2014 IEEE Annual Symposium on Foundations of Computer Science

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

Mark Braveman
Department of Computer Science,
Princeton University.
Princeton NJ
mbraverm@cs.princeton.edu

Klim Efremenko
Department of Computer Science,
University of Chicago.
Chicago, IL
klimefrem@gmail.com

***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 in order to fully characterize the region $\mathcal{R}_U$ 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.

***Keywords*-Interactive Communication; List Decodable Codes; Tree Codes;**

I. 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 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 noninteractive version of the problem is the well-studied problem of encoding a message $M \in \Sigma^N$ with an error-correcting code $C : \Sigma^N \to \Sigma_2^{N'}$ 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 $d_H(C(M_1), C(M_2)) > 2\eta N'$ 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 [1] and Wozencraft [2] 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 [3].

In the interactive setting, it is not at all obvious

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.

Research supported by a Simons Fellowship in Theoretical Computer Science. Part of this work was done while KE was a member of the Institute for Advanced Studies and funded by NSF CCF-0832797 and DMS-0835373 (klimefrem@gmail.com).

that good error correction is possible against adversarial substitution errors of *any* rate. Note that any attempt to apply standard error-correcting codes round by round are bound to fail, since all the adversary has to do to derail the execution of the protocol is to completely corrupt a single round. Therefore, a subconstant error rate of $1/r$ suffices to foil an r-round protocol protected with a round-by-round code. In a striking work, Schulman [4] 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 are depends on the first place where two messages are different.

Interest in interactive error correction has been renewed recently, with Braverman and Rao [5] showing that the error rate that can be tolerated can be improved to $\eta < 1/4$. The constructions of both [4] and [5] are not computationally efficient. A series of recent works made significant progress toward making interactive error correction computationally efficient [6], [7], [8], [9], [10], in some cases by restricting the error parameter or augmenting or restricting the model. In particular, the work of Brakerski and Kalai [8] shows how to implement interactive error correction efficiently for $\eta < 1/16$ with a beautiful scheme that uses private (nonshared) randomness.

In this paper we will consider only *robust protocols*: protocols in which, at all times, the person whose turn it is to speak is known to both Alice and Bob, regardless of the number of errors introduced into the communication so far (even if this amount exceeds the adversary's budget). It is not hard to see [5] that robustness is equivalent to the property that the identity of the speaker at round i at any execution of the protocol depends only on i. Nonrobust protocols have recently received some attention. In particular, very recently [11] and [12] considered nonrobust models and concluded that, indeed nonadaptive protocols can withstand a higher error rate. It would be interesting to combine these results, particularly those from [12], with ours to calculate the tight error-rate region for the nonrobust case.

In this work we investigate the limits of error rates that can be corrected in the interactive setting. Further, we develop the notion of *interactive list decoding*, which is the list analogue of interactive error correction. Its definition is quite straightforward: after the execution of a protocol, each party will output a constant-size list of possible original conversations. If the fraction of errors has not exceeded 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 are able to give a precise characterization 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 [5], 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 $\beta \geq 1/2$. 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 [5].

Recent Related Works: We would like to mention very recent related works of Ghaffari, Haeupler, and Sudan [12], [13], which were developed independently of the present paper. Although these works are closely related to ours in subject matter, the results in these works are almost completely orthogonal. Whereas we study the error model where two parties have different amounts of noise, the main question of [12] 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 [12], [13] is the notion of list-decodable interactive codes, which has been developed independently. While the key definitions are, as expected, similar, there are some differences between the two lines of work. All the codes we consider in our work are constant rate, that is, an n-symbol interactive protocol is always encoded into an $O(n)$-symbol protocol over a constant-size alphabet. The codes in [12] convert n symbols into n^2 symbols, and the improved construction in [13] 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 [13] is. More excitingly, it appears that by combining [13] with the present paper, one obtains efficient, constant-rate schemes with optimal error dependence.[1] The question of finding the *optimal-rate* interactive error correcting schemes remains wide open.

Main Results

list-decoding.: Our first set of results deals with list-decoding interactive protocols. We say that a protocol can handle (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.

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.

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 $CC(\pi) = \exp(\Omega(T))$.*

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

We can give an even tighter result by considering a slightly generalized error notion in Theorem 1. Let us consider only standard protocols 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 $\max(\alpha, \beta) \leq \eta \leq \alpha + \beta$. We show that we can handle η-symmetric noise up to $1 - \varepsilon$, matching the one-way list-decoding bounds.

[1]Haeupler, personal communication.

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 $\eta \leq \alpha + \beta$, 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 correctly recover the original protocol transcript π. 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

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

It is plotted in Figure I. 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 $\pi(x, y)$.*

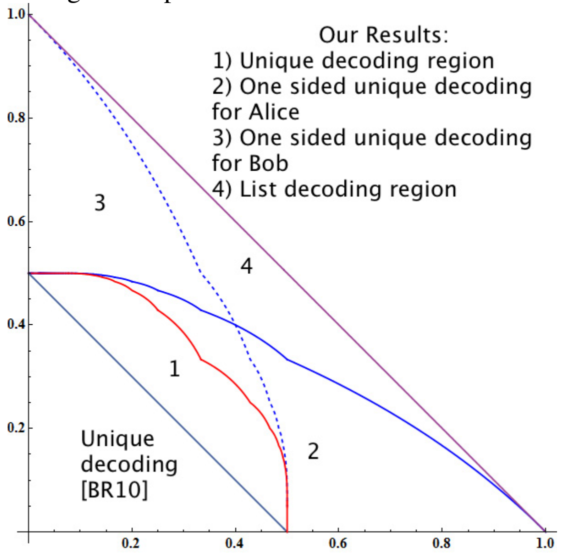
Theorem 4 is stronger than the main upper bound of [5], which shows only how to deal with (α, β) in the subregion $\alpha + \beta < 1/2$. More importantly, Theorem 4 turns out to be essentially tight, as shown in the following theorem. Let $\overline{\mathcal{R}_U}$ be the closure of $\mathcal{R}_U$, that is the union of $\mathcal{R}_U$ and its boundary. The Pointer Jumping Problem can be viewed as the generic communication complexity problem

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 $CC(\pi') = 2^{\Omega_{\alpha,\beta}(T)}$.*

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

Remark 6. *The lower-bound statement of Theorem 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 (noninteractive) good code to send Bob her messages on* all *potential protocol transcripts, and Bob can do the same. This guarantees correct execution, but causes an exponential stretch in communication.*

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



Acknowledgment

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

II. Main Techniques and Discussion

A. List-Decoding

We start with a discussion of the proof of the positive results on interactive list decoding. First we note that any protocol π that sends T bits could be reduced to a Pointer Jumping Problem (PJP) of depth $2T$. PJP is an interactive problem, where Alice receives one edge per node at the odd levels of a binary tree of depth $2T$ and Bob receives one edge per node from the even-level nodes. The task of Alice and Bob is to find a unique path from the root to the leaf using their edges. The overall strategy is similar to that of other recent works: make progress as long as the error is not too high. From the viewpoint of, say, Bob, this means the following: at each step, Bob will try to decode, from Alice's messages he has received so far, what her (noiseless) messages have been, and sends a response that makes progress on the overall protocol. Thus we make progress as long as Bob decodes Alice's messages correctly. It turns out that if Alice and Bob encode their messages using tree codes, this strategy works as long as $\alpha + \beta < 1/2 - \varepsilon$ [5]: in other words, in sufficiently many rounds, Alice and Bob will correctly reconstruct each other's transmissions so far.

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

In order for Alice and Bob to be able to perform list-decoding in interactive settings, we define the analogue of tree codes: a *list-tree code*. The actual definition is somewhat technical, but informally, a list tree code is a prefix code (i.e., it encodes symbols online) such that for any received word w, for almost all rounds, there are at most 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 [6], which can informally be viewed as list-tree codes with list size $\ell = 1 + \varepsilon$.

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

B. Unique Decoding

Next we turn our attention to the unique-decoding regime. It appears that understanding (and being able to carry out) interactive list decoding is needed to achieve optimal unique decoding. To illustrate this point, consider an attempt to break the $(\frac{1}{4}, \frac{1}{4})$-barrier from previous (nonlist decoding) works. To be specific, let $(\alpha, \beta) = (\frac{1}{4}, \frac{1}{4})$. The adversary can corrupt half of Alice's messages in the first half of the protocol, and half of Bob's messages in the second half of the protocol. It is easy to see that in the first half of the execution, unique decoding by Bob is not possible, since there is $\frac{1}{2}$-error on Alice's messages. Therefore, if the parties wait until they can decode uniquely, they will not make any progress in the first half of the protocol. Similarly, they will also not make progress in the second half. On the other hand, with list-decoding, in this scenario we can achieve that by the end of the first part of the protocol Alice has decoded the transcript 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 main list-decodable scheme, Alice and Bob speak at the same rate throughout the protocol (i.e, by the time Alice communicates a p-fraction of her messages, Bob communicates a p-fraction of his messages). It turns out that for some values of (α, β) outside of $\mathcal{R}_2$, we can still achieve unique decoding by having Alice and Bob alter the relative rates at which they speak throughout the execution of the scheme. Note that our scheme is still robust; therefore, these rates will be predetermined by (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 complicated way in which altering relative transmission rates achieves decodability.

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

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

III. List Tree Codes

Definition 7. *A prefix code $C : \Sigma_{in}^n \to \Sigma_{out}^n$ is a code such that $C(x)_i$ depends only on $x[1..i]$. By abuse of notation we will also write for $i < n, x \in \Sigma_{in}^i, C(x) \in \Sigma_{out}^i$.*

Definition 8. *The suffix distance between two strings $x, y \in D(\Sigma)^m$ is defined as*

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

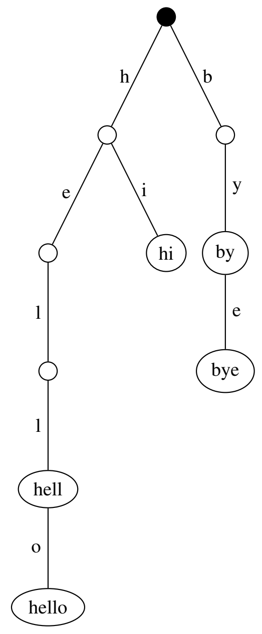


Figure 1. PrefixTree with $List_2 = \{"hi", "by"\}, List_3 = "bye", List_4 = "hell", List_5 = "hello"$

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

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

Definition 9. *For every $w \in D(\Sigma_{out})^n$ the i-th level ε-list of w under C is given by*

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

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

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

We will identify all codewords Σ_{in}^* with a full Σ_{in}-ary tree. Where codewords of length d will be at depth d. We will identify words $x \in \Sigma_{in}^*$ with node in a tree, where $x \in \Sigma_{in}^i$ 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 the Figure 1 we give an example of $PrefixList$. Note that in this example although $|\cup List_i| = 5$. The size of PrefixList is 9.

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

Lemma 10. *Let v be a leaf in the the $PrefixList(w, C, \varepsilon)$ then for every path P ending at v we have $agr(C(P), w(P)) > \varepsilon|P|$*

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

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

Definition 11. *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 first property is included in the second property. We stated here first property since we are going to use mainly this property of the code. The only reason why we require w to be string of distribution rather than just string of symbols is because this way we can perform soft decoding.

Remark 12. *We also want to note that the only reason for defining $PrefixList$ is in order to allow compression for the case the alphabet is of size $O(1)$. For a large alphabet it is enough to require that $|List_i| < \varepsilon' L$ for almost all i.*

Lemma 13. *Let $PL \triangleq PrefixList(w, C, \varepsilon)$, then $agr(C(PL), w(PL)) > \varepsilon|PL|$.*

Proof: We will prove this by induction on number of leaves. If PL has only one leaf v Then PL is a path from root to v and the claim follows from the Lemma 10.

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

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

From Lemma 10 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 14. *For all $0 < \varepsilon < 1$ Σ_{in}, let $|\Sigma_{out}| > (2\Sigma_{in})^{\frac{3}{\varepsilon^2}}$. 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 15. *Let PL be full d-ary tree then there exists at most $(d+1)^{2s}$ rooted subtrees of PL of size s.*

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

IV. Communication transcript

During our protocol Alice and Bob are going to send edges from the tree $\mathcal{T}$ of the original protocol. In order to describe some specific edge from $\mathcal{T}$ it may take $O(T)$ bits. Also as in Braverman Rao [5] we will need one additional level of encoding that will compress our communication. First idea of the compression is that we are not sending random edges, we never send any edge before we send his grandparent. Therefore we can describe an edge by a link to grandparent and a path from grandparent to an edge. This will solve us the problem for the case that we have alphabet polynomial in the length of our communication since in this case we do not care of the length of the links. In case of constant size alphabet the size of the link is important and therefore we sometimes instead of sending link to grandparent we will send link to cousin who may be much closer to the edge we are sending.

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

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

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

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

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

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

Remark 16. *Note that Procedure Add does not works for any edge e, but only for edges at depth at most 2 from already added edges.*

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

- Procedure $\mathbf{size}(A)$: which will return the size of the transcript.
- Procedure $\mathbf{End-round}(A, i)$: which will set the size of the transcript to be maximum between $\log \Sigma_{in}^i$ and $size(A)$ and padding if necessary transcript to this size by adding some 0(which will represent no edge).

V. Recovering from errors using a polynomial size alphabet

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

Theorem 17. *For every $\varepsilon, c > 0$ there exists a protocol π whitch is resilient to $1-\varepsilon$-symmetric noise and which solves the problem of list-decoding of $PJP(T, \Sigma_T)$, where $|\Sigma_T| = T^c$ for some constant $c > 0$ with list of size $O(\frac{1}{\varepsilon})$. Moreover the protocol π runs $O(\frac{T}{\varepsilon})$ rounds and in each round sends $O_{\varepsilon,c}(\log T)$ bits.*

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 previous section. The links in the transcript will be of size at most $O(T)$ therefore every entry of transcript takes at most $O(\log T)$ bits. Let us set 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 14 there exist $C : \Sigma_{in}^n \to \Sigma_{out}^n$ which is $(L, \varepsilon') = (O(\frac{1}{\varepsilon}), O(\varepsilon))$ list tree code with $\log |\Sigma_{out}| = O_\varepsilon(\log |\Sigma_{in}|) = O_{\varepsilon,c}(\log T)$.

Remark 18. *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.

The third one is Σ_{out} which corresponds to an alphabet which we are going to send over noisy channel.

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

[2]In fact we will need cousins only for a small alphabet

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

```
Input: $w_A \in D(\Sigma_{out})^i$
  1) Calculate $Decode(w_A) = List_i(w_A, C, \varepsilon) = \{B_1, B_2, \dots, B_k\}$
  2) For $j = 1, \dots, \min\{k, Ent\}$ do
      a) If $E(B_j) \cup E(A)$ has unique path from root and let $e$ be a last edge on this path then $Add(A, e)$
  3) $End - round(A, i+1)$.
  4) Send $C(A)[i+1]$ to Bob.
```

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

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

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

Let us assume that $\beta < \frac{1}{2}(1-\alpha)$ we are going to show that in this case Alice will output the correct answer. Let $\varepsilon = 1 - 2\beta - \alpha$. Note that $\varepsilon > 0$. Let $\varepsilon' = c\varepsilon$ for some small constant c to be defined later. The protocol is very simple we will perform list decoding protocol from previous section which is resilient to $(1-\varepsilon')$ noise, but instead of outputting a list at the end we will output the closest answer i.e., we will find $c_{output} \in C$ 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 19. *Let C be $(\varepsilon', \frac{1}{\varepsilon'}+1)$ list decodable tree code . For every $x, y \in \Sigma_{in}^n$ let s be first index where $x[s] \neq y[s]$. Then $d(C(x), C(y)) \geq n - s - 2\varepsilon' n$.*

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

$$\begin{aligned} n - \varepsilon' n + agr(C(y)[s, \dots, n], C(x)[s, \dots, n]) \\ \leq \varepsilon' L n = n + \varepsilon' n \end{aligned}$$

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

Let us assume by contradiction that protocol outputs wrong answer. Let us assume that Alice outputs the wrong answer. Let $c_B \in \Sigma_{out}^n$ be a codeword which was sent by Bob. Let us assume that w_A was the codeword received by Alice. Assume 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 a first place where $c_{output}[s] \neq c_B[s]$ and an other is e_{end} output communication point (we will show soon that exists one on the "correct" path c_B). Observe that if output node is located before split point s then we output the correct answer. Thus by contradiction assumption $s < e_{end}$. The proof now follows from Lemma 19 .

Let us define $B_1 = d(c_A[1, \dots s], w_B[1, \dots s])$ and $B_2 = d(c_A[s+1, \dots n], w_B[s+1, \dots n])$ note that $B_1 + B_2 \leq \beta n$. From the Lemma 19 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 thus we get

$$s \geq n - 2B_2 - O(\varepsilon' n) \qquad (1)$$

On other hand it follows that

$$B_1 + \alpha n \geq\geq s - O(\varepsilon' n) \ .$$

Thus we got that

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

Rewrite this and we will get that

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

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

References

[1] P. Elias, “List decoding for noisy channels,” 1957.

[2] J. M. Wozencraft, “List decoding,” *Quarterly Progress Report*, vol. 48, pp. 90–95, 1958.

[3] V. Guruswami, *List decoding of error-correcting codes*. Springer, 2004.

[4] L. J. Schulman, “Coding for interactive communication,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1745–1756, 1996.

[5] M. Braverman and A. Rao, “Towards coding for maximum errors in interactive communication,” in *Proceedings of the 43rd annual ACM symposium on Theory of computing*. ACM, 2011, pp. 159–166.

[6] R. Gelles, A. Moitra, and A. Sahai, “Efficient and explicit coding for interactive communication,” in *FOCS*, R. Ostrovsky, Ed. IEEE, 2011, pp. 768–777. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6108120

[7] M. Braverman, “Towards deterministic tree code constructions,” in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, 2012, pp. 161–167.

[8] Z. Brakerski and Y. T. Kalai, “Efficient interactive coding against adversarial noise,” in *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*. IEEE, 2012, pp. 160–166.

[9] M. Franklin, R. Gelles, R. Ostrovsky, and L. J. Schulman, “Optimal coding for streaming authentication and interactive communication.” in *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 19, 2012, p. 104.

[10] Z. Brakerski and M. Naor, “Fast algorithms for interactive coding.” in *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 20, 2013, p. 14.

[11] S. Agrawal, R. Gelles, and A. Sahai, “Adaptive protocols for interactive communication,” *arXiv preprint arXiv:1312.4182*, 2013.

[12] M. Ghaffari, B. Haeupler, and M. Sudan, “Optimal error rates for interactive coding i: Adaptivity and other settings,” *arXiv preprint arXiv:1312.1764*, 2013.

[13] M. Ghaffari and B. Haeupler, “Optimal error rates for interactive coding ii: Efficiency and list decoding,” *arXiv preprint arXiv:1312.1763*, 2013.