Synchronization Strings: List Decoding for Insertions and Deletions

Bernhard Haeupler[1]
Carnegie Mellon University, Pittsburgh, PA, USA
haeupler@cs.cmu.edu

Amirbehshad Shahrasbi[2]
Carnegie Mellon University, Pittsburgh, PA, USA
shahrasbi@cs.cmu.edu

Madhu Sudan[3]
Harvard University, Cambridge, MA, USA
madhu@cs.harvard.edu

Abstract

We study codes that are list-decodable under insertions and deletions ("insdel codes"). Specifically, we consider the setting where, given a codeword x of length n over some finite alphabet Σ of size q, $\delta \cdot n$ codeword symbols may be adversarially deleted and $\gamma \cdot n$ symbols may be adversarially inserted to yield a corrupted word w. A code is said to be list-decodable if there is an (efficient) algorithm that, given w, reports a small list of codewords that include the original codeword x. Given δ and γ we study what is the rate R for which there exists a constant q and list size L such that there exist codes of rate R correcting δ-fraction insertions and γ-fraction deletions while reporting lists of size at most L.

Using the concept of *synchronization strings*, introduced by the first two authors [Proc. STOC 2017], we show some surprising results. We show that for every $0 \leq \delta < 1$, every $0 \leq \gamma < \infty$ and every $\varepsilon > 0$ there exist codes of rate $1 - \delta - \varepsilon$ and constant alphabet (so $q = O_{\delta,\gamma,\varepsilon}(1)$) and sub-logarithmic list sizes. Furthermore, our codes are accompanied by efficient (polynomial time) decoding algorithms. We stress that the fraction of insertions can be arbitrarily large (more than 100%), and the rate is independent of this parameter. We also prove several tight bounds on the parameters of list-decodable insdel codes. In particular, we show that the alphabet size of insdel codes needs to be exponentially large in ε^{-1}, where ε is the gap to capacity above. Our result even applies to settings where the unique-decoding capacity equals the list-decoding capacity and when it does so, it shows that the alphabet size needs to be exponentially large in the gap to capacity. This is sharp contrast to the Hamming error model where alphabet size polynomial in ε^{-1} suffices for unique decoding. This lower bound also shows that the exponential dependence on the alphabet size in previous works that constructed insdel codes is actually necessary!

Our result sheds light on the remarkable asymmetry between the impact of insertions and deletions from the point of view of error-correction: Whereas deletions cost in the rate of the code, insertion costs are borne by the adversary and not the code! Our results also highlight the dominance of the model of insertions and deletions over the Hamming model: A Hamming error is equal to one insertion and one deletion (at the same location). Thus the effect of δ-fraction Hamming errors can be simulated by δ-fraction of deletions and δ-fraction of insertions — but insdel codes can deal with much more insertions without loss in rate (though at the price of higher alphabet size).

2012 ACM Subject Classification E.4 Coding and Information Theory

[1] Supported in part by NSF grants CCF-1527110, CCF-1618280 and NSF CAREER award CCF-1750808.
[2] Supported in part by NSF grants CCF-1527110, CCF-1618280 and NSF CAREER award CCF-1750808.
[3] Supported in part by a Simons Investigator Award and NSF Awards CCF 1565641 and CCF 1715187.

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Don Sannella; Article No. 188; pp. 188:1–188:14

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

Keywords and phrases List Decoding, Insertions and Deletions, Synchronization Strings

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.188

Related Version An extended version of this article is available at `https://arxiv.org/pdf/1802.08663.pdf`

1 Introduction

We study the complexity of "insdel coding", i.e., codes designed to recover from insertion and deletion of characters, under the model of "list-decoding", i.e., when the decoding algorithm is allowed to report a (short) list of potential codewords that is guaranteed to include the transmitted word if the number of errors is small enough. Recent work by the first two authors and collaborators [12] has shown major progress leading to tight, or nearly tight, bounds on central parameters of codes (with efficient encoding and decoding algorithms as well) under the setting of *unique* decoding. However the list-decoding versions of these questions were not explored previously. Our work complements the previous studies by exploring list-decoding. In the process our results also reveal some striking features of the insdel coding problem that were not exposed by previous works. To explain some of this, we introduce our model and lay out some of the context below.

1.1 Insdel Coding and List Decoding

We use the phrase "insdel coding" to describe the study of codes that are aimed to recover from *insertions* and *deletions*. The principal question we ask is "what is the *rate* of a code that can recover from γ fraction insertions and δ fraction deletions over a sufficiently large alphabet?". Once the answer to this question is determined we ask how small an alphabet suffices to achieve this rate. We define the terms "rate", "alphabet", and "recovery" below.

An insdel *encoder* over alphabet Σ of block length n is an injective function $E : \Sigma^k \to \Sigma^n$. The associated "code" is the image of the function C. The rate of a code is the ratio k/n. We say that an insdel code C is $(\gamma, \delta, L(n))$-list-decodable if there exists a function $D : \Sigma^* \to 2^C$ such that $|D(w)| \leq L(n)$ for every $w \in \Sigma^*$ and for every codeword $x \in C$ and every word w obtained from x be $\delta \cdot n$ deletions of characters in x followed by $\gamma \cdot n$ insertions, it is the case that $x \in D(w)$. In other words the list-decoder D outputs a list of at most $L(n)$ codewords that is guaranteed to include the transmitted word x if the received word w is obtained from x by at most δ-fraction deletions and γ-fraction insertions. Our primary quest in this paper is the largest rate R for which there exists an alphabet of size $q \triangleq |\Sigma|$ and an infinite family of insdel codes of rate at least R, that are $(\gamma, \delta, L(n))$-list-decodable. Of course we are interested in results where $L(n)$ is very slowly growing with n (if at all). In all results below we get $L(n)$ which is polynomially large in terms of n. Furthermore, when a given rate is achievable we seek codes with efficient encoder and decoder (i.e., the functions E and D are polynomial time computable). Finally we also explore the dependence of the rate on the alphabet size (or vice versa).

Previous Work. Insdel coding was first studied by Levenshtein [18] and since then many bounds and constructions for such codes have been given. With respect to unique decoding, Schulman and Zuckerman [21] gave the first construction of efficient insdel codes over a constant alphabet with a (small) constant relative distance and a (small) constant rate in 1999. Guruswami and Wang [9] gave the first efficient codes over fixed alphabets to correct a deletion fraction approaching 1, as well as efficient binary codes to correct a small constant

fraction of deletions with rate approaching 1. A follow-up work gave new and improved codes with similar rate-distance tradeoffs which can be efficiently decoded from insertions and deletions [5]. Finally, [12] gave codes that can correct δ fraction of synchronization errors with a rate approaching $1-\delta-\varepsilon$ for any $\varepsilon > 0$.

A recent work by Wachter-Zeh [23] considers insdel coding with respect to list decoding and provides Johnson-like upper-bounds for insertions and deletions, i.e., bounds on the list size in terms of the minimum edit-distance of a given code. Moreover, for Varshamov-Tenengolts codes, [23] presents lower bounds on the maximum list size as well as a list-decoding algorithm against a constant number of insertions and deletions.

Several other variants of the insdel coding problem have been studied in the previous work and are summarized by the following surveys [22, 20, 19].

1.2 Our Results

We now present our results on the rate and alphabet size of insdel coding under list-decoding. Two points of contrast that we use below are corresponding bounds in (1) the Hamming error setting for list-decoding and (2) the insdel coding setting with unique-decoding.

1.2.1 Rate Under List Decoding

Our main theorem for list-decoding shows that, given $\gamma, \delta, \varepsilon \geq 0$ there is a $q = q_{\varepsilon,\gamma}$ and a slowly growing function $L = L_{\varepsilon,\gamma}(n)$ such that there are q-ary insdel codes that achieve a rate of $1-\delta-\varepsilon$ that are $(\gamma, \delta, L(n))$-list decodable. Furthermore the encoding and decoding are efficient! The formal statement of the main result is as follows.

▶ **Theorem 1.** *For every $0 < \delta, \varepsilon < 1$ and $\gamma > 0$, there exist a family of list-decodable insdel codes that can protect against δ-fraction of deletions and γ-fraction of insertions and achieves a rate of at least $1-\delta-\varepsilon$ or more over an alphabet of size $\left(\frac{\gamma+1}{\varepsilon^2}\right)^{O\left(\frac{\gamma+1}{\varepsilon^3}\right)} = O_{\gamma,\varepsilon}(1)$. These codes are list-decodable with lists of size $L_{\varepsilon,\gamma}(n) = \exp(\exp(\exp(\log^* n)))$, and have polynomial time encoding and decoding complexities.*

The rate in the theorem above is immediately seen to be optimal even for $\gamma = 0$. In particular an adversary that deletes the last $\delta \cdot n$ symbols already guarantees an upper bound on the rate of $1-\delta$.

We now contrast the theorem above with the two contrasting settings listed earlier. Under unique decoding the best possible rate that can be achieved with δ-fraction deletions and γ-fraction insertions is upper bounded by $1-(\gamma+\delta)$. Matching constructions have been achieved, only recently, by Haeupler and Shahrasbi [12]. In contrast our rate has no dependence on γ and thus dominates the above result. The only dependence on γ is in the alphabet size and list-size and we discuss the need for this dependence later below.

We now turn to the standard "Hamming error" setting: Here an adversary may change an arbitrary δ-fraction of the codeword symbols. In this setting it is well-known that given any $\varepsilon > 0$, there are constants $q = q(\varepsilon)$ and $L = L(\varepsilon)$ and an infinite family of q-ary codes of rate at least $1-\delta-\varepsilon$ that are list-decodable from δ fraction errors with list size at most L. In a breakthrough from the last decade, Guruswami and Rudra [6] showed explicit codes that achieve this with efficient algorithms. The state-of-the-art results in this field yield list size $L(n) = o(\log^{(r)} n)$ for any integer r where $\log^{(r)}$ is the rth iterated logarithm and alphabet size $2^{\tilde{O}(\varepsilon^{-2})}$ [11], which are nearly optimal.

The Hamming setting with δ-fraction errors is clearly a weaker setting than the setting with δ-fraction deletions and $\gamma \geq \delta$ fraction of insertions in that an adversary of the latter

kind can simulate the former. (A Hamming error is a deletion followed by an insertion at the same location.) The insdel setting is thus stronger in two senses: it allows $\gamma > \delta$ and gives greater flexibility to the adversary in choosing locations of insertions and deletions. Yet our theorem shows that the stronger adversary can still be dealt with, without qualitative changes in the rate. The only difference is in the dependence of q and L on γ, which we discuss next.

We briefly remark at this stage that, while our result simultaneously "dominates" the results of Haeupler and Shahrasbi [12] as well as Guruswami and Rudra [6], this happens because we use their results in our work. We elaborate further on this in Section 3. Indeed our first result (see Theorem 13) shows how we can obtain Theorem 1 by using capacity achieving "list-recoverable codes" in combination with synchronization strings in a modular fashion.

We believe that using the codes from Theorem 1 in the construction of long-distance synchronization strings from [13] will give synchronization strings that allow us to reduce the decoding complexity of insdel codes of Theorem 1 and [12] to near-linear time.

1.2.2 Rate versus Alphabet Size

We now turn to understanding how large the alphabet size needs to be as a function of δ, ε and γ. We consider two extreme cases, first with only deletions (i.e., $\gamma = 0$ and then with only insertions (i.e., with $\delta = 0$).

We start first with the insertion-only setting. We note here that one cannot hope to find a constant rate family of codes that can protect n symbols out of an alphabet of size q against $(q-1)n$ many insertions or more. This is so since, with $(q-1)n$ insertions, one can turn any string $y \in [1..q]^n$ into the fixed sequence $1, 2, \cdots, q, 1, 2, \cdots, q, \cdots, 1, 2, \cdots, q$ by simply inserting $q-1$ many symbols around each symbol of y to construct a $1, \cdots, q$ there. Hence, Theorem 2 only focuses on codes that protect n rounds of communication over an alphabet of size q against γn insertions for $\gamma < q - 1$.

▶ **Theorem 2.** *Any list-decodable family of codes $\mathcal{C}$ that protects against γ fraction of insertions for some $\gamma < q-1$ and guarantee polynomially-large list size in terms of block length cannot achieve a rate R that is strictly larger than* $1 - \log_q(\gamma+1) - \gamma\left(\log_q \frac{\gamma+1}{\gamma} - \log_q \frac{q}{q-1}\right)$.

In particular, the theorem asserts that if the code has rate $R = 1 - \varepsilon$, then its alphabet size must be exponentially large in $1/\varepsilon$, namely, $q \geq (\gamma+1)^{1/\varepsilon}$.

Next, we turn to the deletion-only case. Here again we note that no constant rate q-ary code can protect against $\delta \geq \frac{q-1}{q}$ fraction of deletions since such a large fraction of deletions may remove all but the most frequent symbol of codewords. Therefore, Theorem 3 below only concerns codes that protect against $\delta \leq \frac{q-1}{q}$ fraction of deletions.

▶ **Theorem 3.** *Any list-decodable family of insdel codes that protect against δ-fraction of deletions (and no insertions) for some $0 \leq \delta < \frac{q-1}{q}$ that are list-decodable with polynomially-bounded list size has rate R upper bounded as below:*

- $R \leq f(\delta) \triangleq (1-\delta)\left(1 - \log_q \frac{1}{1-\delta}\right)$ *where $\delta = \frac{d}{q}$ for some integer d.*
- $R \leq (1 - q\delta')f\left(\frac{d}{q}\right) + q\delta' f\left(\frac{d+1}{q}\right)$ *where $\delta = \frac{d}{q} + \delta'$ for some integer d and $0 \leq \delta' < \frac{1}{q}$.*

In particular if $\delta = d/q$ for integer d and rate is $1 - \delta - \varepsilon$ then the theorem above asserts that $q \geq \left(\frac{1}{1-\delta}\right)^{\frac{1-\delta}{\varepsilon}}$, or in other words q must be exponentially large in $1/\varepsilon$. Indeed such a statement is true for all δ as asserted in the corollary below. (A detailed proof is available in the extended version)

▶ **Corollary 4.** *There exists a function $f : (0,1) \to (0,1)$ such that any family of insdel codes that protects against δ-fraction of deletions with polynomially bounded list sizes and has rate $1 - \delta - \varepsilon$ must have alphabet size $q \geq \exp\left(\frac{f(\delta)}{\varepsilon}\right)$.*

Implications for Unique Decoding. Even though the main thrust of this paper is list-decoding, Corollary 4 also has implications for unique-decoding. (This turns out to be a consequence of the fact that the list-decoding radius for deletions-only equals the unique-decoding radius for the same fraction of deletions.) We start by recalling the main result of Haeupler and Shahrasbi [12]: Given any $\alpha, \varepsilon > 0$ there exists a code of rate $1 - \alpha - \varepsilon$ over an alphabet of size $q = \exp(1/\varepsilon)$ that *uniquely* decodes from any α-fraction synchronization errors, i.e., from γ-fraction insertions and δ-fraction deletions for any pair $0 \leq \gamma, \delta$ satisfying $\gamma + \delta \leq \alpha$. Furthermore, this is the best possible rate one can achieve for α-fraction synchronization error. (See the extended version for a more detailed description with proof.)

Till now this exponential dependence of the alphabet size on ε was unexplained. This is also in sharp contrast to the Hamming error setting, where codes are known to get ε close to unique decoding capacity (half the "Singleton bound" on the distance of code) with alphabets of size polynomial in $1/\varepsilon$. Indeed given this contrast one may be tempted to believe that the exponential growth is a weakness of the "synchronization string" approach of Haeupler and Shahrasbi [12]. But Corollary 4 actually shows that an exponential bound is necessary. We state this result for completeness even though it is immediate from the Corollary above, to stress its importance in understanding the nature of synchronization errors.

▶ **Corollary 5.** *There exists a function $f : (0,1) \to (0,1)$ such that for every $\alpha, \varepsilon > 0$ every family of insdel codes of rate $1 - \alpha - \varepsilon$ that protects against α-fraction of synchronization errors with unique decoding must have alphabet size $q \geq \exp\left(\frac{f(\delta)}{\varepsilon}\right)$.*

Corollary 5 follows immediately from Corollary 4 by setting $\delta = \alpha$ and $\gamma = 0$ (so we get to the zero insertion case) and noticing that a unique-decoding insdel code for α-fraction synchronization error is also a list-decoding insdel code for δ-fractions of deletions (and no insertions). The alphabet size lower bound for the latter is also an alphabet size lower bound for the former.

1.2.3 Analysis of Random Codes

Finally, in Section 5, we provide an analysis of random codes and compute the rates they can achieve while maintaining list-decodability against insertions and deletions. Such rates are essentially lower-bounds for the capacity of insertion and deletion channels and can be compared against the upper-bounds provided in Section 4.

Theorem 6 shows that the family of random codes over an alphabet of size q can, with high probability, protect against δ-fraction of deletions for any $\delta < 1 - 1/q$ up to a rate of $1-(1-\delta)\log_q \frac{1}{1-\delta} - \delta \log_q \frac{1}{\delta} - \delta \log_q(q-1) = 1 - H_q(\delta)$ using list decoding with super-constant list sizes in terms of their block length where H_q represents the q-ary entropy function.

▶ **Theorem 6.** *For any alphabet of size q and any $0 \leq \delta < \frac{q-1}{q}$, the family of random codes with rate $R < 1 - (1-\delta)\log_q \frac{1}{1-\delta} - \delta \log_q \frac{1}{\delta} - \delta \log_q(q-1) - \frac{1-\delta}{l+1}$ is list-decodable with list size of l from any δ fraction of deletions with high probability. Further, the family of random deletion-codes with rate $R > 1 - (1-\delta)\log_q \frac{1}{1-\delta} - \delta \log_q \frac{1}{\delta} - \delta \log_q(q-1)$ is not list-decodable with high probability.*

Further, Theorem 7 shows that the family of random block codes over an alphabet of size q can, with high probability, protect against γ fraction of insertions for any $\gamma < q - 1$ up

to a rate of $1 - \log_q(\gamma + 1) - \gamma \log_q \frac{\gamma+1}{\gamma}$ using list decoding with super-constant list sizes in terms of block length.

▶ **Theorem 7.** *For any alphabet of size q and any $\gamma < q - 1$, the family of random codes with rate $R < 1 - \log_q(\gamma + 1) - \gamma \log_q \frac{\gamma+1}{\gamma} - \frac{\gamma+1}{l+1}$ is list-decodable with a list size of l from any γn insertions with high probability.*

2 Definitions and Preliminaries

2.1 Synchronization Strings

In this section, we briefly recapitulate synchronization strings, introduced by Haeupler and Shahrasbi [12] and further studied in [14, 13]. We will review important definitions and techniques from [12] that will be of use throughout this paper.

Synchronization strings are recently introduced mathematical objects that turn out to be useful tools to overcome synchronization errors, i.e., symbol insertion and symbol deletion errors. The general idea employed in [12, 14] to obtain resilience against synchronization errors in various communication setups is *indexing* each symbol of the communication with symbols of a synchronization string and then *guessing* the actual position of received symbols on the other side using indices. [12] provides a variety of different guessing strategies that guarantee a large number of correct guesses and then overcome the incorrect guesses by utilizing classic error correcting codes. As a matter of fact, synchronization strings essentially translate synchronization errors into ordinary Hamming type errors which are strictly easier to handle. We now proceed to review some of the above-mentioned definitions and techniques more formally.

Suppose that two parties are communicating over a channel that suffers from α-fraction of insertions and deletions and one of the parties sends a pre-shared string S of length n to the other one. A distorted version of S will arrive at the receiving end that we denote by S'. A symbol $S[i]$ is called to be a *successfully transmitted* symbol if it is not removed by the adversary. A *decoding algorithm* on the receiving side is an algorithm that, for any received symbol, guesses its actual position in S by either returning a number in $[1..n]$ or $\top$ which means the algorithm is not able to guess the index. For such a decoding algorithm, a successfully transmitted symbol whose index is not guessed correctly by the decoding algorithm is called a *misdecoding*.

Haeupler and Shahrasbi [12] introduce synchronization strings and find several decoding algorithms for them providing strong misdecoding guarantees and then design insertion-deletion codes based on those decoding algorithms. As details of those algorithms are not relevant to this paper we avoid further discussion of those techniques. To conclude this section we introduce ε-synchronization strings and an important property of them.

▶ **Definition 8** (ε-synchronization string)**.** String $S \in \Sigma^n$ is an ε-synchronization string if for every $1 \leq i < j < k \leq n + 1$ we have that $\text{EditDistance}\left(S[i, j), S[j, k)\right) > (1 - \varepsilon)(k - i)$ where $\text{EditDistance}(x, y)$ is the smallest number of insertions and deletions needed to convert x to y.

The key idea in the construction of insdel codes in [12] is to index an error correcting code with a synchronization string. Here we provide a formal definition of indexing operation.

▶ **Definition 9** (Indexing)**.** The operation of indexing code $\mathcal{C}$ with block length n and String S of length n is to simply replace each codeword $w_1, w_2, \cdots, w_n$ with $(w_1, s_1), (w_2, s_2), \cdots, (w_n, s_n)$. Clearly, this operations expands the alphabet of the code.

It is shown in [12, 13] that ε-synchronization strings exist over alphabets of sizes polynomially large in terms of ε^{-1} and can be efficiently constructed. An important property of ε-synchronization strings discussed in [12] is the self matching property defined as follows.

▶ **Definition 10** (ε-self-matching property)**.** String S satisfies ε-self-matching property if for any two sequences of indices $1 \leq a_1 < a_2 < \cdots < a_k \leq |S|$ and $1 \leq b_1 < b_2 < \cdots < b_k \leq |S|$ that satisfy $S[a_i] = S[b_i]$ and $a_i \neq b_i$, k is not larger than $\varepsilon|S|$.

In the end, we review the following theorem from [12] that shows the close connection between synchronization string property and the self-matching property.

▶ **Theorem 11** (Theorem 6.4 from [12])**.** *If S is an ε-synchronization string, then all substrings of S satisfy ε-self-matching property.*

2.2 List Recoverable Codes

A code $\mathcal{C}$ given by the encoding function $\mathcal{E} : \Sigma^{nr} \to \Sigma^n$ is called to be (α, l, L)-list recoverable if for any collection of n sets $S_1, S_2, \cdots, S_n \subset \Sigma$ of size l or less, there are at most L codewords for which more than αn elements appear in the list that corresponds to their position, i.e.,

$$|\{x \in \mathcal{C} \mid |\{i \in [n] \mid x_i \in S_i\}| \geq \alpha n\}| \leq L.$$

The study of list-recoverable codes was inspired by Guruswami and Sudan's list-decoder for Reed-Solomon codes [7]. Since then, list-recoverable codes have became a very useful tool in coding theory [1, 2, 3, 4] and there have been a variety of constructions provided for them by several works [6, 8, 10, 17, 16, 11, 15]. In this paper, we will make use of the following capacity-approaching polynomial-time list-recoverable codes given by Hemenway, Ron-Zewi, and Wootters [15] that is obtained by altering the approach of Guruswami and Xing [10].

▶ **Theorem 12** (Hemenway et. al. [15, Theorem A.7])**.** *Let q be an even power of a prime, and choose $l, \epsilon > 0$, so that $q \geq \epsilon^{-2}$. Choose $\rho \in (0, 1)$. There is an $m_{min} = O(l \log_q(l/\epsilon)/\epsilon^2)$ so that the following holds for all $m \geq m_{min}$. For infinitely many n (all n of the form $q^{e/2}(\sqrt{q} - 1)$ for any integer e), there is a deterministic polynomial-time construction of an F_q-linear code $C : \mathbb{F}_{q^m}^{\rho n} \to \mathbb{F}_{q^m}^{n}$ of rate ρ and relative distance $1 - \rho - O(\epsilon)$ that is $(1 - \rho - \epsilon, l, L)$-list-recoverable in time $\text{poly}(n, L)$, returning a list that is contained in a subspace over $\mathbb{F}_q$ of dimension at most $\left(\frac{l}{\epsilon}\right)^{2^{\log^*(mn)}}$.*

3 List Decoding for Insertions and Deletions

In this section, we prove Theorem 1 by constructing a list-decodable code of rate $1 - \delta - \varepsilon$ that provides resilience against $0 < \delta < 1$ fraction of deletions and γ fraction of insertions over a constant-sized alphabet. Our construction heavily relies on the following theorem that, in the same fashion as [12], uses the technique of indexing an error correcting code with a synchronization string to convert a given list-recoverable code into an insertion-deletion code.

▶ **Theorem 13.** *Let $\mathcal{C} : \Sigma^{nR} \to \Sigma^n$ be a (α, l, L)-list recoverable code with rate R, encoding complexity T_{Enc} and decoding complexity complexity T_{Dec}. For any $\varepsilon > 0$ and $\gamma \leq \frac{l\varepsilon}{2} - 1$, by indexing $\mathcal{C}$ with an $\frac{\varepsilon^2}{4(1+\gamma)}$-synchronization string, one can obtain an L-list decodable insertion-deletion code $\mathcal{C}' : \Sigma^{nr} \to [\Sigma \times \Gamma]^n$ that corrects from $\delta < 1 - \alpha - \varepsilon$ fraction of deletions and γ fraction of insertions where $|\Gamma| = \left(\varepsilon^2/(1+\gamma)\right)^{-O(1)}$. $\mathcal{C}'$ is encodable and decodable in $O(T_{Enc} + n)$ and $O(T_{Dec} + n^2(1+\gamma^2)/\varepsilon)$ time respectively.*

We take two major steps to prove Theorem 13. In the first step (Theorem 15), we use the synchronization string indexing technique from [12] and show that by indexing the symbols that are conveyed through an insertion-deletion channel with symbols of a synchronization string, the receiver can make *lists* of candidates for any position of the sent string such that $1-\delta-\varepsilon$ fraction of lists are guaranteed to contain the actual symbol sent in the corresponding step and the length of the lists is guaranteed to be smaller than some constant $O_{\gamma,\varepsilon}(1)$.

In the second step, we use list-recoverable codes on top of the indexing scheme to obtain a list decoding using lists of candidates for each position produced by the former step.

We start by the following lemma that directly implies the first step stated in Theorem 15.

▶ **Lemma 14.** *Assume that a sequence of n symbols denoted by $x_1x_2\cdots x_n$ is indexed with an ε-synchronization string and is communicated through a channel that suffers from up to δn deletions for some $0 \leq \delta < 1$ and γn insertions. Then, on the receiving end, it is possible to obtain n lists $A_1, \cdots, A_n$ such that, for any desired integer K, for at least $n \cdot \left(1 - \delta - \frac{1+\gamma}{K} - K \cdot \varepsilon\right)$ of them, $x_i \in A_i$. All lists contain up to K elements and the average list size is at most $1+\gamma$. These lists can be computed in $O\left(K(1+\gamma)n^2\right)$ time.*

Proof. The decoding algorithm we propose to obtain the lists that satisfy the guarantee promised in the statement is the global algorithm introduced in Theorem 6.14 of Haeupler and Shahrasbi [12].

Let S be the ε-synchronization string used for indexing and S' be the index portion of the received string on the other end. Note that S is pre-shared between the sender and the receiver. The decoding algorithm starts by finding a longest common substring M_1 between S and S' and adding the position of any matched element from S' to the list that corresponds to its respective match from side S. Then, it removes every symbol that have been matched from S' and repeats the previous step by finding another longest common subsequence M_2 between S and the remaining elements of S'. This procedure is repeated K times to obtain $M_1, \cdots, M_K$. This way, lists A_i are formed by including every element in S' that is matched to $S[i]$ in any of $M_1, \cdots, M_K$.

A_i contains the actual element that corresponds to $S[i]$, denoted by $S'[j]$, if and only if $S[i]$ is successfully transmitted (i.e., not removed by the adversary), appears in one of M_ks, and matches to $S[i]$ in M_k. Hence, there are three scenarios under which A_i does not contain its corresponding element $S[i]$.

1. $S[i]$ gets deleted by the adversary.
2. $S[i]$ is successfully transmitted but, as $S'[j]$ on the other side, it does not appear on any of M_ks.
3. $S[i]$ is successfully transmitted and, as $S'[j]$ on the other side, it appears in some M_k although it is matched to another element of S.

The first case happens for at most δn elements as adversary is allowed to delete up to δn many elements.

To analyze the second case, note that the sizes of M_ks descend as k grows since we pick the longest common subsequence in each step. If by the end of this procedure p successfully transmitted symbols are still not matched in any of the matchings, they form a common subsequence of size p between S and the remainder of S'. This leads to the fact that $|M_1| + \cdots + |M_K| \geq K \cdot p$. As $|M_1| + \cdots + |M_K|$ cannot exceed $|S'|$, we have $p \leq |S'|/K$. This bounds above the number of symbols falling into the second category by $|S'|/K$.

Finally, as for the third case, we draw the reader's attention to the fact that each successfully transmitted $S[i]$ which arrives at the other end as $S'[j]$ and mistakenly gets matched to another element of S like $S[k]$ in some M_t, implies that $S[i] = S[k]$. We call

the pair (i, k) *a pair of similar elements in* S *implied by* M_t. Note that there is an actual monotone matching M' from S to S' that corresponds to adversary's actions. As M_t and M' are both monotone, the set of similar pairs in S implied by M_t is a self-matching in S. As stated in Theorem 11, the number of such pairs cannot exceed $n\varepsilon$. Therefore, there can be at most $n\varepsilon$ successfully transmitted symbols that get mistakenly matched in M_t for any t. Hence, the number of elements falling into the third category is at most $nK\varepsilon$.

Summing up all above-mentioned bounds gives that the number of bad lists can be bounded above by $n\delta + \frac{|S'|}{K} + nK\varepsilon \leq n\left(\delta + \frac{1+\gamma}{K} + K\varepsilon\right)$. This proves the list quality guarantee. As proposed decoding algorithm computes longest common substring K many times between two strings of length n and $(1+\gamma)n$ or less, it will run in $O(K(1+\gamma)\cdot n^2)$ time. ◀

▶ **Theorem 15.** *Suppose that n symbols denoted by $x_1, x_2, \cdots, x_n$ are being communicated through a channel suffering from up to δn deletions for some $0 \leq \delta < 1$ and γn insertions for some constant $\gamma \geq 0$. If one indexes these symbols with an $\varepsilon' = \frac{\varepsilon^2}{4(1+\gamma)}$-synchronization string, then, on the receiving end, it is possible to obtain n lists $A_1, \cdots, A_n$ of size $2(1+\gamma)/\varepsilon$ such that, for at least $n\cdot(1-\delta-\varepsilon)$ of them, $x_i \in A_i$. These lists can be computed in $O\left(n^2(1+\gamma)^2/\varepsilon\right)$ time.*

Proof. Using an $\varepsilon' = \frac{\varepsilon^2}{4(1+\gamma)}$-synchronization string in the statement of Lemma 14 and choosing $K = \frac{2(1+\gamma)}{\varepsilon}$ directly gives that the runtime is $O\left(n^2(1+\gamma)^2/\varepsilon\right)$ and list hit ratio is at least $n\cdot\left(1-\delta-\frac{1+\gamma}{K} - K\cdot\varepsilon'\right) = n\cdot(1-\delta-\varepsilon/2-\varepsilon/2) = n\cdot(1-\delta-\varepsilon)$ ◀

Theorem 15 facilitates the conversion of list-recoverable error correcting codes into list-decodable insertion-deletion codes as stated in Theorem 13.

Proof of Theorem 13. To prove this, we simply index code $\mathcal{C}$, entry by entry, with an $\varepsilon' = \frac{\varepsilon^2}{4(1+\gamma)}$ synchronization string. In the decoding procedure, according to Theorem 15, the receiver can use the index portion of the received symbol to maintain lists of up to $2(1+\gamma)/\varepsilon \leq l$ candidates for each position of the sent codeword of $\mathcal{C}$ so that $1-\delta-\varepsilon > \alpha$ fraction of those contain the actual corresponding sent message. Having such lists, the receiver can use the decoding function of $\mathcal{C}$ to obtain an L-list-decoding for $\mathcal{C}'$. Finally, the alphabet size and encoding complexity follow from the fact that synchronization strings over alphabets of size ${\varepsilon'}^{-O(1)}$ can be constructed in linear time [12, 13]. ◀

One can use any list-recoverable error correcting code to obtain insertion-deletion codes according to Theorem 13. In particular, using the efficient capacity-approaching list-recoverable code introduced by Hemenway, Ron-Zewi, and Wootters [15], one obtains the insertion-deletion codes as described in Theorem 1.

Proof of Theorem 1. By setting parameters $\rho = 1-\delta-\frac{\varepsilon}{2}$, $l = \frac{2(\gamma+1)}{\varepsilon}$, and $\epsilon = \frac{\varepsilon}{4}$ in Theorem 12, one can obtain a family of codes $\mathcal{C}$ that achieves rate $\rho = 1-\delta-\frac{\varepsilon}{2}$ and is (α, l, L)-recoverable in polynomial time for $\alpha = 1-\delta-\varepsilon/4$ and some $L = \exp\left(\exp\left(\exp\left(\log^* n\right)\right)\right)$ (by treating γ and ε as constants). Such family of codes can be found over an alphabet $\Sigma_{\mathcal{C}}$ of size $q = (l/\epsilon)^{O(l/\epsilon^2)} = \left(\frac{\gamma+1}{\varepsilon^2}\right)^{O\left(\frac{\gamma+1}{\varepsilon^3}\right)} = O_{\gamma,\varepsilon}(1)$ or infinitely many integer numbers larger than q.

Plugging this family of codes into the indexing scheme from Theorem 13 by choosing the parameter $\varepsilon' = \frac{\varepsilon}{4}$, one obtains a family of codes that can recover from $1-\alpha-\varepsilon' = 1-(1-\delta-\varepsilon/4)-\varepsilon/4 = \delta$ fraction of deletions and γ-fraction of insertions and achieves a rate of $\frac{1-\delta-\varepsilon/2}{1+\log|\Sigma_S|/\log|\Sigma_{\mathcal{C}}|}$ which, by taking $|\Sigma_{\mathcal{C}}|$ large enough in terms of ε, is larger than $1-\delta-\varepsilon$. As $\mathcal{C}$ is encodable and decodable in polynomial time, the encoding and decoding complexities of the indexed code will be polynomial as well. ◀

▶ Remark. We remark that by using capacity-approaching near-linear-time list-recoverable code introduced in Theorem 7.1 of Hemenway, Ron-Zewi, and Wootters [15] in the framework of Theorem 13, one can obtain similar list-decodable insertion-deletion codes as in Theorem 1 with a randomized quadratic time decoding. Further, one can use the efficient list-recoverable in the recent work of Guruswami and Xing [11] to obtain same result as in Theorem 1 except with polylogarithmic list sizes.

4 Upper Bounds on the Rate of List-Decodable Synchronization Codes

4.1 Deletion Codes (Theorem 3)

Proof of Theorem 3. To prove this claim, we propose a strategy for the adversary which can reduce the number of strings that may possibly arrive at the receiving side to a number small enough that implies the claimed upper bound for the rate.

We start by proving the theorem for the case where δq is integer. For an arbitrary code $\mathcal{C}$, upon transmission of any codeword, the adversary can remove all occurrences of δq least frequent symbols as the total number of appearances of such symbols does not exceed δn. In case there are more deletions left, adversary may choose to remove arbitrary symbols among the remaining ones. This way, the received string would be a string of $n(1-\delta)$ symbols consisted of only $q-q\delta$ many distinct symbols. Therefore, one can bound above the size of the ensemble of strings that can possibly be received by the$|\mathcal{E}| \leq \binom{q}{q(1-\delta)} [q(1-\delta)]^{n(1-\delta)}$. As the best rate that any $L = \text{poly}(n)$-list decodable code can get is at most $\frac{\log(|\mathcal{E}|\cdot L)}{n\log q} = \frac{\log|\mathcal{E}|}{n\log q} + o(1)$, the following would be an upper bound for the best rate one might hope for.

$$\frac{\log|\mathcal{E}|}{n\log q} + o(1) = \frac{\log\binom{q}{q(1-\delta)} + n(1-\delta)(\log(q(1-\delta)))}{n\log q} + o(1) = (1-\delta)\left(1 - \log_q \frac{1}{1-\delta}\right) + o(1)$$

This shows that for the case where $q\delta$ is an integer number, there are no family of codes that achieve a rate that is strictly larger than $(1-\delta)\left(1-\log_q \frac{1}{1-\delta}\right)$.

We now proceed to the general case where $\delta = d/q + \delta'$ for some integer d and $0 \leq \delta' < \frac{1}{q}$. We closely follow the idea that we utilized for the former case. The adversary can partition n sent symbols into two parts of size $nq\delta'$ and $n(1-q\delta')$, and then, similar to the former case, removes the $d+1$ least frequent symbols from the first part by performing $\frac{d+1}{q}\cdot nq\delta'$ deletions and d least frequent symbols from the second one by performing $\frac{d}{q}\cdot n(1-q\delta')$ ones. This is possible because $\frac{d+1}{q}\cdot nq\delta' + \frac{d}{q}\cdot n(1-q\delta') = n\delta$. Doing so, the string received after deletions would contain up to $q-d-1$ distinct symbols in its first $nq\delta'\,(1-(d+1)/q)$ positions and up to $q-d$ distinct symbols in the other $n(1-q\delta')\,(1-d/q)$ positions. Therefore, the size of the ensemble of strings that can be received is bounded above as follows.

$$|\mathcal{E}| \leq \binom{q}{q-d-1} [q-d-1]^{nq\delta'\left(1-\frac{d+1}{q}\right)} \cdot \binom{q}{q-d} [q-d]^{n(1-q\delta')\left(1-\frac{d}{q}\right)}$$

This bounds above the rate of any family of list-decodable insdel codes by the following.

$$\begin{aligned}
&\frac{\log|\mathcal{E}|}{n\log q} \\
&= \frac{\log\binom{q}{q-d-1} + nq\delta'\left(1-\frac{d+1}{q}\right)\log(q-d-1) + \log\binom{q}{q-d} + n(1-q\delta')\left(1-\frac{d}{q}\right)\log(q-d)}{n\log q} \\
&= q\delta'\left[\left(1-\frac{d+1}{q}\right)\left(1-\log_q\frac{1}{1-(d+1)/q}\right)\right] + (1-q\delta')\left[\left(1-\frac{d}{q}\right)\left(1-\log_q\frac{1}{1-d/q}\right)\right]
\end{aligned}$$

◀

4.2 Insertion Codes (Theorem 2)

Before providing the proof of Theorem 2, we first point out that any $q-1$ insertions can be essentially used as a single erasure. As a matter of fact, by inserting $q-1$ symbols around the first symbol adversary can make a $1, 2, \cdots, q$ substring around first symbol and therefore, essentially, make the receiver unable to gain any information about it. In fact, with γn insertions, the adversary can repeat this procedure around any $\left\lfloor \frac{\gamma n}{q-1} \right\rfloor$ symbols he wishes. This basically gives that, with γn insertions, adversary can *erase* $\left\lfloor \frac{\gamma n}{q-1} \right\rfloor$ many symbols. Thus, one cannot hope for finding list-decodable codes with rate $1 - \frac{\gamma}{q-1}$ or more protecting against γn insertions.

Proof of Theorem 2. To prove this, consider a code $\mathcal{C}$ with rate $R \geq 1 - \log_q(\gamma+1) - \gamma\left(\log_q \frac{\gamma+1}{\gamma} - \log_q \frac{q}{q-1}\right) + \varepsilon$ for some $\varepsilon > 0$. We will show that there exist c_0^n many codewords in C that can be turned into one specific string $z \in [1..q]^{n(\gamma+1)}$ with γn insertions for some constant $c_0 > 1$ that merely depends on q and ε.

First, the lower bound assumed for the rate implies that

$$|C| = q^{nR} \geq q^{n\left(1-\log_q(\gamma+1)-\gamma\left(\log_q \frac{\gamma+1}{\gamma} - \log_q \frac{q}{q-1}\right)+\varepsilon\right)}. \tag{1}$$

Let Z be a random string of length $(\gamma+1)n$ over the alphabet $[1..q]$. We compute the expected number of codewords of $\mathcal{C}$ that are subsequences of Z denoted by X.

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{y\in\mathcal{C}} \Pr\{y \text{ is a subsequence of } Z\} \\
&= \sum_{y\in\mathcal{C}} \sum_{1\leq a_1<a_2<\cdots<a_n\leq n(\gamma+1)} \frac{1}{q^n}\left(1-\frac{1}{q}\right)^{a_n-n} && (2)\\
&= |\mathcal{C}|\,(q-1)^{-n} \sum_{l=n}^{n(1+\gamma)} \binom{l}{n}\left(\frac{q-1}{q}\right)^l \\
&\leq |\mathcal{C}|\,(q-1)^{-n}\, n\gamma \binom{n(1+\gamma)}{n}\left(\frac{q-1}{q}\right)^{n(1+\gamma)} && (3)\\
&= n\gamma|\mathcal{C}|(q-1)^{n\gamma} q^{-n(1+\gamma)} 2^{n(1+\gamma)H\left(\frac{1}{1+\gamma}\right)+o(n)} \\
&= n\gamma|\mathcal{C}| q^{n\left(\gamma\log_q(q-1)-1-\gamma+\log_q(1+\gamma)+\gamma\log_q\frac{1+\gamma}{\gamma}\right)+o(1)} \\
&= q^{n\varepsilon+o(n)} && (4)
\end{aligned}$$

Step (2) is obtained by conditioning the probability of y being a subsequence of Z over the leftmost occurrence of y in Z indicated by $a_1, a_2, \cdots, a_n$ as indices of Z where the leftmost occurrence of y is located. In that event, Z_{a_i} has to be similar to y_i and y_i cannot appear in $Z[y_{i-1}+1, y_i-1]$. Therefore, the probability of this event is $\left(\frac{1}{q}\right)^n \left(1-\frac{1}{q}\right)^{a_n-n}$. To verify Step (3), we show that the summation in previous step takes its largest value when $l = n(1+\gamma)$ and bound the summation above by $n\gamma$ times that term. To see that $\binom{l}{n}\left(\frac{q-1}{q}\right)^l$ is maximized for $l = n(1+\gamma)$ in $n \leq l \leq n(1+\gamma)$ it suffices to show that the ratio of

consecutive terms is larger than one for $l \leq n(1+\gamma)$:

$$\frac{\binom{l}{n}\left(\frac{q-1}{q}\right)^{l}}{\binom{l-1}{n}\left(\frac{q-1}{q}\right)^{l-1}} = \frac{l}{l-n}\cdot\frac{q-1}{q} = \frac{1-\frac{1}{q}}{1-\frac{n}{l}} \geq 1$$

The last inequality follows from the fact that $l \leq n(\gamma+1) \leq nq \Rightarrow \frac{1}{q} < \frac{n}{l}$.

Finally, by (4), there exists some $z \in [1..q]^{(a+1)n}$ to which at least $q^{\varepsilon n + o(n)}$, i.e., exponentially many codewords of $\mathcal{C}$ are subsequences. Therefore, polynomial-sized list decoding for received message z is impossible and proof is complete. ◀

5 Analysis of Random Codes

5.1 Random Insertion Codes (Theorem 7)

Proof of Theorem 7. We prove the claim by considering a random insertion code $\mathcal{C}$ that maps any $x \in [1..q]^{Rn}$ to some uniformly at random chosen member of $[1..q]^n$ denoted by $E_{\mathcal{C}}(x)$ and showing that it is possible to list-decode $\mathcal{C}$ with high probability.

Note that in an insertion channel, the original message sent by Alice is a substring of the message received on Bob's side. Therefore, a random insertion code $\mathcal{C}$ is l-list decodable if for any $z \in [1..q]^{(\gamma+1)n}$, there are at most l codewords of $\mathcal{C}$ that are subsequences of z. For some fixed $z \in [1..q]^{(\gamma+1)n}$, the probability of some uniformly at random chosen $y \in [1..q]^n$ being a substring of z can be bounded above as follows.

$$\begin{aligned}
\Pr_y\{y \text{ is a subsequence of } z\} &\leq \binom{(\gamma+1)n}{n} q^{-n} \\
&= 2^{n(\gamma+1)H(\frac{1}{\gamma+1})+o(n)} q^{-n} \\
&= q^{n\left(\log_q(\gamma+1)+\gamma\log_q\frac{\gamma+1}{\gamma}-1+o(1)\right)}
\end{aligned}$$

Therefore, for a random code $\mathcal{C}$ of rate R and any $m_1, \cdots, m_{l+1} \in [1..q]^{nR}$ and some fixed $z \in [1..q]^{n(\gamma+1)}$:

$$\Pr\{E_{\mathcal{C}}(m_1), \cdots, E_{\mathcal{C}}(m_{l+1}) \text{ are subsequences of } z\} \leq q^{n(l+1)\left(\log_q(\gamma+1)+\gamma\log_q\frac{\gamma+1}{\gamma}-1+o(1)\right)}$$

Hence, using the union bound over $z \in [1..q]^{n(\gamma+1)}$, for the random code $\mathcal{C}$:

$$\begin{aligned}
&\Pr_{\mathcal{C}}\left\{\exists z \in [1..q]^{n(\gamma+1)}, m_1, \cdots, m_{l+1} \in q^{nR} \text{ s.t. } E_{\mathcal{C}}(m_1), \cdots \text{ are subsequences of } z\right\} \\
\leq\ & q^{n(\gamma+1)}\left(q^{Rn}\right)^{l+1} q^{n(l+1)\left(\log_q(\gamma+1)+\gamma\log_q\frac{\gamma+1}{\gamma}-1+o(1)\right)} \\
=\ & q^{n(\gamma+1)+Rn(l+1)+n(l+1)\left(\log_q(\gamma+1)+\gamma\log_q\frac{\gamma+1}{\gamma}-1+o(1)\right)} \qquad (5)
\end{aligned}$$

As long as q's exponent in (5) is negative, this probability is less than one and drops exponentially to zero as n grows.

$$\begin{aligned}
& n(\gamma+1)+Rn(l+1)+n(l+1)\left(\log_q(\gamma+1)+\gamma\log_q\frac{\gamma+1}{\gamma}-1+o(1)\right) < 0 \\
\Leftrightarrow\quad & R < 1-\log_q(\gamma+1)-\gamma\log_q\frac{\gamma+1}{\gamma}-\frac{\gamma+1}{l+1}+o(1) \qquad (6)
\end{aligned}$$

Therefore, the family of random codes with any rate R that satisfies (6) is list-decodable with a list of size l with high probability. ◀

The analysis for random deletion codes (Theorem 6) can be found in the extended version of this article.

References

1. Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 658–667. IEEE, 2001.
2. Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 812–821. ACM, 2002.
3. Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 126–135. ACM, 2003.
4. Venkatesan Guruswami and Piotr Indyk. Linear-time list decoding in error-free settings. In *International Colloquium on Automata, Languages, and Programming*, pages 695–707. Springer, 2004.
5. Venkatesan Guruswami and Ray Li. Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 620–624. IEEE, 2016.
6. Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.
7. Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 28–37. IEEE, 1998.
8. Venkatesan Guruswami and Carol Wang. Optimal rate list decoding via derivative codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 593–604. Springer, 2011.
9. Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *IEEE Transactions on Information Theory*, 63(4):1961–1970, 2017.
10. Venkatesan Guruswami and Chaoping Xing. List decoding reed-solomon, algebraic-geometric, and gabidulin subcodes up to the singleton bound. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 843–852. ACM, 2013.
11. Venkatesan Guruswami and Chaoping Xing. Optimal rate list decoding over bounded alphabets using algebraic-geometric codes. *arXiv preprint arXiv:1708.01070*, 2017.
12. Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: Codes for insertions and deletions approaching the singleton bound. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2017.
13. Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: Explicit constructions, local decoding, and applications. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2018.
14. Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization strings: Channel simulations and interactive coding for insertions and deletions. *Proceedings of the International Conference on Automata, Languages, and Programming (ICALP)*, 2017.
15. Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes and applications. *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2017.
16. Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. In *International Colloquium on Automata, Languages, and Programming*, pages 701–712. Springer, 2015.
17. Swastik Kopparty. List-decoding multiplicity codes. *Theory of Computing*, 11(5):149–182, 2015.

511 **18** Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals.
512 *Doklady Akademii Nauk SSSR 163*, 4:845–848, 1965.
513 **19** Hugues Mercier, Vijay K Bhargava, and Vahid Tarokh. A survey of error-correcting codes
514 for channels with symbol synchronization errors. *IEEE Communications Surveys & Tutori-*
515 *als*, 12(1), 2010.
516 **20** Michael Mitzenmacher. A survey of results for deletion channels and related synchronization
517 channels. *Probability Surveys*, 6:1–33, 2009.
518 **21** Leonard J. Schulman and David Zuckerman. Asymptotically good codes correcting inser-
519 tions, deletions, and transpositions. *IEEE transactions on information theory*, 45(7):2552–
520 2557, 1999.
521 **22** Neil JA Sloane. On single-deletion-correcting codes. *Codes and designs*, 10:273–291, 2002.
522 **23** Antonia Wachter-Zeh. List decoding of insertions and deletions. *IEEE Transactions on*
523 *Information Theory*, 2017.