob in this block at the same cost. In other words we count number of exchanges that where corrupted. Let us give formal definition.

Definition 14 (Communication Protocols Resilient to α-Symmetric Noise)**.** *In this definition, we will consider only protocols where Alice speaks at each even round and Bob speaks at odd rounds. The protocol runs for $2n$ rounds.*

We say that the protocol computing function $f(x, y)$ is resilient to α-symmetric noise if, for every input x, y and for every $w_A \in D(\Sigma_{out})^n, w_B \in D(\Sigma_{out})^n$ such that ,

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

Alice and Bob can compute $f(x, y)$ from $\{x, w_A\}$ and $\{y, w_B\}$ respectively.

Part I
List Decoding

5 List-Tree Codes

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

We will denote by Σ^* all strings of all length over alphabet Σ.

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

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

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

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

Definition 17. *For every $w \in D(\Sigma_{out})^n$, the ith-level ε-list of w under C is given by*

$$List_i(w,C,\varepsilon) := \{x \in \Sigma_{in}^i : \ \delta_s(C(x), w[1,\ldots,i]) \le 1-\varepsilon\}.$$

The ε-list of w under C is given by

$$List(w,C,\varepsilon) := \bigcup_{i=1}^{|w|} List_i(w,C,\varepsilon).$$

We also define

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

We will identify all codewords Σ_{in}^* with a nodes of the full Σ_{in}-ary tree, where codewords of length d will be at depth d, where if $x \in \Sigma_{in}^i, c \in \Sigma_{in}$ then x will be a father of $x \circ c \in \Sigma_{in}^{i+1}$. In this case we will write c on the edge going from x to $x \circ c$. The tree $PrefixList$ is the rooted subtree that spans all the nodes in $List$. In Figure 2 we give an example of $PrefixList$. Note in this example that $|\cup List_i| = 5$. The size of PrefixList is 9.

For any rooted subtree PL of a full Σ_{in}-ary tree of depth at most $|w|$, we will denote by $w(PL)$ is the tree where we write $w[i]$ on all edges at depth i. We denote by $C(PL)$ a labelling of PL by code C where we label edge going to v by $C(v)$(recall we identify v with some string in Σ_{in}^*). We will always consider trees in this paper as one-way graphs directed from root downward. If we have some tree PL and two different labelings of nodes $w(PL), C(PL)$ then $d(w(PL), C(PL)) = \sum_{e \in PL} d(w(e), C(e))$ and $agr(w(PL), C(PL)) = |PL| - d(w(PL), C(PL))$. Note that if P is some path in the tree ending at v, then $C(P)$ is just a labeling of this path. The string that we get from this labeling is a suffix of $C(v)$ since $C(v)$ is the labeling of path from root v.

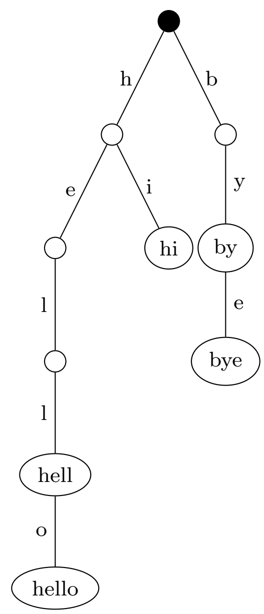

Figure 2: PrefixTree with $List_2 = \{$"*hi*", "*by*"$\}, List_3 =$ "*bye*"$, List_4 =$ "*hell*"$, List_5 =$ "*hello*".

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 by definition $C(P) = C(v)[i - k - 1, i]$ as well. Thus from the definition of δ_s, the lemma follows. □

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

Definition 19. *An (ε, L)-*list-tree code *$C : \Sigma_{in}^n \to \Sigma_{out}^n$ 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 $PL(w) = PrefixList(w, C, \varepsilon)$; 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, the first property is included in the second property. We stated the first property here since we are going to use mainly this property of the code. The only reason we require w to be string of distributions rather than just a 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 $|List_i| < L$ for almost all i.*

Lemma 21. *Let $PL \triangleq PrefixList(w, C, \varepsilon)$. Then $agr(C(PL), w(PL)) > \varepsilon |PL|$.*

Proof. We will prove this by induction on the number of leaves. If PL has only one leaf v, then PL is a path from the root to v, and the claim follows from the Lemma 18.
Induction step: Let PL be a tree with i leaves. Let P be a branch of v (i.e., a path from w to v, where w is a first predecessor of v which has more than one child.) Then $PL \backslash P$ has one leaf less than PL. Thus from induction we see that

$$agr(C(PL \backslash P), w(PL \backslash P)) > \varepsilon(|PL| - |P|).$$

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

$$agr(C(PL), w(PL)) = agr(C(PL \backslash P), w(PL \backslash P)) + agr(C(P), w(P)) > \varepsilon |PL| \ .$$

□

Theorem 22. *For all $0 < \varepsilon < 1$, Σ_{in}, let $|\Sigma_{out}| > (2|\Sigma_{in}|)^{3/\varepsilon^2}$. Then a random prefix code $C : \Sigma_{in}^n \to \Sigma_{out}^n$ is a (ε, 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 a full d-ary tree. Then there exist 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 indicates to what child to go, and symbol $d + 1$ will say to go up. Next, note that if we make DFS a 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 with $2s$ symbols. □

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

Let us assume that w is a word that violates the conditions of the Definition 19 . Then Lemma 21 also implies that if $PL = PL(w)$, then $agr(C(PL), w(PL)) > \varepsilon \max\{|PL|, Ln\}$. Now we are going to prove that w.h.p. $agr(C(PL), w(PL)) < \varepsilon \max\{|PL|, Ln\}$ for every subtree PL and for every w. It is enough to prove the claim for $|PL| \geq Ln$, since if $|PL| < Ln$ extend PL to any subtree PL' of size $|Ln|$ and for this subtree it will holds that $agr(C(PL'), w(PL')) < \varepsilon Ln$ and therefore $agr(C(PL), w(PL)) < \varepsilon Ln$. The following claim gives an upper bound on the probability that the conditions of the list-tree code broken for specific subtree PL and specific $w \in \Sigma_{out}^n$.

Claim 24. *For every $w \in \Sigma_{out}^n$ and for every tree PL of size at most s, it holds that*

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

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 see that the probability that C is not a 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 a list-tree code is bounded by

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

Since $|\Sigma_{out}| > (2|\Sigma_{in}|)^{\frac{3}{\varepsilon^2}}$, the preceding quantity is bounded by

$$(2|\Sigma_{in}|)^{\frac{3n}{\varepsilon^2}} \sum_{s=Ln}^{\infty} (2|\Sigma_{in}|)^{-\frac{3s}{\varepsilon}} 2^s (|\Sigma_{in}|+1)^{2s} \leq 2(2|\Sigma_{in}|)^{\frac{3n}{\varepsilon^2}} (2|\Sigma_{in}|)^{-\frac{3Ln}{\varepsilon}} 2^{Ln} (|\Sigma_{in}|+1)^{2Ln},$$

since $(4|\Sigma_{in}|)^{-3/\varepsilon}(|\Sigma_{in}|+1)^2 < 1$. Recalling that $L = \frac{1+\varepsilon}{\varepsilon}$, we can bound, the preceding as

$$4(|\Sigma_{in}|)^{-\frac{3}{\varepsilon}n} 2^{\frac{1+\varepsilon}{\varepsilon}n} (|\Sigma_{in}|+1)^{\frac{2+2\varepsilon}{\varepsilon}n} \leq 2^{-n/\varepsilon} \leq 2^{-n}.$$

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 Σ_{out}^n. □

6 Communication Transcript

During our protocol, Alice and Bob are going to send edges from the tree $\mathcal{T}$ of the original protocol. To describe some particular edge from $\mathcal{T}$, it may take $O(T)$ bits. Also, as in Braverman and Rao [BR11], we will need a way to represent edges that will require much fewer bits. The first idea is that we are not sending random edges; we never send any edge before we send its grandparent. Therefore, we can describe an edge by a link to the grandparent and a path from the grandparent to an edge. This will solve the problem for the case in which we have the alphabet polynomial in the length of our communication since in this case we do not care about the length of the links. In the case of a constant-size alphabet, the length of the link is important and therefore, we instead of sending a link to the grandparent, we will sometimes send the link to the cousin, who may be much closer to the edge we are sending.

Since links to a cousin look a little bit mysterious let us try to give a brief idea as to why it is necessary. Let us consider a tree of size t. Let us assume that we have a subset of nodes of this tree that we call special nodes. Let us also assume that every special node has a link pointing to the other special node. The length of the link is the difference in depth between nodes. If every special node points to its predecessor in this case we have no control other the sum of (or even logs of) lengths of links. Say consider an example when one special node is the root and others are at the bottom. However, if we allow links to cousin's, then we will show that the sum of the length of links is bounded by the size of the tree (for more details see Claim 46). In our protocol links that are posted by a party are sent in full and the transmission is never aborted. It is possible that a protocol in the spirit of [BR11] can avoid the use of cousin links and only use grandparent links, but it is not clear the protocol and its analysis would in fact be simpler.

Each entry of our transcript will correspond to some edge in $\mathcal{T}$. Every entry $a_i = (r_i, b_i, s_i)$ of the transcript will consist of an integer $r_i \in \mathbb{N}$, which will be a pointer to another edge that appeared r_i entries earlier in our transcript. A bit $b_i \in \{0, 1\}$ will indicate whether the reference edge is a grandparent or a cousin,[3] and $s_i \in \Sigma_T^2$ will indicate a path from a grandparent to the edge. We will assume that an integer k takes at most $2 \log k + 2$ bits to encode.

For $A = (a_1, a_2, \dots a_k)$ we have the procedures $E(A)$ which returns edges encoded by A, and $Add(A, e)$, which adds edge e to transcript A.

Procedure for decoding $E(A)$: Now let us describe formally how we will decode our transcript for $i = 1, \dots, k$ 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 a depth of at most 2 specified by bits s_i.
2. If $r_i \geq i$, return an error message.
3. If $b_i = 0$, set $p_i = e_{i-r_i}$.
4. If $b_i = 1$, set p_i to be the 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 take (at most) two steps in the tree using the bits s_i.

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

[3] In fact, we will need cousins only for a small alphabet.

Remark 25. *Note that the procedure Add does not work for any edge e, only for edges at depth at most 2 from already added edges.*

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

- Procedure **size**(A): Returns the size of the transcript in bits.
- Procedure **End** $-$ **round**(A, i): Sets the size of the transcript to be maximum between $\log \Sigma_{in}^{i}$ [4] and $size(A)$ and if necessary pads the 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 π that is resilient to $(1-\varepsilon)$-symmetric noise and that solves the problem of list-decoding of $PJP(T, \Sigma_T)$, where $|\Sigma_T| = T^c$ with list of size $O(\frac{1}{\varepsilon})$. Moreover, the protocol π 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. Also define $L = \frac{1}{\varepsilon'} + 1$. During the protocol, we are going to send $O(T)$ edges and encode them with the transcript defined in the previous section. The links in the transcript will be at most of size $O(T)$; therefore, every entry of the transcript takes at most $O(\log T)$ bits. Let us set Σ_{in} to be large enough to hold $Ent = \frac{L}{\varepsilon'} = O(\frac{1}{\varepsilon^2})$ 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 exists $C : \Sigma_{in}^n \to \Sigma_{out}^n$, which is the $(\varepsilon', L) = (O(\varepsilon), O(\frac{1}{\varepsilon}))$ 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 Σ_T of size T^c.
Second is Σ_{in}, which corresponds to a non-encoded single message that we are going to send during each round.
Third is Σ_{out} that corresponds to an alphabet which we are going to send over noisy channel.

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

[4] Σ_{in} will be defined later.

Alice's and Bob's protocols of will are symmetric so that we will introduce only the protocol for Alice. We will denote by w_A and w_B codewords received by Alice and Bob, respectively. Alice will maintain transcript A, which will be initialized by the empty set. Formally, at round i Alice will do the following:

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. For $j = 1, \ldots, \min\{k, Ent\}$,
   (a) If $E(B_j) \cup E(A)$ has a unique path from the root and let e is a last edge on this path, then $Add(A, e)$.
3. $End - round(A, i+1)$.
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 an* output node *if $E(v) \cup X$ has a unique path from the root, and this path ends at a leaf ℓ and none of predecessors of v has this property. Then $value(v)$ is the leaf ℓ.*

In other words, the output node v is the node that 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 $s \in \Sigma_{out}^{j-i+1}$ be string corresponding to path from v to v_s. Then $agr(v \to v_s, w) \triangleq agr(s, w[i, \ldots, j])$. 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 there exists successor of v, $v_s \in PrefixList(w_A, C, \varepsilon)$ such that $agr(v \to v_s, w_A) > \varepsilon' n$ then output $value(v)$.

In other words, Alice will output all answers that correspond to communication and have at least $\varepsilon' n$ places of agreement with w after the output node.

Lemma 29. *The above protocol solves the task of list decoding the Pointer Jumping Problem with a 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 $\varepsilon' Ln$. 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 are 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 $\frac{\varepsilon' Ln}{\varepsilon' n} = L = O(\frac{1}{\varepsilon})$. □

In the remainder 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 her or 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., $B_{[1,\dots,i-1]} \in List_{i-1}(w_A, C, \varepsilon)$ and $A_{[1,\dots,i-1]} \in List_{i-1}(w_B, C, \varepsilon)$.*
2. *The list size of both Alice and Bob at block i was less than 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 the 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.g. that $e \in X$. Then e is a last edge from $E(B) \cup X$. Thus we will advance in this block. □

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 $PrefixList$, $\sum_{i=1}^{n} |List_i(w_A, C, \varepsilon)| \le |PrefixList(w_A, C, \varepsilon)|$. Since at every bad block we have that $List_i(w_A, C, \varepsilon) \ge Ent = \frac{L}{\varepsilon'}$ (or same for Bob.) We have at most

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

such blocks. □

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

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

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*

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

Then $|S_\alpha(n)| \geq (\beta-\alpha)n$.

Proof of Lemma 35. Recall that by definition of $List_i$, we have that $B[1,\dots,i]$ is in $List_i(w_A, C, \varepsilon')$ if

$$\delta_s(w_A[1,\dots,i], c_B[1,\dots,i]) \leq 1-\varepsilon' \ ,$$

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 $d(w,c) \leq (1-\varepsilon)n$. Using Claim 36 we will get that there are at least $(\varepsilon-\varepsilon')n$ indexes i such that $\delta_s(w_{[1,\dots,i]}, c_{[1,\dots,i]}) \leq 1-\varepsilon'$. For every such i the first condition holds since by definition of w, c it holds that

$$\delta_s(w_A[1,\dots,i], c_B[1,\dots,i]) \leq \delta_s(w[1,\dots,i], c[1,\dots,i]) \leq 1-\varepsilon',$$

And the same for Bob, i.e.,

$$\delta_s(w_B[1,\dots,i], c_A[1,\dots,i]) \leq \delta_s(w[1,\dots,i], c[1,\dots,i]) \leq 1-\varepsilon'$$

□

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

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

Induction step $(n-1) \to n$. 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 $w_{[1,\dots,n-1]}, c_{[1,\dots,n-1]}$ we know that

$$d(w_{[1,\dots,n-1]}, c_{[1,\dots,n-1]}) \leq (1-\beta)n = (1 - \frac{\beta n - 1}{n-1})(n-1)$$

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 $d(c_{[n',\dots,n]}, w_{[n',\dots,n]}) > (n-n'+1)(1-\alpha)$ thus

$$\begin{aligned} &d(w_{[1,\dots,n'-1]}, c_{[1,n'-1]}) < (1-\beta)n - (n-n'+1)(1-\alpha) = \\ &= (n'-1) - (\alpha(n'-1) + (\beta-\alpha)n). \end{aligned}$$

By induction on $n'-1$ we get that

$$|S_\alpha(n)| \geq |S_\alpha(n'-1)| \geq \alpha(n'-1) + (\beta-\alpha)n - \alpha(n'-1) = (\beta-\alpha)n$$

□

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. From the Corollary above it follows that we will have $\varepsilon n - O(\varepsilon' n) > T$ good blocks thus it will be an output node v with a correct value in the PrefixList. In order to show that this answer will be in the output list we need to show agreement between v and its last successor is at least $\varepsilon' n$.

Let $P \subset PL_A$ be a path in PrefixList that corresponds to the correct path (i.e. it is the message that Bob has sent).

If we choose $c_1 < \frac{1}{2}$ (recall that $c_1 = \frac{\varepsilon'}{\varepsilon}$), then $\varepsilon n - 2\varepsilon' n > 0$. Let i_0 be a place such that we have agreement $\varepsilon n - 2\varepsilon' n$ before i_0 and agreement $2\varepsilon' n$ after i_0; that is,

$$\sum_{i=1}^{i_0} \max(d(w_B[i], c_A[i]), d(w_A[i], c_B[i])) = i_0 - \varepsilon n + 2\varepsilon' n.$$

First we claim that if c_1 is small enough then output node with the correct answer is before i_0. This is since there are at least $\varepsilon n - 2\varepsilon' n - O(\varepsilon' n) = \frac{\varepsilon' n}{c_1} - O(\varepsilon' n)$ good blocks before i_0. Thus by taking c_1 small enough, we will get at least T good blocks before i_0.

All that is left to prove is 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 between i_1 and i_0 is at least $\varepsilon' n$. Note that since agreement between i_0 and n is at least $2\varepsilon' n$ it is enough to prove that agreement from i_1 to n is at most $\varepsilon' n$. The proof to this fact is similar to the proof of Claim 36. 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 δ_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 $i_2 \neq 1$ 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

$$\begin{gathered} d(w_A[i_3,\ldots,n], c_B[i_3,\ldots,n]) = \\ d(w_A[i_3,\ldots,i_2-1], c_B[i_3,\ldots,i_2-1]) + d(w_A[i_2,\ldots,n], c_B[i_2,\ldots,n]) > \\ (1-\varepsilon')(i_2-i_3+n-i_2+1) = (1-\varepsilon')(n-i_3+1) \end{gathered}$$

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 the agreement between i_0 which is after output node and i_1 point in PrefixList is at least $\varepsilon' n$. In other words $agr(w_A[i_0,\ldots,i_1], c_B[i_0,\ldots,i_1]) \geq \varepsilon' n$ 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 the link to grandfather may be of size $\log n$. We are going to prove that the sum of all links is at most $O(n)$. Thus on average we are going to send links 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 π 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 π 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 3. Since Pointer Jumping Problem is complete for communication complexity any protocol π 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 π' 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 Σ_{in} to be large enough to contain $Ent = \frac{L}{\varepsilon'}$ transcript entries with link of size $MLSize = \frac{L^2}{\varepsilon'^2} = O(\frac{1}{\varepsilon^4})$. Note that the size of Σ_{in} is a constant depending on ε. By Theorem 22 there exist $C : \Sigma_{in}^n \to \Sigma_{out}^n$ 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 the size larger than $MLSize$ we will say that we send a long link. 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 the cost of not sending other edges we wanted to send in this and maybe next several rounds (up to $\log n$ rounds). We are going to modify our protocol a little bit 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 are 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 long links since most of the time we will respond to an adversary communication. We are going to make 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 the case that grandparent of e was sent at the 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 a link to his grandfather at node v. So that the sum of links will be not too large we need a cousin trick[5].

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 we initiate for every $v \in PL_A$ value $Edge(v)$ by $\emptyset$.

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) If for all predecessors u of v it holds that $Edge(u) = \emptyset$ then
   If $E(B_j)$ contains exactly one son of the root set this son to be e_{father} go to $3.(e).i$ [6]
   (d) Let $u \in PL_A$ be the last predecessor (i.e. closest to v) of v with $Edge(u) \neq \emptyset$ Let $e_{\text{grandparent}} = Edge(u) \in \mathcal{T}$.
   (e) If $E(B_j)$ contains exactly one son of $e_{\text{grandparent}}$(let us call it e_{father}) then:
      i. Let $e_{\text{son}} \in X$ to be son of e_{father}.
      ii. $Add(A, e_{son})$.
      iii. Set $Edge(v) = e_{\text{son}}$.
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 $n = \frac{T}{\varepsilon'} = O(\frac{T}{\varepsilon})$ 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 $agr(v \to v_s, w_A) > T$ then output v.

[5] Recall that we send a link to the closest either grandfather or cousin

[6] For Bob this should be replaced with: Set e_{son} to be the edge at level 1 go to 3.e.ii

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 the large alphabet with small adaptations. All the The bound on the list size is exactly the same. If some node v from PL_A or PL_B, $Edge(v) \neq \emptyset$ we will say that we speak at node v. We will have here several differences. 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. In other words if c_A, c_B is what Alice and Bob have send than $P_A = PL_A \cap c_B$ and $P_B = PL_B \cap c_A$. First, we say that we advance at block i if Alice or Bob spoke at $PL_A[i]$ or $PL_B[i]$.

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

Then we advance at block i if 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 $Edge(P_A[i])$ for $i = 1, \ldots, i$ and the same Y_i with P_B. 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 on the list.

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

Proof. The proof follows trivially from the definitions of X_i, Y_i and the fact that in our protocol Alice can send a new edge at node v only after receiving a response from the last edge she sent. For completeness, we include a formal proof.

Let us prove this by induction. At round 0 the statement trivially follows. If at round i $Edge(P_A[i]) = Edge(P_B[i]) = \emptyset$ then $X_i = X_{i-1}, Y_i = Y_{i-1}$ and again it trivially follows. If at block i we have $Edge(P_A[i])$ (or the same proof for Bob $P_B[i]$), let $j < i$ be maximal such that $Edge(P_A[j]) \neq \emptyset$ let $e_j = Edge(P_A[j]) \in \mathcal{T}$. 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_{father} next node on P and e_{son} is the node after it. $\square$

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., $B[1, \ldots, i-1] \in List_{i-1}(w_A, C, \varepsilon)$ and $A[1, \ldots, i-1] \in List_{i-1}(w_B, C, \varepsilon)$.

2. The list size at block i was less than Ent.

3. We did not send the long link at block i, and we did not send the 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 the case with large alphabet we know that there are at most $O(\varepsilon' n)$ blocks which do not satisfy the 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 with $e = Edge(v) \neq \emptyset$, recall from Section 6 we encode every edge by (r_e, b_e, s_e) where r_i is a pointer. We set $link(v) = r_e$ be a pointer of the 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 the length of the link by $j - k$. We denote it by $|link(v)|$.*

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

Claim 45. $link(v) \leq Ent \cdot (|link(v)| + 1)$.

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*

$$\sum_{u \in PL_A(v) \setminus \{v\}} |link(u)| \leq |PL_A(v)|.$$

Proof. We are going to prove the lemma by induction. For leaves, it is trivially true. Denote by

$$S(v) = \sum_{u \in PL_A(v) \setminus \{v\}} |link(u)|.$$

Next assume that at node v we have sent an edge e. Let us consider all nodes in $PL_A(v)$: $u_1, u_2, \dots, u_k \in PL_A(v)$ 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

$$S(v) = \sum_{i=1}^{k} |link(u_i)| + \sum_{i=1}^{k} S(u_i) = \sum_{i=1}^{k} |link(u_i)| + \sum_{i=1}^{k} |PL_A(u_i)|.$$

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

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

Therefore

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

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

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

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 $l > MLSize$ is sent is at most

$$\frac{2\log l + 2}{\log MLSize} < \frac{l}{MLSize} + 3$$

To last inequality note that it holds for $l = MLSize$ and note that derivative with respect to l of LHS is less than derivative of RHS for $l > MLSize$.

From Claims 45, 46 it follows that:

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

And the same for PL_B. Since long link is a link of size at least $MLSize$ from Markov inequality it follows that the number of long links is bounded by: $LL = \frac{2L^2}{\varepsilon' MLSize} n = O(\varepsilon' n)$.

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

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

□

We will use the following theorem for unique decoding:

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

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

Then there exists output node on both paths P_A, P_B[7] *and this node is located before level i_0.*

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*

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

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_1 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 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 To prove the theorem, we will execute the protocol above. Note that number of rounds is $O_\varepsilon(T)$. Also note that Σ_{in} is of constant size (depending on ε, but not on T), thus Σ_{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 α, β with $\alpha + \beta \geq 1$ there exists protocol such that any list decoding protocol resilient to (α, β) noise with linear communication complexity must have exponential list size.

We will show this by showing that if one can perform some task with (α, β) 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 (α, β) 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.*

[7] Recall that this are the "correct" paths

Proof. If Alice and Bob speak alternatively, then this theorem follows very easily since the adversary can erase all the communication of Alice first α 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. However, if Alice and Bob do not speak alternatively, then it may happen that Alice speaks more at the beginning, and Bob speaks more at the end, so this way at the first α fraction of communication Alice speaks more than α 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_{max} be a point at which this function is maximal. If $n_A - A(i_{max}) \leq \alpha n_A$ and $B(i_{max}) \leq \beta n_B$ then adversary can destroy communication of Bob before i_{max} and communication of Alice after i_{max} and thus we will get protocol equivalent to a two round protocol where Alice send all her information in rounds $1, \ldots, i_{max}$ and Bob sends all his information in rounds $i_{max}, \ldots, n$.

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

$$\frac{A(i_{max})}{n_A} - \frac{B(i_{max})}{n_B} \geq \frac{A(i_0)}{n_A} - \frac{B(i_0)}{n_B}\ .$$

Rearranging this inequality we will get

$$\frac{A(i_{max}) - A(i_0)}{n_A} \geq \frac{B(i_{max}) - B(i_0)}{n_B} = \beta$$

Thus Alice communication in the intervals everywhere except in $[i_0, i_{max}]$ is at most $1-\beta \leq \alpha$.

Now let us assume that $n_A - A(i_{max}) > \alpha n_A$. Let i_1 be such that $A(i_1) - A(i_{max}) = \alpha n_A$. In this case we will destroy communication of Alice in the interval $[i_{max}, i_1]$ and the communication of Bob everywhere else. In this case from definition of i_{max} we will get

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

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

Now we are ready to prove Theorem 2.

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

Proof. If protocol π is resilient to (α, β) 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. $\square$

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 π and assume that adversary can corrupt up to an α fraction of messages sent by Alice and β fraction of messages 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 is quite surprising.

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

In this section we are going to show how to perform two sided unique decoding when $\alpha+2\beta < 1$ and $2\alpha + \beta < 1$. However, as we will see from the next sections, this is not optimal. The algorithm described here gives us 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 the list-decoding protocol from the previous section that is resilient to $(1-\varepsilon')$-noise, but instead of outputting a list at the end, we will output the closest answer that is we will find $c_{output} \in C$ such that

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

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

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

Lemma 52. *Let C be a $(\varepsilon', \frac{1}{\varepsilon'} + 1)$ list-decodable tree code. For every $x, y \in \Sigma_{in}^n$, let s be the 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 $n - \varepsilon' n$ locations and to be $C(y)$ at the last $\varepsilon' n$ locations. Then $\delta_s(w, C(y)) \leq 1 - \varepsilon'$ and, also,

$$\delta_s(w[1, \dots, n - \varepsilon' n], C(x)[1, \dots, n - \varepsilon' n]) = 0 \leq 1 - \varepsilon'.$$

By the 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 at the starting level s, $C(x)$ and $C(y)$ represent different branches in the PrefixList. Agreement between w and $C(y)$ at the starting level s is at least the agreement between $C(x)_{[s,\dots,n]}$ and $C(y)[s, \dots, n]$. Thus

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

Consequently, $agr(C(y)[s, \dots, n], C(x)[s, \dots, n]) \leq 2\varepsilon' n$. $\square$

Let us assume by contradiction that protocol outputs a wrong answer. Let us assume w.l.g. that Alice outputs the wrong answer. Let $c_B \in \Sigma_{out}^n$ be a codeword that was sent by Bob. Let us assume that w_A was the codeword received by Alice and assume that c_{output} is the closest codeword to w_A. There are two important points on c_B: one is a "split" point s to be the first place where $c_{output}[s] \neq c_B[s]$ and the other is e_{end} output communication point (we will show soon that one exists on the "correct" path c_B). Observe that if the output node is located before the split point s, then we output the correct answer. Thus, by contradiction, assumption $s < e_{end}$. 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_{end}, and, thus, also on s.

Let us define $B_1 = d(c_B[1,\dots,s], w_A[1,\dots,s])$ and $B_2 = d(c_B[s+1,\dots,n], w_A[s+1,\dots,n])$. Note that $B_1 + B_2 \leq \beta n$. From the Lemma 52, we know that $d(c_{output}, c_B) \geq n - s - O(\varepsilon' n)$; thus, since w_A is closer to c_{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, we get

$$s \geq n - 2B_2 - O(\varepsilon' n). \tag{2}$$

On other hand, from Corollary 48 it follows that

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

Thus we have that

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

Rewriting this gives

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

Thus we have that $2\beta + \alpha > 1 - O(\varepsilon')$; consequently, by taking c small enough, we will see that $2\beta + \alpha > 1 - \varepsilon$, a contradiction to the definition of ε.

12 Repetition Power of the Protocol

To construct a protocol that can handle the optimal fraction of errors, we would like to give a different price to the cost of corruption of any symbol in the protocol. One way to do this is by using the 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, we will do the same: we will just send the symbol of the ith round $a(i)$ times, where $a : [n] \to \mathbb{N}$; 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 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 that Alice sends at her ith round and $b(i)$ is the number of symbols Bob sends at his ith round.

Let π be a protocol with $2n$ alternating rounds of Alice and Bob over an alphabet Σ_{out}. Let us 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 π, Alice or Bob receives symbol from $D(\Sigma_{out})$ and outputs symbol in Σ_{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 $\pi = (p_A, p_B)$ is the following protocol: in the protocol $\pi^{a,b}$, at round $2i-1$ Alice will receive $b(i-1)$ symbols $w_1, \dots, w_{b(i-1)} \in D(\Sigma_{out})$, calculate the average $w_A = \frac{1}{b(i-1)} \sum_{i=1}^{b(i-1)} w_i$. Send $a(i)$ copies of $\pi_A(w_A)$ to Bob. Bob, at round $2i$, will receive $a(i)$ symbols$w_1, \dots, w_{a(i)} \in D(\Sigma_{out})$, calculate the average $w_B = \frac{1}{a(i)} \sum_{i=1}^{a(i)} w_i$. Send $b(i)$ copies of $\pi_B(w_B)$ to Alice.*

First note the simple connection between communication costs of π and $\pi^{a,b}$.

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

Proof. The proof follows from the definition. $\square$

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

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

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

The same holds for Bob, that is,

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

This lemma essentially shows that by 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 $1 \le i \le n, 1 \le j \le a(i)$. 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 $\sigma_1, \dots, \sigma_k$, 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 Section 11. The only way for adversary to corrupt Alice's output is the following: Pick $i \in [1, n]$. Corrupt at least α_1, β_1 fraction of Alice's and Bob's communication before i, such that $\alpha_1 + \beta_1 > 1 - O(\varepsilon)$. Corrupt $1/2 - O(\varepsilon)$ fraction of the communication sent by Bob at $n - i$ last rounds. This ensures that Alice and Bob essentially make no progress in the first i rounds, and that Alice is confused between two possible Bobs in the last $n - i$ rounds, once they start making progress.

In the scenario when Alice and Bob speak at the same speed it always optimal for the adversary to corrupt Alice's communication before i and half of Bob's communication after i. Now let us change the protocol such that at the beginning Alice speaks more and at the end, Bob speaks more. In this case, the adversary will spend 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 choose functions carefully a, b and our protocol in this section will be $\pi^{a,b}$ where π is the list decodable protocol from earlier in the paper. Consider the following example: let $\alpha = \frac{1}{3}$. Then we divide our communication into three parts where at each part we will have $\frac{n}{3}$ of 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 Bob Alice will send one message each time. In the second part Bob will send two messages on every Alice's message; in the last part, Bob will send his message four times, on every Alice's 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 $O(\varepsilon n)$), communication before i and half of Bob's messages in rounds after i. If we want to corrupt the minimum amount of Bob's communication, 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 for every one Alice's messages. Let us now define for every f:

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

Remark 56. *We should mention that once we have picked s, the function g that minimizes the preceding expression is 1 on a set S, $\mu(S) = s - \alpha$. Here S is a set of all x's with $f(x) \le t$. Since it may happen that for many x, $f(x) = t$ we will take only subset of this* *set, i.e, set S is of the form $\{x < s : f(x) < t\} \cup A_t$, where $A_t \subset \{x : \ x < s \wedge f(x) = t\}$ for some t.*

In other words, if the rate $f(t)$ at which Bob talks at Alice's round t is fixed, s and g correspond to an adversary's strategy for trying to prevent Alice from unique decoding, and $L_1(\alpha, f)$ corresponds to the smallest amount of error on Bob's side it needs to succeed, assuming the amount of error on Alice's side is α. Specifically, s in the definition of L_1 is exactly where Alice and Bob accomplish the list-decoding task: from that point on, the adversary needs to cause errors at a rate of $\geq 1/2$ on Bob's end to prevent Alice from uniquely determining the output. $g(t)$ represents to the fraction of Bob's communication at round tn corrupted by the adversary. Up to point s the adversary needs to corrupt either Alice or Bob at essentially every round. It can corrupt Alice in αn rounds, hence the requirement $\int_0^s g(t)dt \geq s - \alpha$, which ensures that Bob is corrupted in $(s-\alpha)n$ rounds before point s is reached.

We want to define non-explicitly the function that will be the boundary of the region in which we can perform one-sided unique decoding. In the next subsection, we will give an explicit formula for 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 an α error, and we want Alice to output the correct answer.

The following theorem says that we can decode with a 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 (α, β) such that $\beta < L_1(\alpha)$, there exists a protocol that is resilient to (α, β)-noise and that solves the one way Pointer Jumping Problem of depth T. Communication complexity of this protocol is $O_{\alpha,\beta}(T)$.*

Proof. Let 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

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

Let $a(i) = \frac{1}{\varepsilon}$ for every i. Let π 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 corresponding messages on which list decoding protocol π 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 sends at least one and at most (constant) $\frac{R}{\varepsilon}$ messages. Note also that from Lemma 54 it follows that in protocol $\pi^{a,b}$ Alice sends $\frac{n}{\varepsilon}$ messages and Bob sends $\frac{n}{\varepsilon} - t$ messages, where $t < n$. 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_{output} be the closest codeword to w_A, where here we give weight $b(i)$ to symbol i, that is, the $c_{output} \in C$ 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_{output} are different is $s \cdot n$. Let us assume by contradiction that Alice outputs a wrong answer and that output node is located after $s \cdot n$.

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

$$\sum_{i=sn}^{n} (1 - d(c_{output}[i], c_B[i])) \leq O(\varepsilon n)$$

Thus we have:

$$\sum_{i=s \cdot n}^{n} b(i) d(c_{output}[i], c_B[i]) = \sum_{i=s \cdot n}^{n} b(i) - \sum_{i=sn}^{n} b(i)(1 - d(c_{output}[i], c_B[i])) \geq$$
$$\sum_{i=s \cdot n}^{n} b(i) - \frac{R}{\varepsilon} \cdot O(\varepsilon n) = \sum_{i=s \cdot n}^{n} b(i) - O(n)$$

Consequently,

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

On other hand, it follows from Corollary 48 that

$$d(w_B[1, \ldots, s \cdot n], c_A[1, \ldots, s \cdot n]) + d(w_A[1, \ldots, s \cdot n], c_B[1, \ldots, s \cdot n]) \geq s \cdot n - O(\varepsilon n) \ . \quad (3)$$

Note that since $a(i) = \frac{1}{\varepsilon}$ is constant, from Lemma 55 it follows that

$$d(w_B[1, \ldots, s \cdot n], c_A[1, \ldots, s \cdot n]) = \varepsilon d(w'_B[1, \ldots, \frac{s \cdot n}{\varepsilon}], c'_A[1, \ldots, \frac{s \cdot n}{\varepsilon}]). \quad (4)$$

We are assuming that at most α fraction of Alice's communication was corrupted thus

$$d(w'_B[1, \ldots, \frac{s \cdot n}{\varepsilon}], c'_A[1, \ldots, \frac{s \cdot n}{\varepsilon}]) \leq \alpha \frac{n}{\varepsilon} \ . \quad (5)$$

Therefore, from Equations (3), (4), and (5) it follows that,

$$d(w_A[1, \ldots, s \cdot n], c_B[1, \ldots, s \cdot n]) \geq s \cdot n - \alpha n - O(\varepsilon n) \ . \quad (6)$$

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 $g(t) = d(w_A[\lfloor tn \rfloor], c_B[\lfloor tn \rfloor])$. Then we have

$$d(w'_A[1,\ldots,s'\cdot n], c'_B[1,\ldots,s'\cdot n]) \geq \frac{n}{\varepsilon}(\int_0^s g(t)f(t)dt) - n.$$

Consequently,

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

It follows from Equation (6) that

$$\int_0^s g(t) \geq n(s - \alpha - O(\varepsilon)).$$

Therefore, since f is bounded

$$d(w'_A, c'_B) \geq n(L_1(\alpha, f) - O(\varepsilon)) .$$

By taking ε small enough, we will get a contradiction.

□

We also claim that the opposite direction is true:

Theorem 58. *For every α, β with $\beta > L_1(\alpha)$, if a protocol π is resilient to (α, β)-noise, then there exists a protocol π' that runs constant $C = O(\frac{1}{\beta - L_1(\alpha)})$ number of rounds. Such that π' computes the output of Alice and $CC(\pi') = CC(\pi)$.*

Proof. Let us consider the protocol π. 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 εn_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 εn_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 $\int_0^s f(t)g(t)dt + \frac{1}{2}\int_s^1 f(t)dt \leq L_1(\alpha) < \beta$.

Let $t = \lfloor \frac{s-\alpha}{\varepsilon} \rfloor$ Note that from Remark 56 it follows that we can assume w.l.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

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

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 $f(x)$. Let us define $S' = \cup_{k=1}^{t+1}[(i_k-1)\varepsilon, i_k\varepsilon]$ note that $\mu(\bar{S}') > \alpha - \varepsilon$. Value of f on $[(i_{t+1}-1)\varepsilon, i_{t+1}\varepsilon]$ is at most $\frac{1}{\alpha-\varepsilon}$, since else value of f on set $\bar{S}'$ will be grater than $\frac{1}{\alpha-\varepsilon}$ thus $\int_{\bar{S}'} f(t)dt > 1$ in contradiction to fact that $\int_0^1 f(t)dt = 1$. Then for small enough ε 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 adversary has erased Bob's communication on the intervals $[T(i_k-1),\ldots,T(i_k)]$ (from the preceding representation of the set S), and Alice's communication in all the rest of the intervals before s. Note that since $\mu(S') \geq s - \alpha$, we have erased at most α fraction of the Alice's communication and at most $\int_0^s f(t)1_{S'}$ fraction of Bob's communication. Note that after time s Bob communicates $\int_s^1 f(t)$ communication thus after time s adversary 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 $\frac{2}{\varepsilon}$ rounds protocol in which Alice solves the task. Assume, by contradiction, that there exist two inputs to Bob that are consistent with what Alice has seen by time s; that is, there exist 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 adversary can corrupt $\frac{1}{2}$ of the communication after round s, adversary will construct a codeword that in $\frac{1}{2}$ rounds takes value sent by $\pi_B(x,y_1)$ and in $\frac{1}{2}$ rounds takes takes value sent by $\pi_B(x,y_2)$. Hence, we got codeword w_A which Alice could receive for both Bob's inputs y_1, y_2. Therefore, since we assume that Alice's output on x, y_1 is different from the output on input x, y_2, on one of these inputs Alice output the wrong answer.

□

14 Calculating the Functions L_1

In this section for every α we want to calculate $\sup_f L_1(\alpha, f)$. With very little more effort, we can calculate the more general function that we will need in the next section. For $\alpha < \gamma \leq 1$ let us define

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

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

Remark. During the first reading of the proofs in this and subsequent sections, *the reader should assume that α is of the form $\alpha = \frac{1}{m}$ for some integer m and that γ is an integer* *multiple of* α. This simplifies the expressions and the proofs significantly. For example, because the fractional part $\left\{\frac{1}{\alpha}\right\} = 0$. Once the proofs for this important special case are sorted out, the extension to the rest of the values is just a matter of figuring out the exact dependence on α within each interval of the form $\alpha \in \left[\frac{1}{m+1}, \frac{1}{m}\right]$, which is a bit tedious and complicates the presentation.

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 $l = \lfloor \frac{\gamma}{\alpha} \rfloor$. 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 $L_1(\alpha, \gamma, C) = L_1(\alpha, f)$.

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 it holds that

$$\frac{1}{2}(1 - H(s)) + H(s - \alpha) \geq L_1(\alpha, h) \geq L_1(\alpha, f) = \frac{1}{2}(1 - F(s)) + F(s - \alpha) = T(\alpha). \quad (7)$$

Now let us give an intuition for the proof of the theorem. First let us consider the case when $\frac{\gamma}{\alpha}$ is an integer (equal to l) — as noted earlier, this is an insightful special case to keep in mind. Now we will show that:

1. $H((l-1)\alpha) \geq F((l-1)\alpha)$
2. $H((l-1)\alpha) \leq F((l-1)\alpha)$

From these two inequalities it follows that $H((l-1)\alpha) = F((l-1)\alpha)$ therefore if we substitute this into Equation 7 with $s = \gamma$ (recall that by definition $H(\gamma) \geq F(\gamma) = C$) we will get that all inequalities there are in fact equalities and therefore $L_1(\alpha, h) = L_1(\alpha, f)$.

Now let us explain how to prove two inequalities above. To show the first inequality all what we need is to set $s = \gamma$ in Equation 7 (again, recall that $H(\gamma) \geq F(\gamma) = C$). We are going to show the second inequality by induction. We will show that $H(i\alpha) \geq F(i\alpha)$ for $i = 1, \ldots l - 1$. First, let us set $s = \alpha$ in Equation 7. Since $H(s - \alpha) = F(s - \alpha) = 0$ we will get that $H(\alpha) \leq F(\alpha)$. The rest of the inductive argument is given in the claim below.

Note that when $\frac{\gamma}{\alpha}$ was an integer we did not use third property of the theorem for proof. When it is not integer we need to be more careful. We will use the third property of the theorem together with a carefully chosen function g(from the definition of L_1) to upper bound the value of $L_1(\alpha, h)$.

Claim 60. *For every $i = 1, \dots, l$ it holds that $H(i\alpha) - H((i-1)\alpha) \le F(i\alpha) - F((i-1)\alpha)$ in particular $H(i\alpha) \le F(i\alpha)$.*

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

For i we see from Equation (7), for $s = i\alpha$ and the induction assumption on $i-1$,

$$\frac{1}{2}(1 - H(i\alpha)) + H((i-1)\alpha) \ge \frac{1}{2}(1 - F(i\alpha)) + F((i-1)\alpha).$$

Rewriting this inequality gives:

$$H(i\alpha) - H((i-1)\alpha) \le F(i\alpha) - F((i-1)\alpha) - (F((i-1)\alpha) - H((i-1)\alpha)).$$

Since by induction we assume that $F((i-1)\alpha) - H((i-1)\alpha) \ge 0$ the claim follows. □

On other hand by using Equation (7) at $s = \gamma$, it follows from the second property of the theorem that

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

Using the third property of the theorem we see that $F(\gamma - \alpha) = F((l-1)\alpha) + \{\frac{\gamma}{\alpha}\}(F(l\alpha) - F((l-1)\alpha))$, thus

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

Now, let us take in the definition, of L_1 value of s to be γ and $g(x)$ to be 1 on $[0, (l-1)\alpha]$ and $\left\{\frac{\gamma}{\alpha}\right\}$ on $[(l-1)\alpha, l\alpha]$. 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 the equation

$$\frac{1}{2}(1 - H(\gamma)) + H((l-1)\alpha) + \left\{\frac{\gamma}{\alpha}\right\}(H(l\alpha) - H((l-1)\alpha)) \ge T(\alpha).$$

Since $H(\gamma) \ge C$, it follows that

$$H((l-1)\alpha) + \left\{\frac{\gamma}{\alpha}\right\}(H(l\alpha) - H((l-1)\alpha)) \ge T(\alpha) - \frac{1-C}{2}\ .$$

From Equation 8,

$$H((l-1)\alpha) + \left\{\frac{\gamma}{\alpha}\right\}(H(l\alpha) - H((l-1)\alpha)) \ge F((l-1)\alpha) + \left\{\frac{\gamma}{\alpha}\right\}(F(l\alpha) - F((l-1)\alpha))\ .$$

Therefore, from Claim 60 and the preceding equation, it follows that an equality should hold. Consequently,

$$\begin{aligned} L_1(\alpha, h) &\le \tfrac{1}{2}(1 - H(\gamma)) + H((l-1)\alpha) + \left\{\tfrac{\gamma}{\alpha}\right\}(H(l\alpha) - H((l-1)\alpha)) \\ &= \tfrac{1}{2}(1 - H(\gamma)) - \tfrac{1-C}{2} + T(\alpha) \le T(\alpha) = L_1(\alpha, f). \end{aligned}$$

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_α 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_α 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. Note that from Remark 56 and the fact that f_α is increasing function $L_1(\alpha, f_\alpha)$ is achieved for some s and $g = 1_{[0,s-\alpha]}$. Thus $L_1(\alpha, f_\alpha) = \frac{1}{2}(1 - F_\alpha(s)) + F_\alpha(s-\alpha)$. Therefore, 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 $f_\alpha(s-\alpha) = 2f(s)$. □

Thus from Theorem 59 it holds that: $L_1(\alpha) = L_1(\alpha, f_\alpha) = \frac{1}{2}(1 - F_\alpha(s)) + F_\alpha(s-\alpha)$ for every s. We can now easily calculate $L_1(\alpha)$.

Lemma 62.

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

Proof. From Property 4 it follows that $L_1(\alpha) = \frac{1}{2}(1 - F_\alpha(s)) + F_\alpha(s-\alpha)$ for every s. We can pick $s = \alpha$. In this case $T(s) = \frac{1}{2}(1 - F_\alpha(\alpha)) = \frac{1}{2}\left(1 - \frac{1}{(1+\{\frac{1}{\alpha}\})2^{\lfloor\frac{1}{\alpha}\rfloor}-1}\right)$. □

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 α it is $\frac{1}{2}\left(1 - \frac{1}{(1+\{\frac{1}{\alpha}\})2^{\lfloor\frac{1}{\alpha}\rfloor}-1}\right)$.*

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

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

Let us define

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

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

Theorem 64. *For every (α, β) such that $\beta < L_2(\alpha)$ there exists a protocol π that is resilient to (α, β)-noise and that solves Pointer Jumping Problem of depth T. Communication complexity of this protocol is $O_{\alpha,\beta}(T)$.*

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 $a(i) = \frac{1}{\varepsilon}$ for every i. Let π be the list-decoding protocol from the previous section with code C that 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 the list decoding protocol π 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 the expression

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

Since $a(i)$ is constant this is the same as outputting the closest codeword to w_B. Alice's output is correct by Theorem 57. So, now we need to prove that Bob's output is correct. Let c_{output} the output of the Bob. Let $s \cdot n$ be a first place where c_{output} is different from c_A; then we know from Lemma 52 that

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

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 $g(t) = d(w_A[\lfloor tn \rfloor], c_B[\lfloor tn \rfloor])$. By taking ε small enough we have

$$\int_0^s g(t) f(t) dt < L_2(\alpha, f) \ .$$

Thus, by definition of L_2 we see 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 ε

$$d(w_B[1,\ldots,s\cdot n], c_A[1,\ldots,s\cdot n]) > \frac{s\cdot n + 2\alpha n - n}{2} \ .$$

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 α fraction of the communication is corrupted □

Theorem 65. *For every α, β with $\beta > L_2(\alpha)$, if protocol π resilient to (α, β) 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 π. Let as assume that Alice sent n_A symbols and Bob sent n_B symbols during the protocol. Let $T(1)$ be the time when Alice sent her εn_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 εn_A of her messages. Let $n_i = T(i) - i n_A \varepsilon$ 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 $\frac{n_i - n_{i-1}}{\varepsilon n_B}$ 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.

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 π. 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 $t = \lfloor \frac{s-2\alpha'+1}{2\varepsilon} \rfloor$ We can assume w.l.g. that $g = 1_{S'}$ where S' is union of t of intervals of form $[(i-1)\varepsilon, i\varepsilon]$ plus part of such interval. Let us define

$$S = \cup_{k=1}^{t} [(i_k - 1)\varepsilon, i_k \varepsilon] \ .$$

Then it holds that

$$\int_0^s f(t) 1_S dt < \beta$$

Let us now assume that adversary has erased Bob's communication on the intervals $[T(i_k - 1), T(i_k)]$ (from the definition of S) and the Alice's communication 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 $s - (\frac{s-2\alpha'+1}{2} - \varepsilon) \leq \frac{s+2\alpha-1}{2}$ fraction of Alice's communication before s and at most β fraction of Bob's communication. Note that after time s, Alice communicates $1-s$ fraction of her communication. Thus after time s adversary can corrupt at least $\alpha - \frac{s+2\alpha-1}{2} = \frac{1-s}{2}$. In other words, the adversary can corrupt $\frac{1}{2}$ of Alice's communication after time s.

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

□

16 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 it will be $(1 + \{\frac{1}{\alpha}\}) 2^{\lfloor \frac{1}{\alpha} \rfloor - 2}$ in the interval $[1-2\alpha, 1]$. Then we will define f_α to be the normalisation of $\tilde{f}_\alpha$. Let us also define $F_\alpha(s) = \int_0^s f_\alpha(t)dt$.

As noted earlier, the proofs below are significantly easier to follow in the important special case when $\frac{1}{\alpha} = m$ is an integer. In that case $\lfloor \frac{1}{\alpha} \rfloor = m$, and $\{ \frac{1}{\alpha} \} = 0$. The hardest part of this subsection is the following lemma.

Lemma 66.

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

In particular, $L_1(1/k) = L_2(1/k) = \frac{1}{2}(1 - \frac{1}{2^{k-1}-1})$.

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 $g = 1_{[0,s-\alpha]}$. 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 $T'(s) = f(s - \alpha) - \frac{1}{2}f_\alpha(s) \geq 0$. 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 $[0, \dots, 1 - \alpha(2 + \{\frac{1}{\alpha}\})]$ 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

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

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 $[\frac{s - 2\alpha(1 + \{\frac{1}{\alpha}\}) + 1}{2}, s]$. Thus

$$T_2(s) = F_\alpha(1 - \alpha(2 + \{\frac{1}{\alpha}\})) + F_\alpha(s) - F_\alpha(\frac{s - 2\alpha(1 + \{\frac{1}{\alpha}\}) + 1}{2}) \ .$$

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

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

□

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_α 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 $f_\alpha(s) = 2f_\alpha(s - \alpha)$. □

17 Proof of Theorems 4 and 5

Since from Theorems 64 and 65 it follows that L_2 is the boundary of (α, β), L_2 must be symmetric in α, β; that is, 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 $\mathcal{R}_u$ 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 π there exists another protocol π', with $CC(\pi') = O_{\alpha,\beta}(CC(\pi))$ which is resilient (α, β) 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. □

Theorem 69. *For every $(\alpha, \beta) \notin \overline{\mathcal{R}}_U$ and for every T the following holds. Let π' be a protocol resilient to (α, β) adversarial noise that solves the Pointer Jumping Problem of depth T. Then $CC(\pi') = 2^{\Omega_{\alpha,\beta}(T)}$.*

Proof. From Theorem 65 it follows that there exists constant $O_{\alpha,\beta}(1)$ round protocol π'' with the same communication complexity as the protocol that solves the one-way Pointer Jumping Problem. Then from Corollary 9 it follows that communication complexity of π'' 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 essential for unique decoding of interactive communication. Our encodings are a constant rate, that is, they incur only constant blow-up in communication. However, we have not tried to optimize this constant, leaving such optimization as an open problem. We highlight this and additional outstanding open problems:

Open Problem 1. *The big problem is to understand what the best possible rate is for interactive communication. We want to note that this question is wide-open even in the random-noise scenario. Recently, there was some progress on this question in random noise scenario with a noise rate approaching 0 by Kol and Raz [KR13] Haeupler [Hae14], and by Gelles at al [GHK+16].*

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 of two in the error rates that one can handle. However, it is not clear at all that these error rates are the best possible. Establishing the region of (α, β) for which unique decoding is possible over a binary alphabet is an open problem.*

A good start for attacking Problem 2 would be to understand the limits of list-decoding over binary (and other small) alphabets.

Our tight-negative results (and the matching protocols) all work in the so-called *non-adaptive* model, where whether Alice or Bob speaks at a given round of the protocol is determined *a-priori* and does not depend of the messages, or the adversary's behavior. In the unique-decoding model it has been shown that one can define a meaningful model of adaptive protocols [GHS13], and achieve an increased error-tolerance in such model. Note that if one allows "who speaks" to depend on the adversary's behavior, then one needs to define the model's behavior if both Alice and Bob (or neither) decide to speak at the same round. A further complication when discussing resistance to (α, β) errors is that in adaptive protocols, the total number of messages Alice sends depends on the transcript, and thus, unlike the total length of the protocol, is not known *a-priori*. Thus the adversary may exceed its budget with respect to Alice's messages if Alice chooses to stop communicating when she realizes more than an α-fraction of her messages so far had been corrupted. Therefore, additional modeling decisions will need to be made to address the adaptive setting.

Open Problem 3. *Define the "right" model of adaptive* (α, β)*-errors, and find the region of errors which can be corrected in the adaptive model.*

We believe that our lower bounds hold if the total amount of communication from each side needs to be fixed in advance (even if the order is allowed to be adaptive).

Finally, the construction of explicit list-tree codes is a fascinating combinatorial problem:

Open Problem 4. *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 constructing regular tree codes — a problem that appears to be difficult [MS14] — 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

[AGS13] Shweta Agrawal, Ran Gelles, and Amit Sahai. Adaptive protocols for interactive communication. *arXiv preprint arXiv:1312.4182*, 2013.

[BK12] Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 160–166. IEEE, 2012.

[BN13] Zvika Brakerski and Moni Naor. Fast algorithms for interactive coding. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 20, page 14, 2013.

[BR11] Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 159–166. ACM, 2011.

[Bra12] Mark Braverman. Towards deterministic tree code constructions. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 161–167. ACM, 2012.

[Eli57] Peter Elias. List decoding for noisy channels. 1957.

[FGOS12] Mattew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J Schulman. Optimal coding for streaming authentication and interactive communication. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 19, page 104, 2012.

[GH13] Mohsen Ghaffari and Bernhard Haeupler. Optimal error rates for interactive coding II: Efficiency and list decoding. *arXiv preprint arXiv:1312.1763*, 2013.

[GHK$^+$16] Ran Gelles, Bernhard Haeupler, Gillat Kol, Noga Ron-Zewi, and Avi Wigderson. Towards optimal deterministic coding for interactive communication. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1922–1936, 2016.

[GHS13] Mohsen Ghaffari, Bernhard Haeupler, and Madhu Sudan. Optimal error rates for interactive coding I: Adaptivity and other settings. *arXiv preprint arXiv:1312.1764*, 2013.

[GMS11] Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In Rafail Ostrovsky, editor, *FOCS*, pages 768–777. IEEE, 2011.

[Gur04] Venkatesan Guruswami. *List decoding of error-correcting codes*. Springer, 2004.

[Hae14] Bernhard Haeupler. Interactive channel capacity revisited. *CoRR*, abs/1408.1467, 2014.

[KR13] Gillat Kol and Ran Raz. Interactive channel capacity. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 715–724. ACM, 2013.

[MS14] Cristopher Moore and Leonard J Schulman. Tree codes and a conjecture on exponential sums. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 145–154. ACM, 2014.

[NW93] Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, February 1993.

[Sch96] Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

[Woz58] John M Wozencraft. List decoding. *Quarterly Progress Report*, 48:90–95, 1958.