Coding for Interactive Communication Correcting Insertions and Deletions*

Mark Braverman[†1], Ran Gelles[2], Jieming Mao[3], and Rafail Ostrovsky[‡4]

1 **Department of Computer Science, Princeton University, Princeton, USA**
mbraverm@cs.princeton.edu

2 **Department of Computer Science, Princeton University, Princeton, USA**
rgelles@cs.princeton.edu

3 **Department of Computer Science, Princeton University, Princeton, USA**
jiemingm@cs.princeton.edu

4 **Department of Computer Science and Department of Mathematics, UCLA, Los Angeles, USA**
rafail@cs.ucla.edu

Abstract

We consider the question of interactive communication, in which two remote parties perform a computation while their communication channel is (adversarially) noisy. We extend here the discussion into a more general and stronger class of noise, namely, we allow the channel to perform *insertions* and *deletions* of symbols. These types of errors may bring the parties "out of sync", so that there is no consensus regarding the current round of the protocol.

In this more general noise model, we obtain the first interactive coding scheme that has a constant rate and tolerates noise rates of up to $1/18-\varepsilon$. To this end we develop a novel primitive we name *edit distance tree code*. The edit distance tree code is designed to replace the Hamming distance constraints in Schulman's tree codes (STOC 93), with a stronger edit distance requirement. However, the straightforward generalization of tree codes to edit distance does not seem to yield a primitive that suffices for communication in the presence of synchronization problems. Giving the "right" definition of edit distance tree codes is a main conceptual contribution of this work.

1998 ACM Subject Classification E.4 Coding and Information Theory, F.1.2 Models of Computation

Keywords and phrases Interactive communication, coding, edit distance

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.61

* Full version of this paper can be found at http://arxiv.org/abs/1508.00514.

† Mark Braverman is supported in part by an NSF CAREER award (CCF-1149888), NSF CCF-1215990, a Turing Centenary Fellowship, a Packard Fellowship in Science and Engineering, and the Simons Collaboration on Algorithms and Geometry.

‡ Rafail Ostrovsky is supported in part by NSF grants 09165174, 1065276, 1118126 and 1136174, DARPA, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 61; pp. 61:1–61:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In the setting of interactive communication two remote parties, Alice and Bob, wish to run a distributed protocol utilizing a noisy communication channel. The study of this problem was initiated by the seminal work of Schulman [22, 23, 24] who showed a coding scheme for interactive protocols in which the communication complexity of the resilient protocol is larger than the communication of the input (noiseless) protocol by only a constant factor. Schulman's coding schemes tolerates *random noise* where each bit is flipped with a small probability, as well as some *adversarial noise* where the only restriction on the noise is the amount of bits being flipped by the adversary. Subsequently, many works considered the question of interactive communication, obtaining coding schemes reaching optimality in terms of their computational efficiency [13, 14, 2, 4, 3, 15], communication efficiency [19, 17, 12], and noise resilience, both in the standard setting [7, 8, 6, 15], and in various other noise models and settings [21, 10, 11, 16, 15, 1, 5, 9].

The recent successes in developing the theory of interactive error-correcting codes brought the study of two-way interactive coding to nearly match what we know about good codes against adversarial noise in the one-way setting.

So far, all works focused on either substitutions (where Eve can substitute a sent symbol with a different symbol from the alphabet) or erasures (where Eve can substitute a sent symbol with a $\perp$). In this work we extend the question of coding for interactive communication over noisy channels to a more general type of noise. Namely, we consider channels with *insertions and deletions* (indels). In the one-way setting, this corresponds to the insertion and deletion model, where Eve is allowed to completely remove transmitted symbols, or inject new symbols. Note that this model is stronger than the substitution model, since a substitution can always be implemented as a deletion followed by an insertion. Even in the one-way regime, this model is more difficult to analyze than the model with substitution errors. As an example, Schulman and Zuckerman [25] gave a polynomial-time encodable and decodable codes for insertion/deletion errors. Their code can tolerate around $\frac{1}{100}$ fraction of insertion/deletion errors. This should be contrasted with efficient codes in the standard noise setting, e.g., [18], tolerating about $\frac{1}{4}$ fraction of bit flips.

The major additional challenge in dealing with indels in the interactive setting compared to the non-interactive indel model and the interactive substitutions model, is that we can no longer assume that Alice and Bob are synchronized: at a given time they may be at different stages of their sides of the protocol! Indeed, if Eve deletes Alice's transmission to Bob and additionally injects a 'spoofed' reply from Bob back to Alice, then while Bob has received no message and assumes the protocol hasn't advanced yet, Alice has received a spoofed reply, and proceeds to the next step of her protocol. From this point and on, unless the insertion/deletion is detected, the parties are unsynchronized, as they run different steps of the protocol. The challenge in dealing with this model is to design a protocol that manages to succeed even without knowing whether the two parties are synchronized.

1.1 Modeling insertions and deletions

Some care is required when dealing with insertion and deletion noise patterns, as certain choices make the model too strong or too weak. For example, consider the case where Alice and Bob send each other symbols in an alternating way. Then, even if a single deletion of a symbol is allowed the noise can cause the protocol to "hang": Bob will be waiting for a symbol from Alice, while Alice will be waiting for Bob's response. Clearly, such a model is

too strong for our purpose, and we should restrict the allowed noise patterns to preserve the protocol's liveliness.

There are two main paradigms for distributed protocol in which parties are not fully-synchronized. The first is a *message-driven* paradigm, in which each party "sleeps" until the arrival of a new message that triggers it into performing some computation and sending a message to the other party. The second is *clock-driven*, where each party holds a clock: each clock tick, the party wakes up, checks the incoming messages queue, performs some computation, and sends a message to the other side. The issue here is that different parties may have mismatching or skewed clocks. Then, instead of acting in an alternating manner, one party may wake up several times while the other party is still asleep.

We emphasize that if the parties have matching clocks, then no insertions and deletions are possible – channel corruption in this case has either the effect of changing one symbol to another (as in a standard noisy channel), or causing a detectible corruption, i.e., an erasure. Both these types of noise are substantially weaker than insertions and deletions, and were already analyzed in previous work (e.g., [24, 8, 11, 9]).

Our noise model, which we describe shortly, makes sense for both the above paradigms: it guarantees liveliness in a message-driven setting; for the clock-driven model, we can show that any such settings reduces to our model, that is, any resilient protocol in our model can be used to obtain a resilient protocol in the clock-driven setting. The skewness of the clocks in that case, is related to the noise-resilience of the protocol in our model. See the full version for the complete details.

In this work we assume a message-driven setting, where the parties normally speak in alternating manner. Any corruption in our model must be a deletion which is followed by an insertion. We name each such tampering as an *edit corruption*.

▶ **Definition 1** (Edit Corruption)**.** An edit-corruption is a single deletion followed by a single insertion (whether the inserted symbol is aimed at the same or the opposite party as the deleted message).

This gives rise to two types of attacks Eve can perform: (i) delete a symbol and replace it with a different symbol (insert a symbol at the same direction as the deleted symbol; a substitution attack); (ii) delete a symbol, and insert a spoofed reply from the other side (insert a symbol at the opposite direction of the deleted symbol). The second type of corruption has an effect of making the parties 'out-of-sync': one party advances one step in the protocol, while the other does not; see Figure 1. Note that a substitution has a cost a single corruption, i.e., it is counted as a single deletion followed by a single insertion. Also note that although an outside viewer can split Eve's attack into pairs of deletion-and-insertion, the string that a certain party receives, from that party's own view, suffers an *arbitrary* pattern of insertions/deletions.

1.2 Our Results and Techniques

Tree codes with edit distance. When only a single message is to be transmitted (i.e., in the one-way setting), codes that withstand insertions and deletions were first considered by Levenshtein [20]. In such codes, each two codewords must be far away in their *edit distance*, a notion of distance that captures the amount of insertions/deletions it takes to convert one codeword to another. The edit distance replaces the Hamming distance, which essentially counts the amount of bit flips required to turn one codeword to another.

A key ingredient in interactive-communication schemes is the *tree code* [24], a labeled tree such that the labels on each descending path in the tree can be seen as a codeword. The

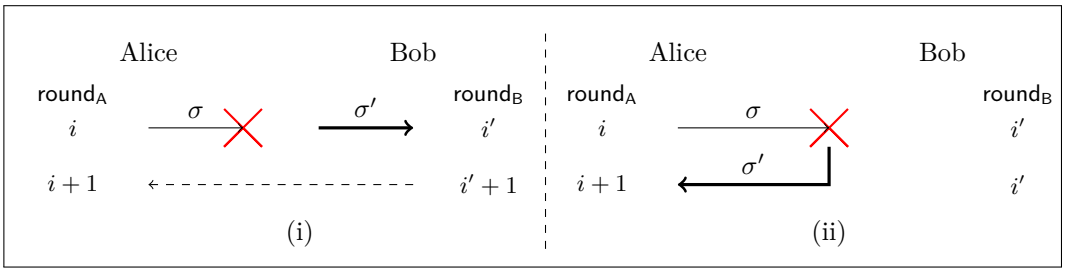


Figure 1 An illustration of the two insertion/deletion attacks: (i) a deletion followed by an insertion in the same direction (a substitution); (ii) a deletion followed by an insertion to the opposite direction (an out-of-sync attack). The deleted transmission is marked with a cross, and the inserted transmission is marked with a bold arrow. The dashed arrow denotes a possible (non-interrupted) reply.

tree code is parametrized by a distance parameter α, and it holds that any two codewords whose paths diverge from the same node, are at least α-apart in their Hamming distance. Encoding a message via a tree code allows a party to *eventually* obtain the message sent by the other side, as long as not too many errors have happened. This in turn allows the parties to correct errors that previously occurred in the simulation, and revert the simulation back into a correct state [24, 8]. In order to keep the communication overhead a constant factor when tree code encoding is used in interactive schemes, it is required that each label of the tree comes from an alphabet of constant size (that is, the size of the alphabet is independent of the tree's depth and thus independent of the length of protocol to simulate).

It is only natural to believe that we could obtain interactive-communication schemes that withstand insertions/deletions by replacing tree codes with a stronger notion of codes, namely, *edit distance tree codes*. In edit distance tree codes, each two codewords (possibly of different lengths) which diverge at a certain point, are required to be far apart in their edit distance rather than their Hamming distance. Yet, since the parties are not synchronized, new difficulties arise. To give a simple example, assume Alice sends one of the two following encodings $s_1 = ABCAAABBB$ and $s_2 = ABCABCAAA$, and assume Bob has receives the string $ABCBBB$. If Bob knew that Alice thinks she is in round 6 of the protocol, he would decode to s_2; if he knew that Alice thinks she is in round 9, he would decode to s_1. Alas, he does not know which is the case!

To mitigate situations in which not being synchronized may hurt us, we require an even stronger property, namely, we wish that the suffixes (of arbitrary lengths) of any two overlapping codewords will have appropriately large edit distance (see Definitions 14 and 15 for the precise condition). This stronger property guarantees that two "branches" in the tree are far apart in their edit distance, even when they are shifted with respect to each other due to lack of synchronization possibly caused by previous indels. We can then show that as long as not too many indels occurred in the *suffix* of the received codeword, the tree-code succeeds to recover the entire sent message. Crucial in this approach is a notion of distance we call *suffix distance* (Definition 21), that measures the amount of noise in a codewords' suffix. This generalizes a distance measure by Franklin et al. [10, 11] (see also Braverman and Efremenko [6]) to the case where the received word may be misaligned with respect to the sent word, due to indels.

Alas, while (Hamming distance) tree codes over a constant alphabet were shown to exist by Schulman [24], it is not clear if such trees exist for edit distance, and if so, for which distance parameter α, as Schulman's proof doesn't carry over to the edit distance case.

Our first result (Section 3) shows the existence of edit distance tree codes, for any distance parameter α,

▶ **Theorem 2** (Informal). *For any $\alpha < 1$ and any $d, n \in \mathbb{N}$ there exists a d-ary edit distance tree code of depth n over a constant-size alphabet.*

As in the case of standard tree codes, finding an *efficient* construction for such trees remains an important open question. Building on the techniques of Gelles, Moitra and Sahai [13, 14] we give in the full version an efficient randomized construction of a relaxed notion for edit-distance tree codes, we call a *potent edit distance* tree codes. These trees satisfy the edit-distance guarantee *almost* everywhere, and are good enough to replace the tree-code notion of Theorem 2 in most applications.

▶ **Theorem 3** (Informal). *For any $\alpha < 1$ and any $d, n \in \mathbb{N}$ there exists a randomized construction of a d-ary potent edit distance tree code of length n over a constant-size alphabet. The construction is efficient and succeeds with overwhelming probability (in n).*

While in the rest of the paper (namely, for our coding scheme) we assume the edit-distance notion of Theorem 2, all our schemes work the same when the tree is replaced with a potent one; see the full version for further details.

Interactive-communication schemes tolerating insertions/deletions. Equipped with edit distance tree codes, we show a protocol that solves the *pointer jumping problem* over a noisy channel with insertions and deletions and exhibits linear communication complexity in the noiseless communication. Since the pointer jumping problem is complete for two-party interactive communication, this implies a coding scheme that can simulate any protocol over a channel that may introduce insertion/deletions. Specifically, in Sections 4 and 5 we show that for any $\varepsilon > 0$ and any noiseless protocol π and inputs x, y, there is a scheme that correctly simulates π (that is, produces the transcript $\pi(x, y)$ at both parties), withstands $1/18 - \varepsilon$ fraction of edit-corruptions, and has a linear communication complexity with respect to the communication of the protocol π.

▶ **Theorem 4.** *For any $\varepsilon > 0$, and for any binary (noiseless) protocol π with communication $CC(\pi)$, there exists a noise-resilient coding scheme with communication $O_\varepsilon(CC(\pi))$ that succeeds in simulating π as long as the adversarial edit-corruption rate is at most $1/18 - \varepsilon$.*

Our coding scheme and analysis follows ideas by Braverman and Rao [8] and by Braverman and Efremenko [6] – first focusing on channels with polynomial-size alphabet and then generalizing to channels with constant-size alphabet – however, the analysis in the light of insertions and deletions is more complicated and subtle. In particular, similar to [6], our analysis uses the notion of suffix distance for relating the effect of the noise to the progress of the simulation.

We note again that due to out-of-sync attacks, it is possible that the parties' belief of the "current" round of the protocol is different. In the worst case, while Alice reaches the end of the coding protocol (say, round N), it is possible that Bob has only reached round $(1-2\rho)N$, e.g., due to $2\rho N$ deletions in his received communication (ρ is the fraction of edit-corruptions in that instance). Therefore, if we wish to tolerate a ρ-fraction of edit-corruptions, it is imperative that the parties output the correct answer already at round $(1-2\rho)N$. Our coding scheme (Theorem 4) satisfies even this more strict requirement.

Finally, we show that for a family of rigid protocols, in which we require *both* parties to output the correct value at round $(1-2\rho)N$, then $\rho = 1/6$ is an upper bound on the admissible edit-corruption rate. Details can be found in the full version.

▶ **Theorem 5** (Informal). *If both parties are required to give output at round* $(1-2\rho)N$*, then no coding scheme of length* N *can tolerate an edit-corruption rate of* $\rho = 1/6$*.*

Closing the gap between the upper bound of 1/6, and the resilience 1/18 achieved by the scheme of Theorem 4 is left for future work.

2 Preliminaries

For any finite set S, we denote by $x \underset{U}{\leftarrow} S$ the case that x is uniformly distributed over S. All logarithms are taken to base 2. We denote the set $\{1, 2, \ldots, n\}$ by $[n]$. For a set Σ we denote $\Sigma^{\leq n} = \cup_{0 \leq i \leq n} \Sigma^i$, and $\Sigma^* = \cup_{i \geq 0} \Sigma^i$. Let $s \in \Sigma^l$ be a string of length $|s| = l$. For $1 \leq i \leq j \leq l$, we use $s[i]$ to denote the i-th symbol of s and $s[i..j]$ to denote the string $s[i] \circ s[i+1] \circ \cdots \circ s[j]$.

▶ **Definition 6** (Pointer Jumping Problem). Any communication protocol of T rounds where the parties alternately exchange bits can be reduced to the following *pointer jumping problem* $PJP(T)$: Let $\mathcal{T}$ be a binary tree of depth T. Alice's input X is a set of *consistent* edges leaving vertices at even depths. Bob's input Y is a set of consistent edges leaving odd-level vertices. A set of edges is *consistent* on a specified set of vertices, if it contains exactly one edge leaving every vertex in that specified set. Due to being consistent on all the nodes, $X \cup Y$ defines a unique root-to-leaf path. The parties' goal is to output this unique path. Note that T alternating rounds of communication suffice to compute this path, assuming noiseless channels.

▶ **Definition 7** (Protocols). An interactive protocol π for a function $f(x, y)$ is a distributed algorithm that dictates for each party, at every round, the next message (symbol) to send given the party's input and the messages received so far. Each transmitted symbol is assumed to be out of a fixed alphabet Σ. The protocol runs for N rounds (also called the length of the protocol), after which the parties give output. An instance of the protocol, on inputs x, y is said to be *correct* if both parties output $f(x, y)$ at the end of the protocol.

In this work we focus on *alternating* protocols, where as long as there is no noise, the protocol runs for $2N$ rounds in which Alice and Bob send symbols alternately. In the presence of ρ-fraction of edit corruptions (i.e., at most $2\rho N$ insertion/deletion errors), it is possible that some party receives only $N(1-2\rho)$ symbols throughout the protocol. We say that the protocol is correct in presence of ρ-fraction of edit corruption if both parties output $f(x, y)$ by round $N(1-2\rho)$ (according to their own round counting, which may differ from the count of the other party).

▶ **Definition 8** (String Matching). We say that $\tau = (\tau_1, \tau_2)$ is a string matching between a sent message sm and a received message rm (denoted $\tau : sm \to rm$), if $|\tau_1| = |\tau_2|$, $del(\tau_1) = sm$, $del(\tau_2) = rm$, and $\tau_1[i] \approx \tau_2[i]$ for all $i = 1, \ldots, |\tau_1|$. Here del is a function that deletes all the $*$'s in the string and two characters a and b satisfy $a \approx b$ if $a = b$ or one of a and b is $*$. We assume that $*$ is a special symbol that does not appear in sm and rm.

▶ **Definition 9** (Edit Distance). The edit distance of between $sm \in \Sigma^*$ and $rm \in \Sigma^*$ is defined as $ED(sm, rm) = \min_{\tau : sm \to rm} sc(\tau_1) + sc(\tau_2)$. Here $sc(\tau_1)$ is the number of $*$'s in τ_1.

▶ **Fact 10** (Triangle Inequality). *For any three strings* x, y, z*,* $ED(x, y) \leq ED(x, z) + ED(y, z)$*.*

▶ **Fact 11.** *For any two strings* x, y*,* $ED(x, y) = |x| + |y| - 2 \cdot LCS(x, y)$*. Here* $LCS(x, y)$ *is the longest common substring of* x *and* y*.*

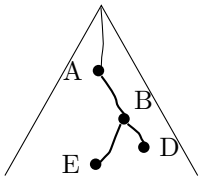


Figure 2 An illustration of lambda structure.

▶ **Lemma 12.** *Let $x \in \Sigma^m$ be some given string, $m \geq n$ and $|\Sigma| \geq 4$. For any constant $\alpha \in [0,1]$, $\Pr_{y \underset{U}{\leftarrow} \Sigma^n}[ED(x,y) \leq \alpha \cdot m] \leq |\Sigma|^{-\frac{(1-\alpha)m}{2}}$.*

The proof of the above Lemma, along with other missing proofs, are deferred to the full version of this work.

3 Edit-distance tree code

In this section we recall the notion of tree-codes [24] and provide a novel primitive, namely, the edit-distance tree-code.

▶ **Definition 13** (Prefix Code)**.** A prefix code $C : \Sigma_{in}^n \to \Sigma_{out}^n$ is a code such that $C(x)[i]$ only depends on $x[1..i]$. C can be also considered as a $|\Sigma_{in}|$-ary tree of depth n with symbols written on edges of the tree using an alphabet size $|\Sigma_{out}|$. On this tree, each tree path from the root of length l corresponds to a string $x \in \Sigma_{in}^l$ and the symbol written on the deepest edge of this path corresponds to $C(x)[l]$.

▶ **Definition 14** (α-bad Lambda)**.** We say that a prefix code C contains an α-bad lambda if when you consider this prefix code as a tree, there exist four tree nodes A,B,D,E such that $B \neq D$, $B \neq E$, B is D and E's common ancestor, A is B's ancestor or B itself, and $ED(AD, BE) \leq \alpha \cdot \max(|AD|, |BE|)$. Here AD and BE are strings of symbols along the tree path from A to D and the tree path from B to E. See Figure 2.

▶ **Definition 15** (Edit-distance Tree Code)**.** We say that a prefix code C: $\Sigma_{in}^n \to \Sigma_{out}^n$ is a α-edit-distance tree code if C does not contain an α-bad lambda.

Our main theorem in this section is the existence of edit-distance tree codes,

▶ **Theorem 16.** *For any $d \geq 2$, $n > 0$ and $0 < \alpha < 1$, there exists an α-edit-distance tree code of depth n with alphabet size $|\Sigma_{in}| = d$ and $|\Sigma_{out}| = (176 \cdot d)^{4/(1-\alpha)}$.*

Proof. We prove this theorem by induction on n. To this end we define a slightly stronger notion than α-bad lambda free, which we call "excellent". Intuitively, if a tree-code C is excellent, then with a good probability, C will not cause a bad lambda in a tree-code that contains C as a subtree. This would allow us to construct lambda-free trees of length n building on lambda-free trees with a smaller depth as subtrees.

▶ **Definition 17** (Potential Probability)**.** For any prefix code $C : \Sigma_{in}^n \to \Sigma_{out}^n$, and any $i \geq 0$, consider C as a tree and define a new (non-regular) tree C' by connecting a simple path of length i to the root of C, making the other end of this path the root of C'. Label each edge along the new path with a symbol from Σ_{out} chosen uniformly and independently.

The potential probability $P_i(C)$ is defined as the probability that the new tree C' has a α-bad lambda with $A =$ the root of C' and $B =$ the root of C.

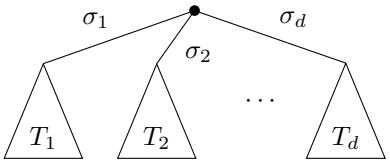
▶ **Definition 18** (Excellent)**.** For a constant $c_1 > 1$, which we will fix shortly, we say that a prefix code C is excellent if $\forall i \geq 0$, $P_i(C) \cdot (d \cdot c_1)^i < 1$ and C is α-bad-lambda-free.

In the proof, we are going to construct a set called S_n which would contain only excellent d-ary prefix codes of depth n with alphabet size $d^{O_\alpha(1)}$. Since excellent is a stronger notion than α-bad lambda-free, if we can construct S_n and show that for any $n > 0$ it is non-empty, then we are done. In each S_n, all codes use the same Σ_{in} and Σ_{out}. We have $|\Sigma_{in}| = d$ and $|\Sigma_{out}| = s$ where the conditions we put on s thorough the following proof are given by:

$$s = \max\left\{ \frac{\alpha d}{(1-\alpha)} \cdot (d \cdot c_1)^{\frac{\alpha}{1-\alpha}} + 1, d^2, (\frac{4d}{c_2})^{\frac{4}{1-\alpha}}, (c_1 \cdot d \cdot 4)^{\frac{4}{1-\alpha}} \right\}.$$

Here $c_1 = 44$, and $c_2 = \frac{1}{11}$. Therefore, taking $s \geq (176d)^{4/(1-\alpha)}$ satisfies all the above conditions.

Let us now inductively construct S_n. For notation convenience, let $S_0 = \{\text{a single node}\}$. For $n > 0$, let

$$\tilde{S}_n = \left\{ T = \right.$$


$$\left| \begin{array}{l} T_1, T_2, ..., T_d \in S_{n-1} \text{ and} \\ \sigma_1, \sigma_2, ..., \sigma_d \in \Sigma_{out} \end{array} \right\}.$$

and define $S_n = \{C \in \tilde{S}_n |\ C \text{ is excellet}\}$. From this definition, we directly have that every $C \in S_n$ is excellent, and we are only left to show that S_n is non-empty. Actually, we are going to prove the following claim by induction.

▶ **Claim 19.** *For all n, $\frac{|S_n|}{|\tilde{S}_n|} \geq c_2 = \frac{1}{11}$.*

Base case ($n = 1$): Consider the following set.

$$S_1' = \{C \mid C \text{ is a } d\text{-ary prefix code of depth 1 and } d \text{ different codewords}\}$$

We want to show that $S_1' \subseteq S_1$. For any $C \in S_1'$, it is clear that C does not have any α-bad lambda. Because in this depth 1 tree, one can only pick $A = B$ to be the root, and D, E to be some different leaves. Then $ED(AD, BE) = 2$ and $|AD| = |BE| = 1$. So $ED(AD, BE) > \alpha \cdot \max(|AD|, |BE|)$. Now let's consider the potential probability $P_i(C)$. Suppose there exists an α-bad lambda in the tree after adding a path of length i. Then in this α-bad lambda, B is the original root, AB is the path added to the root and D and E are two different leaves of the tree. There are two cases to consider:

1. $i = |AB| > \alpha/(1-\alpha)$: In this case, since $|BE| = 1$, $(|AD| - |BE|)/|AD| = 1 - \frac{1}{1+|AB|} > \alpha$. So it is not possible that $ED(AD, BE) < \alpha \cdot |AD|$. Thus $P_i(C) = 0$.
2. $i = |AB| \leq \alpha/(1-\alpha)$: In this case, let W be the event that one of the labels along the path AB equals to one of the d codewords of C. Clearly, when W does not happen, $ED(AD, BE) = |AD| + |BE| > \alpha \cdot |AD|$. It is also immediate that $\Pr[W] \leq i \cdot \frac{d}{s}$. We require $s > \frac{\alpha d}{(1-\alpha)} \cdot (d \cdot c_1)^{\frac{\alpha}{1-\alpha}}$, and get that $P_i(C) \cdot (d \cdot c_1)^i \leq \Pr[W] \cdot (d \cdot c_1)^i \leq \frac{\alpha d}{(1-\alpha)s} \cdot (d \cdot c_1)^{\frac{\alpha}{1-\alpha}} < 1$.

Therefore, we have proved that all the prefix codes in S_1' are excellent and therefore in S_1. Then because $s \geq d^2$, we get $\frac{|S_1|}{|\tilde{S}_1|} \geq \frac{|S_1'|}{|\tilde{S}_1|} = \frac{s(s-1)\times\cdots\times(s-d+1)}{s^d} \geq (1 - 1/d)^d > 1/11 = c_2$.

Inductive step: Suppose we already know that for $1 \le i < n$, $\frac{|S_i|}{|\tilde{S}_i|} \ge c_2$, and let us prove that $\frac{|S_n|}{|\tilde{S}_n|} \ge c_2$. We will use the following lemma, whose proof is quite straightforward.

▶ **Lemma 20.** *If for all* $1 \le i < n$, $\frac{|S_i|}{|\tilde{S}_i|} \ge c_2$. *Then for any* $x \in \Sigma_{out}^j$ *where* $0 < j \le n$ *and* $y \in \Sigma_{in}^n$, *it holds that* $\Pr_{C \xleftarrow[U]{} \tilde{S}_n}[C(y)[1..j] = x] \le c_2^{1-j} s^{-j}$.

In order to show that $\frac{|S_n|}{|\tilde{S}_n|} \ge c_2$, we can choose C randomly from $\tilde{S}_n$, and show that $\Pr[C \text{ is excellent}] \ge c_2$. To this end, consider the conditions of being excellent in turn:

1. Let's first consider $\Pr[P_i(C) \cdot (d \cdot c_1)^i \ge 1]$. By Lemma 20 and Lemma 12, we get

$$\begin{aligned}
&\mathbb{E}_{C \xleftarrow[U]{} \tilde{S}_n}[P_i(C)] \\
\le &\sum_{1 \le n_1, n_2 \le n} \sum_{\substack{x \in \Sigma_{in}^{n_1}, y \in \Sigma_{in}^{n_2}, \\ x[1] \ne y[1]}} \Pr_{\substack{z \xleftarrow[U]{} \Sigma_{out}^i \\ C \xleftarrow[U]{} \tilde{S}_n}} [ED(z \circ C(x), C(y)) \le \alpha \cdot \max(i + n_1, n_2)] \\
\le &\sum_{1 \le n_1, n_2 \le n} d^{n_1+n_2} \cdot c_2^{2-n_1-n_2} \Pr_{x \xleftarrow[U]{} \Sigma_{out}^{n_1}, y \xleftarrow[U]{} \Sigma_{out}^{n_2}, z \xleftarrow[U]{} \Sigma_{out}^i} [ED(z \circ x, y) \le \alpha \cdot \max(i + n_1, n_2)] \\
\le &\sum_{1 \le n_1, n_2 \le n} \left(\frac{d}{c_2}\right)^{n_1+n_2} s^{-\frac{1-\alpha}{2} \cdot \frac{n_1+n_2+i}{2}} \le \sum_{1 \le n_1, n_2 \le n} \left(\frac{d}{c_2} \cdot s^{-\frac{1-\alpha}{4}}\right)^{n_1+n_2} s^{-\frac{(1-\alpha)i}{4}} \\
\le &\sum_{1 \le n_1, n_2 \le n} \left(\frac{1}{4}\right)^{n_1+n_2} (c_1 \cdot d \cdot 4)^{-i} \le \sum_{j=2}^{\infty} \frac{j}{4^j} \cdot (c_1 \cdot d \cdot 4)^{-i} = \frac{7}{36} (c_1 \cdot d \cdot 4)^{-i}
\end{aligned}$$

   By Markov's inequality, $\Pr[P_i(C) \cdot (d \cdot c_1)^i \ge 1] \le \frac{7}{36} \cdot \frac{1}{4^i}$. Then $\sum_{i=0}^{\infty} \Pr[P_i(C) \cdot (d \cdot c_1)^i \ge 1] \le \frac{7}{36} \sum_{i=0}^{\infty} \frac{1}{4^i} = \frac{7}{27} < \frac{5}{11} = \frac{1-c_2}{2}$.
2. Now let's consider the event that C has an α-bad lambda with $A = B = root$. It is easy to see that this event is equivalent to $P_0(C) \ge 1$. Therefore it is covered by the previous case.
3. Now let's consider the event that C has an α-bad lambda with $A = root$ and $B \ne root$. By Lemma 20 and Definition 18, we have

$$\begin{aligned}
&\Pr[C \text{ has an } \alpha\text{-bad lambda with } A = root,\ B \ne root] \\
&\le \sum_{n_1 \le n} d^{n_1} \cdot c_2^{1-n_1} \cdot (d \cdot c_1)^{-n_1} \le \sum_{n_1=1}^{\infty} (\frac{1}{c_1 c_2})^{n_1} = \sum_{n_1=1}^{\infty} (\frac{1}{4})^{n_1} = \frac{1}{3} < \frac{5}{11} = \frac{1-c_2}{2}.
\end{aligned}$$

4. Finally let's consider the case that C has an α-bad lambda with $A \ne root$. It is easy to see that this event never happens because C's subtrees rooted at depth 1 in C are all excellent.

Therefore by union bound, $\Pr[C \text{ is excellent}]$ is at least

$$\begin{aligned}
&\ge 1 - \sum_{i=0}^{\infty} \Pr[P_i(C) \cdot (d \cdot c_1)^i \ge 1] - \Pr[C \text{ has an } \alpha\text{-bad lambda with } A = root,\ B \ne root] \\
&\ge 1 - \frac{1-c_2}{2} - \frac{1-c_2}{2} = c_2.
\end{aligned}$$

◀

3.1 Decoding of edit-distance tree codes

The decoding of a codeword $rm \in \Sigma_{out}^j$ via an edit-distance tree-code C amounts to finding a message whose encoding minimizes the suffix distance to rm, i.e., $\mathsf{DEC}(rm) = \arg\min_{m \in \Sigma_{in}^{\le n}} SD(rm, C(m))$, where the suffix distance $SD(\cdot, \cdot)$ is defined as follows.

▶ **Definition 21** (Suffix Distance). Given any two strings $sm, rm \in \Sigma^*$, the suffix distance between sm and rm is $SD(sm, rm) = \min_{\tau: sm \to rm} \max_{i=1}^{|\tau_1|} \frac{sc(\tau_1[i..|\tau_1|]) + sc(\tau_2[i..|\tau_2|])}{|\tau_1| - i + 1 - sc(\tau_1[i..|\tau_1|])}$.

The following lemma, which plays an important role in the analysis of the simulation protocols that we present in the next sections, shows that if a message sm is encoded by some α-edit-distance tree code and the received message rm satisfies $SD(sm, rm) \leq \frac{\alpha}{2}$, then the receiver can recover the entire sent message correctly.

▶ **Lemma 22.** *Let $C\colon \Sigma_{in}^n \to \Sigma_{out}^n$ be an α-edit-distance tree code, and let $rm \in \Sigma_{out}^m$ (m can be different from n). There exists at most one $sm \in \cup_{i=1}^n C(\Sigma_{in}^n)[1..i]$ such that $SD(sm, rm) \leq \frac{\alpha}{2}$.*

4 A coding scheme with a polynomial alphabet size

In this section, we show a protocol π that solves $PJP(T)$ in $O(T)$ rounds over channels with alphabet size $poly(T)$, and is resilient to (a constant fraction of) insertion/deletion errors. Since the $PJP(T)$ is complete for interactive communication, this implies that any binary protocol with T rounds can be simulated in $O(T)$ rounds over a channel with polynomially-large alphabet that corrupts at most a fraction $1/18 - \varepsilon$ of the transmissions. While this protocol does not exhibit a constant rate, it contains all the main ideas for the constant-rate protocol of Theorem 4, and thus we focus on this simple variant first. Then, in Section 5 we discuss how to reduce the alphabet size and achieve a protocol with $O(T)$ communication complexity with the same resilience guarantees.

Assume π has $2N$ alternating rounds, that is, Alice and Bob send N symbols each, assuming there are no errors. We would like the protocol to resist a fraction of ρ edit-corruptions, that is, the protocol should succeed as long as there are at most $2\rho N$ insertion/deletion errors. Due our assumption that the adversary never causes the protocol to "get stuck", this amounts to at most $2\rho N$ deletions, where each deletion is followed by an insertion.

We assume that Alice and Bob share some fixed α-edit-distance tree code $C : \Sigma_{in}^N \to \Sigma_{out}^N$ given by Theorem 16. We will set the values of N, α, and Σ_{in} later. Currently we only need to know $N = poly(T)$.

Let us begin with a high-level outline of the protocol π. The protocol basically progresses by sending edges in the tree $\mathcal{T}$ of the underlying $PJP(T)$, interactively constructing the joint path (similar to [8]). To communicate an edge e, the parties encode it as a pair of numbers (n, s), where $0 \leq n \leq N$ and $s \in \{1, 2, 3, 4\}$. The value n indicates the number of some previous round in which some edge e' was sent, and the value s determines e as the s-grandchild of e'. That is, we always send an edge e by linking it to an edge e' that was previously sent, such that e' is located two levels above in unique path leading from the root to e. If e does not have a grandparent (e.g., it is the at the first or second level in $\mathcal{T}$), we will set $n = 0$. Sometimes the parties have no edge to send, in which case they set $n = N$ and say that e is an empty edge. We take Σ_{in} to be all the possible encodings (n, s). As $0 \leq n \leq N$ and $1 \leq s \leq 4$, we have $|\Sigma_{in}| = poly(N) = poly(T)$. As $|\Sigma_{out}|$ is polynomial in $|\Sigma_{in}|$ (Theorem 16), we also have $|\Sigma_{out}| = poly(N) = poly(T)$.

The protocol π is described in Protocol 1. The description is for Alice side; Bob's part is symmetric. Here we explain more than the pseudocode on how to get $E(d_A)$. Basically d_A is a string of symbols in Σ_{in} and those symbols are in the form (n, s). $E(d_A)$ will be the set of edges these symbols in d_A represent. To get the edge each symbol (n, s) represents, we will first find the edge sent in n-th round and get its proper grandchild according to s. If n is not in the correct range then we consider d_A as not valid.

Protocol 1 The protocol π

Let $\mathcal{T}$ be given by $PJP(T)$. Recall that Alice's input is X. Assume the parties share some fixed α-edit-distance tree code $C : \Sigma_{in}^N \to \Sigma_{out}^N$.

Initially we set the counter $i = 0$. For any leaf node v in $\mathcal{T}$ we initialize a counter $s(v) = 0$. Run the following for N times.

1. $i \leftarrow i + 1$.
2. Receive a symbol $r_A[i]$ from the other party. (For Alice, if $i = 1$, skip this step)
3. Find $d_A \in \Sigma_{in}^*$ which minimizes $SD(d_A, r_A[1 \dots i])$.
4. If $E(d_A) \cup X$ has a unique path from the root in $\mathcal{T}$, do the following. Here $E(d_A)$ is the set of edges indicated by d_A, if d_A is not a valid string of symbols, $E(d_A) = \emptyset$.
   a. If this path reaches a leaf node v, then $s(v) \leftarrow s(v) + 1$.
   b. Let e be the deepest edge on the the unique path from root. If $e \in X$, and e is either an edge in the first or second level of $\mathcal{T}$ or e's grandparent has been sent, set $s_A[i]$ to be encoding of e, otherwise set $s_A[i]$ to be encoding of an empty edge.
5. If $E(d_A) \cup X$ does not have a unique path from the root in $\mathcal{T}$, set $s_A[i]$ to be encoding of an empty edge.
6. If $i = N(1 - 2\rho)$, output the leaf node v with the largest $s(v)$.
7. Send $C(s_A[1 \dots i])[i]$ to Bob.

We now analyze Protocol 1 and prove it resists up to $(1/18-\varepsilon)$-fraction of edit corruptions. Let N_A and N_B be the counter i of Alice and Bob respectively, when one of them reaches the end of the protocol π. Let $\tau_A = (\tau_1, \tau_2)$ be the string matching between $s_B[1..N_B]$ and $r_A[1..N_A]$ that is consistent with the protocol. Let $\tau_B = (\tau_3, \tau_4)$ be the string matching between $s_A[1..N_A]$ and $r_B[1..N_B]$. Recall that we use $sc(\tau)$ to denote the number of $*$'s in the string. By definition, $sc(\tau_1) + sc(\tau_3) \le 2\rho N$ and $sc(\tau_2) + sc(\tau_4) \le 2\rho N$.

In the analysis we count the number of rounds in which Alice correctly decodes the entire (current) set of edges sent by Bob. We call each such round a *good decoding*.

▶ **Definition 23** (Good Decoding)**.** When a party decodes a message, we say it is a *good decoding* if the decoded messages is exactly the one sent by the other side (i.e., $d_A = s_B[1..i]$ or $d_B = s_A[1..i]$, assuming the other side is at round i), and the symbol just received is not an adversarial insertion. If a decoding is not good, we call it a bad decoding.

In the following lemma show that as long as noise is small enough, there will be many rounds with good decodings. The proof can be found in the full version.

▶ **Lemma 24.** *Alice has at least* $N_A + (1 - \frac{2}{\alpha})sc(\tau_2) - (1 + \frac{2}{\alpha})sc(\tau_1)$ *good decodings. Bob has at least* $N_B + (1 - \frac{2}{\alpha})sc(\tau_4) - (1 + \frac{2}{\alpha})sc(\tau_3)$ *good decodings.*

After we have established that Alice and Bob will have many good decodings, we show that this implies they will have good progress in constructing their joint path in the underlying $PJP(T)$. Consider the $N_A + N_B$ decodings that happen during the protocol, and sort them in a natural order; we say that these decodings occur at "times" $t = 1, 2, \dots, N_A + N_B$. Note that the decodings need not be alternating – Eve's insertions and deletions may cause one party to perform several consecutive decodings while the other party does not receive any symbol, and performs no decoding. We also assume that for each decoding, the sending of the next symbol happens at the same "time" as the decoding. Let $e_A(t)$ be the set of edges Alice has sent at time t and $e_B(t)$ be the set of edges Bob has sent at time t. Let P be the correct path of length T of $PJP(T)$. Define $l(t)$ to be the length of the longest path from

the root using edges in $P \cap (e_A(t) \cup e_B(t))$. Basically $l(t)$ measures how much progress Alice and Bob have made. Let us also define $m(i)$ to be the first time t such that $l(t) \geq i$. For notation convenience, let $m(0) = 0$.

The following lemma shows that if the parties do not make progress, many bad decodings (and thus, many errors) must have occurred. The proof can be found in the full version.

▶ **Lemma 25.** *For $i = 0, \ldots, T-1$, if $m(i+1) \neq m(i)+1$, then during times $m(i)+1, \ldots, m(i+1)-1$, the following is true.*

1. *If i is odd, then there are no good decodings of Bob. The number of good decodings of Alice is at most the number of bad decodings of Bob.*
2. *If i is even, then there are no good decodings of Alice. The number of good decodings of Bob is at most the number of bad decodings of Alice.*

Combining the above lemmas, we get the main theorem for protocols with polynomial size alphabet.

▶ **Theorem 26.** *For any $\varepsilon > 0$, the protocol π of Protocol 1 with $N = \lceil \frac{T}{16\varepsilon} \rceil$, and a $(1-\varepsilon)$-edit tree code, solves PJP(T) and is resilient to a $(1/18 - \varepsilon)$-fraction of edit corruptions.*

Proof. Set $\rho = \frac{1}{18} - \varepsilon$ and $\alpha = 1 - \varepsilon$. Let g_A be the number of good decodings of Alice, $b_A = N_A - g_A$ be the number of bad decodings of Alice. Similarly, let g_B be the number of good decodings of Bob, and let $b_B = N_B - g_b$. Recall that $sc(\tau_1) + sc(\tau_3) = sc(\tau_2) + sc(\tau_4) \leq 2\rho N$, then by Lemma 24, we have

$$b_A + b_B \leq \frac{2}{\alpha}(sc(\tau_1) + sc(\tau_2)) + sc(\tau_1) - sc(\tau_2) + \frac{2}{\alpha}(sc(\tau_3) + sc(\tau_4)) + sc(\tau_3) - sc(\tau_4),$$

thus $b_A + b_B \leq 8\rho N/\alpha$. Then we have,

$$\begin{aligned} g_A = N_A - b_A &\geq N_A - \frac{8\rho N}{\alpha} \geq (N_A - N(1-2\rho)) + N(1-2\rho) - \frac{8\rho N}{\alpha} \\ &\geq b_A + b_B - \frac{8\rho N}{\alpha} - (N_A - N(1-2\rho)) + N(1-2\rho) + \frac{8\rho N}{\alpha} \\ &> b_A + b_B + (N_A - N(1-2\rho)) + N(1-(1-18\varepsilon)(1+2\varepsilon)) \\ &\geq b_A + b_B + (N_A - N(1-2\rho)) + 16\varepsilon N \geq b_A + b_B + (N_A - N(1-2\rho)) + T. \end{aligned}$$

Similarly we have $g_B > b_A + b_B + (N_B - N(1-2\rho)) + T$.

Using Lemma 25 we deduce that by time $m(T)$ the number of good decodings Alice may have is bounded by $T + b_B$. From this point and on, every good decoding at Alice's side adds one vote for the correct leaf, making at least $g_A - (T + b_B) > b_A + (N_A - N(1-2\rho))$ votes for that node by the end of the protocol, and at least $b_A + 1$ votes until Alice reaches round $N(1-2\rho)$ when she gives her output. On the other hand, any wrong output can get at most b_A votes, thus Alice outputs the correct leaf node at round $N(1-2\rho)$. By a similar reasoning, Bob also outputs the correct leaf node when he reaches round $N(1-2\rho)$. ◀

5 A coding scheme with a constant alphabet size

Based on the protocol in Section 4, with some modifications, we obtain a protocol that has a constant size alphabet and a constant rate. To this end, we show how to encode each edge using varying-length encoding over a constant size alphabet. Although substantially more technically involved, this protocol is quite a straightforward extension of the protocol presented above. We thus defer the detailed analysis of this protocol to the full version.

▶ **Theorem 27.** *For any $\varepsilon > 0$, there exists a simulation protocol π' with $N = \lceil \frac{T}{\varepsilon^2} \rceil$, and a $(1-\varepsilon)$-edit distance tree code, solves $PJP(T)$ and is resilient to a $(1/18-\varepsilon)$-fraction of edit corruptions.*

Since $PJP(T)$ is complete for interactive communication, the above theorem proves Theorem 4.

References

1. Shweta Agrawal, Ran Gelles, and Amit Sahai. Adaptive protocols for interactive communication. *arXiv preprint arXiv:1312.4182*, 2013.
2. Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. In *Foundations of Computer Science, IEEE Annual Symposium on*, pages 160–166. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.22.
3. Zvika Brakerski, Yael Tauman Kalai, and Moni Naor. Fast interactive coding against adversarial noise. *J. ACM*, 61(6):35:1–35:30, December 2014. doi:10.1145/2661628.
4. Zvika Brakerski and Moni Naor. Fast algorithms for interactive coding. In *SODA'13: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 443–456, 2013.
5. Gilles Brassard, Ashwin Nayak, Alain Tapp, Dave Touchette, and Falk Unger. Noisy interactive quantum communication. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 296–305, 2014. doi:10.1109/FOCS.2014.39.
6. Mark Braverman and Klim Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. In *Foundations of Computer Science (FOCS), IEEE 55th Annual Symposium on*, pages 236–245, 2014.
7. Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC'11, pages 159–166, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993659.
8. Mark Braverman and Anup Rao. Toward coding for maximum errors in interactive communication. *Information Theory, IEEE Transactions on*, 60(11):7248–7255, Nov 2014. doi:10.1109/TIT.2014.2353994.
9. Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Maximal noise in interactive communication over erasure channels and channels with feedback. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS'15, pages 11–20, New York, NY, USA, 2015. ACM. doi:10.1145/2688073.2688077.
10. Matthew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J. Schulman. Optimal coding for streaming authentication and interactive communication. In Ran Canetti and Juan A. Garay, editors, *CRYPTO'13*, volume 8043 of *LNCS*, pages 258–276. Springer Berlin, 2013. doi:10.1007/978-3-642-40084-1_15.
11. Matthew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J. Schulman. Optimal coding for streaming authentication and interactive communication. *Information Theory, IEEE Transactions on*, 61(1):133–145, Jan 2015. doi:10.1109/TIT.2014.2367094.
12. Ran Gelles and Bernhard Haeupler. Capacity of interactive communication over erasure channels and channels with feedback. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 1296–1311, 2015.
13. Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *Foundations of Computer Science, 2011 IEEE 52nd Annual Symposium on*, pages 768–777. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.51.

14 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *Information Theory, IEEE Transactions on*, 60(3):1899–1913, March 2014. doi:10.1109/TIT.2013.2294186.

15 Mohsen Ghaffari and Bernhard Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. In *Foundations of Computer Science (FOCS), IEEE 55th Annual Symposium on*, pages 394–403, 2014.

16 Mohsen Ghaffari, Bernhard Haeupler, and Madhu Sudan. Optimal error rates for interactive coding I: Adaptivity and other settings. In *STOC'14: Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 794–803, 2014. doi:10.1145/2591796.2591872.

17 Bernhard Haeupler. Interactive channel capacity revisited. In *Foundations of Computer Science (FOCS), IEEE 55th Annual Symposium on*, pages 226–235, 2014.

18 Jørn Justesen. Class of constructive asymptotically good algebraic codes. *Information Theory, IEEE Transactions on*, 18(5):652–656, Sep 1972. doi:10.1109/TIT.1972.1054893.

19 Gillat Kol and Ran Raz. Interactive channel capacity. In *STOC'13: Proceedings of the 45th annual ACM Symposium on theory of computing*, pages 715–724, 2013. doi:10.1145/2488608.2488699.

20 Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

21 Rafail Ostrovsky, Yuval Rabani, and Leonard J. Schulman. Error-correcting codes for automatic control. *Information Theory, IEEE Transactions on*, 55(7):2931–2941, July 2009. doi:10.1109/TIT.2009.2021303.

22 Leonard J. Schulman. Communication on noisy channels: a coding theorem for computation. *Foundations of Computer Science, Annual IEEE Symposium on*, pages 724–733, 1992. doi:10.1109/SFCS.1992.267778.

23 Leonard J. Schulman. Deterministic coding for interactive communication. In *STOC'93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 747–756, New York, NY, USA, 1993. ACM. doi:10.1145/167088.167279.

24 Leonard J. Schulman. Coding for interactive communication. *IEEE Trans. Inf. Theory*, 42(6):1745–1756, 1996.

25 Leonard J. Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *Information Theory, IEEE Transactions on*, 45(7):2552–2557, Nov 1999. doi:10.1109/18.796406.