Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes

Venkatesan Guruswami
Carnegie Mellon University
Pittsburgh, PA 15143
Email: guruswami@cmu.edu

Ray Li
Carnegie Mellon University
Pittsburgh, PA 15143
Email: ryli@andrew.cmu.edu

***Abstract*—This work constructs codes that are efficiently decodable from a constant fraction of *worst-case* insertion and deletion errors in three parameter settings: (i) Binary codes with rate approaching 1; (ii) Codes with constant rate for error fraction approaching 1 over fixed alphabet size; and (iii) Constant rate codes over an alphabet of size k for error fraction approaching $(k-1)/(k+1)$. When errors are constrained to deletions alone, efficiently decodable codes in each of these regimes were constructed recently. We complete the picture by constructing similar codes that are efficiently decodable in the insertion/deletion regime.**

I. Introduction

This work addresses the problem of constructing error-correcting codes that can be efficiently decoded from a constant fraction of *worst-case* insertions and deletions. The main results generalize analogous results for the situation when errors are restricted to only deletions.

Coding for a constant fraction of adversarial insertions and deletions has been considered previously by Schulman and Zuckerman [1]. They construct constant-rate binary codes that are efficiently decodable from a small constant fraction of worst-case insertions and deletions and can also handle a small fraction of transpositions.

Our work primarily builds off recent results by Guruswami, Bukh, and Wang [2], [3], which address the construction and efficient decoding of codes for constant fractions of deletions. These works establish three results, providing families of codes with each of the following parameters.

1) Families with rate approaching 1 decoding a constant fraction of deletions
2) Families with constant rate decoding a fraction of deletions approaching 1
3) Families over a fixed alphabet of size k with constant rate and decoding a fraction of deletions approaching $1-\frac{2}{k+1}$ (In particular, one gets binary codes for correcting a deletion fraction approaching $1/3$.)

Over an alphabet of size k, it is impossible to have a constant rate code that corrects a $1-\frac{1}{k}$ fraction of deletions. The last result establishes that the maximum correctable fraction of deletions of a constant rate code is $1-\Theta(\frac{1}{k})$.

Combinatorially, decoding a given number of worst-case insertions and deletions is identical to decoding the same number of worst-case deletions. This is established in the following lemma, originally given by Levenshtein [4].

Lemma I.1. *Let $C \subseteq [k]^n$ be a code, and let $t < n$ be a positive integer. The following are equivalent. 1) C is decodable under up to t insertions. 2) C is decodable under up to t deletions. 3) C is decodable under up to t insertions and deletions.*

Lemma I.1 establishes that the codes provided in the three constructions must also be capable of decoding both insertions and deletions. The task that remains, and which our work addresses, is to construct codes in the *same parameter settings* that can efficiently correct a combination of insertions *and* deletions.

The regime under which errors are insertions and deletions is closely related to *edit-distance* (also known as Levenshtein distance), which measures errors of a code under insertions, deletions, and substitutions. A substitution can be viewed as a deletion followed by an insertion. Thus, all results established in the insertion and deletion regime, both constructive and algorithmic, hold in the edit-distance regime when the number of errors is cut in half, and therefore in the traditional coding theory setting in which the only errors are substitutions. The edit-distance is a more challenging model, however; while the Gilbert-Varshamov bound gives codes over size k alphabets that can correct up to a fraction of substitutions approaching $\frac{1}{2}(1-\frac{1}{k})$, the question of whether there exist positive rate codes capable of correcting a deletion fraction approaching $1-\frac{1}{k}$ is still open.

A. Prior Results

These are the efficiently decodable code constructions in the deletion-only regime that we are generalizing to the insertion/deletion regime.

1) A binary code family of rate $1-\tilde{O}(\sqrt{\epsilon})$ that can be efficiently decoded from an ϵ fraction of worst-case deletions, for all ϵ smaller than some absolute constant $\epsilon_0 > 0$. Furthermore, the codes are constructible, encodable, and decodable, in time $N^{\text{poly}(1/\epsilon)}$, where N is the block length. [Theorem 4.1 from [3]]

2) For any $\epsilon > 0$, a code family over an alphabet of size $\text{poly}(1/\epsilon)$ and rate $\Omega(\epsilon^2)$ that can be decoded from a $1-\epsilon$ fraction of worst-case deletions. Furthermore, this code is constructible, encodable, and decodable in time $N^{\text{poly}(1/\epsilon)}$. [Theorem 3.1 from [3]]

978-1-5090-1805-5/16/$31.00 ©2016 IEEE

3) For all integers $k \geq 2$ and all $\epsilon > 0$, a code family over alphabet size k of positive rate $r(k, \epsilon) > 0$ that can be decoded from a $1 - \frac{2}{k+1} - \epsilon$ fraction of worst-case deletions in $O_{k,\epsilon}(N^3 (\log N)^{O(1)})$ time.

B. Our Results and Techniques

Our work constructs the following three families of codes.

1) alphabet size: 2, rate: $1 - \tilde{O}(\sqrt{\epsilon})$, insertion/deletion fraction: ϵ, decoding time: $N^{\text{poly}(1/\epsilon)}$. (Thm. III.2)
2) alphabet size: $\text{poly}(1/\epsilon)$, rate: $\Omega(\epsilon^5)$, insertion/deletion fraction: $1 - \epsilon$, decoding time: $N^{\text{poly}(1/\epsilon)}$. (Thm. IV.4)
3) alphabet size: $k \geq 2$, rate: $(\epsilon/k)^{\text{poly}(1/\epsilon)}$, insertion/deletion fraction: $1 - \frac{2}{k+1} - \epsilon$, decoding time: $O_{k,\epsilon}(N^3 \operatorname{poly} \log(N))$. (Thm. IV.7)

Remark. Theorem IV.4 gives constant rate codes that decode from a $1 - \epsilon$ fraction of insertions/deletions. This also follows as a corollary from Theorem IV.7. However, the rate of the construction in Theorem IV.7 is $(\epsilon/k)^{\text{poly}(1/\epsilon)}$, which is far worse than $\text{poly}(\epsilon)$. The main point of IV.7 is to highlight the near-tight trade-off between alphabet size and insertion/deletion fraction.

Remark. At the expense of slightly worse parameters, the construction and decoding complexities in Theorems III.2 and IV.4 can be improved to $\text{poly}(N) \cdot (\log N)^{\text{poly}(1/\epsilon)}$. See Theorems III.7 and IV.5.

Theorems IV.4 and IV.7 use the powerful idea of list decoding, exemplified in [2]. A normal decoding algorithm is required to return the exact codeword, but a list decoding algorithm is allowed to return a list of codewords containing the correct codeword. The codes for both theorems are decoded by first applying a list decoding algorithm, and then noting that if the easier list decoding is gauranteed to succeed (that is, returns a list containing the correct codeword), one can simply pass through the resulting list and choose the unique codeword that has sufficiently small distance from the received word. The codeword will be unique because the codes constructed are provably decodable under the required number of insertion/deletions according to the results in [2], [3].

The extent of difference between the insertion/deletion decoding algorithms and their deletion-only analogues varies depending on the parameter setting. For a $1 - \frac{2}{k+1} - \epsilon$ fraction of insertions/deletions, the decoding algorithm uses the same list decoding approach as the deletion-only decoding algorithm in [2]. For a $1 - \epsilon$ fraction of insertions/deletions, we adopt the list decoding approach that in fact simplifies the construction presented in [3]. For achieving a rate of $1 - \epsilon$, we use the same code as in [3] with different parameters, but considerably more bookkeeping is done to provide a provably correct decoding algorithm. In particular, both Theorem 4.1 from [3] and Theorem III.2 place chunks of 0s between inner codewords. However, while identifying buffers in the received word in the deletion-only case merely requires identifying long runs of 0s, identifying buffers in the insertion/deletion case requires identifying strings of fixed length with sufficiently small fraction of 1s.

II. Preliminaries

Let $[n]$ denote the set $\{1, \ldots, n\}$. For a string s, let $|s|$ denote the length of the string. Define $\Delta(c, c')$ to be the *insertion/deletion distance* between c and c', that is, the number of insertions/deletions needed to manipulate c into c'. For two words $c, c' \in C$, let $\text{LCS}(c, c')$ be the length of the longest common subsequence of c and c'. Define $\text{LCS}(C) = \max_{c,c' \in C, c \neq c'} \text{LCS}(c, c')$. For the same reason that Lemma I.1 is true, we have $\Delta(c, c') = |c| + |c'| - 2\,\text{LCS}(c, c')$.

A code C of block length n over an alphabet Σ is a subset $C \subseteq \Sigma^n$. The rate of C is defined to be $\frac{\log |C|}{n \log |\Sigma|}$. The encoding function of a code is a map $\text{Enc} : [|C|] \to \Sigma^n$ whose image equals C (with messages identified with $[|C|]$ in some canonical way), and the decoding function of a code is a map $\text{Dec} : \Sigma^* \to C$.

A code is encodable in time $f(n)$ if, for all elements of $[|C|]$, the map Enc can be computed in time $f(n)$. A code is decodable from t (or, a δ fraction of) worst-case insertions and deletions in time $f(n)$ if, for all $c \in C$, and for all s such that $\Delta(s, c) \leq t$ (or δn), $\text{Dec}(s)$ can be computed in time $f(n)$ and evaluates to c. A code is constructible in time $f(n)$ if descriptions of C, Dec, and Enc can be produced in time $f(n)$.

Just as in [1], [2], [3], our constructions use the idea of code concatenation: If $C_{\text{out}} \subseteq \Sigma_{\text{out}}^n$ is an "outer code" with encoding function Enc_{out}, and $C_{\text{in}} \subseteq \Sigma_{\text{in}}^m$ is an "inner code" with encoding function $\text{Enc}_{\text{in}} : \Sigma_{\text{out}} \to \Sigma_{\text{in}}^m$, then the concatenated code $C_{\text{out}} \circ C_{\text{in}} \subseteq \Sigma_{\text{in}}^{nm}$ is a code whose encoding function first applied Enc_{out} to the message, and then applied Enc_{in} to each symbol of the resulting outer codeword.

Thoughout the paper, c, c' denote codewords, s, s' denote codewords modified under insertions and deletions, and w, w' denote inner codewords of concatenated codes. We let n denote the block length of the code, unless we deal with a concatenated code, in which case n denotes the block length of the outer code, m denotes the block length of the inner code, and $N = nm$ denotes the block length of the entire code. Alphabet sizes are denoted by k, and field sizes for outer Reed Solomon codes are denoted by q.

III. High Rate

Lemma III.1 (Proposition 2.5 of [3]). *Let $\delta, \beta \in (0, 1)$. Then, for every m, there exists a code $C \subseteq \{0, 1\}^m$ of rate $R = 1 - 2h(\delta) - O(\log(\delta m)/m) - 2^{-\Omega(\beta m)/m}$ such that*

- *for every string $s \in C$, every interval of length βm in s, contains at least $\beta m/10$ 1's,*
- *C can be corrected from a δ fraction of worst-case deletions, and*
- *C can be found, encoded, and decoded in time $2^{O(m)}$.*

Theorem III.2. *There exists a constant $\epsilon_0 > 0$ such that the following holds. Let $0 < \epsilon < \epsilon_0$. There is an explicit binary code $C \subseteq \{0, 1\}^N$ with rate $1 - \tilde{O}(\sqrt{\epsilon})$ that is decodable* *from an ϵ fraction of insertions/deletions in $N^{\mathrm{poly}(1/\epsilon)}$ time. Furthermore, C can be constructed and encoded in time $N^{\mathrm{poly}(1/\epsilon)}$.*

Proof. With hindsight, let $\epsilon_0 = \frac{1}{121^2}$, and let $0 < \epsilon < \epsilon_0$. Consider the concatenated construction with the outer code being a Reed-Solomon code that can correct a $60\sqrt{\epsilon}$ fraction of errors and erasures. For each $1 \leq i \leq n$, we replace the ith coordinate c_i with the pair (i, c_i); to ensure that this doesn't affect the rate much, we take the RS code to be over $\mathbb{F}_{q^h}$, where $n = q$ is the block length and $h = 1/\epsilon$. We encode each outer symbol pair in the inner code, defined as follows.

The inner code is a good binary insertion/deletion code C_1 of block length m decoding a $\delta = 40\sqrt{\epsilon} < \frac{1}{2}$ fraction of insertions and deletions, such that every interval of length $\delta m/16$ in a codeword has at least $1/10$ fraction of 1s. This code can be found using Lemma III.1. We also assume each codeword begins and ends with a 1.

Now take our concatenated Reed-Solomon code of block length mn, and between each pair of adjacent inner codewords of C_1, insert a *chunk* of δm 0s. This gives us our final code C with block length $N = nm(1 + \delta)$.

Lemma III.3. *The rate of C is $1 - \tilde{O}(\sqrt{\epsilon})$.*

Proof. The rate of the outer RS code is $(1 - 120\sqrt{\epsilon})\frac{h}{h+1}$, and the rate of the inner code can be taken to be $1 - 2h(\delta) - o(1)$ by Lemma III.1. Adding in the buffers reduces the rate by a factor of $\frac{1}{1+\delta}$. Combining these with our choice of δ gives us a total rate for C of $1 - \tilde{O}(\sqrt{\epsilon})$. □

Lemma III.4. *The code C can be decoded from an ϵ fraction of insertions and deletions in time $N^{\mathrm{poly}(1/\epsilon)}$.*

Consider the following algorithm that runs in time $N^{\mathrm{poly}(1/\epsilon)}$ for decoding the received word:

1) Scan from the left of the received word. Every time we encounter a substring of length exactly δm with at most $\frac{1}{160}$ fraction of 1s (or $\delta m/160$ 1s), mark it as a *decoding buffer*. Then, continue scanning from the end of the buffer and repeat. This guarantees no two buffers overlap. This takes time $\mathrm{poly}(N)$.

2) Start with an empty set L. The buffers divide the received word into strings which we call *decoding windows*. For each decoding window, apply the decoder from Lemma III.1 to recover a pair (i, r_i). If we succeed, add this pair to L. This takes $N^{\mathrm{poly}(1/\epsilon)}$ time.

3) If for any i, L contains multiple pairs with first coordinate i, remove all such pairs from L. L thus contains at most one pair (i, r_i) for each index i. Then apply the RS decoding algorithm to the string r whose ith coordinate is r_i if $(i, r_i) \in L$ and erased otherwise. This takes time $\mathrm{poly}(N)$.

Remark. In the deletion only case, the decoding buffers are runs of at least $\delta m/2$ contiguous zeros. Runs of consecutive zeros are obviously a poor choice for decoding buffers in the presence of insertions, as we can destroy any buffer with a constant number of insertions.

Note that the total number of insertions/deletions we can make is at most $(1 + \delta)mn\epsilon < 2\epsilon mn$.

Suppose our received codeword is $s = u_1 \circ y_1 \circ u_2 \circ \cdots \circ u_{n'}$, where $y_1, \ldots, y_{n'-1}$ are the identified decoding buffers and $u_1, \ldots, u_{n'}$ are the decoding windows. Then consider a canonical mapping from characters of c to characters of s where u_i is mapped to by a substring t_i of c, y_i is mapped to by a string x_i, so that $c = t_1 \circ x_1 \circ \cdots \circ t_{n'}$ and $\Delta(c, s) = \sum_{i=1}^{n'} \Delta(u_i, t_i) + \sum_{i=1}^{n'-1} \Delta(y_i, x_i)$.

With our canonical mapping, we can identify $\mathrm{LCS}(c, s)$ many characters in s with characters in c. Intuitively, these are the characters that are uncorrupted when we transform c into s using insertions and deletions. Call a received buffer y_i in s a *good decoding buffer* (or *good buffer* for short) if at least $\frac{3}{4}\delta m$ of its characters are identified with characters from a single chunk of δm 0s in c. Call a decoding buffer *bad* otherwise. Call a chunk of δm 0s in c *good* if at least $\frac{3}{4}\delta m$ of its zeros map to characters in single decoding buffer. Note that there is a natural bijection between good chunks in c and good decoding buffers in s.

Lemma III.5. *The number of bad decoding buffers of s is at most $8\sqrt{\epsilon}n$.*

Proof. Suppose we have a bad buffer y_i. It either contains characters from at least two different chunks of δm 0s in c or contains at most $\frac{3\delta m}{4}$ characters from a single chunk.

In the first case, x_i must contain characters in two different chunks so its length must be at least m, so y_i must have been obtained from at least $m - \delta m > \delta m > 40\sqrt{\epsilon}m$ deletions from x_i.

In the second case, if x_i has length at most $\frac{7\delta m}{8}$ then the insertion/deletion distance between x_i and y_i is at least $\frac{\delta m}{8} = 5\sqrt{\epsilon}m$. Otherwise, x_i has at least $\frac{\delta m}{8}$ charaters in some inner codeword of c, so x_i has at least $\frac{\delta m}{80}$ 1s, so we need at least $\frac{\delta m}{80} - \frac{\delta m}{160} = \frac{1}{4}\sqrt{\epsilon}m$ deletions to obtain y_i from x_i.

By a simple counting argument, the total number of bad buffers we can have is at most $\frac{2\epsilon mn}{\frac{1}{4}\sqrt{\epsilon}m} = 8\sqrt{\epsilon}n$. □

Lemma III.6. *The number of good decoding buffers of s is at least $(1 - 8\sqrt{\epsilon})n$.*

Proof. It suffices to prove the number of good chunks of c is at least $(1 - 8\sqrt{\epsilon})n$. If a chunk is not mapped to a good buffer, at least one of the following is true.

1) The chunk is "deleted" by inserting enough 1s.
2) Part of the chunk is mapped to a bad buffer that contains characters from $t - 1 \geq 1$ other chunks.
3) Part of the chunk is mapped to a bad buffer that contains no characters from other chunks.

In the first case, we need at least $\frac{\delta m}{160} = \frac{1}{4}\sqrt{\epsilon}m$ insertions to delete the chunk. In the second case, creating the bad buffer costs at least $(t-1)(m-\delta m) \geq \frac{t\delta m}{2}$ deletions, which is at least $20\sqrt{\epsilon}m$ deletions per chunk. In the third case, creating the bad buffer costs at least $\frac{1}{4}\sqrt{\epsilon}m$ edits by the argument in Lemma III.5. Thus, we have at most $\frac{2\epsilon mn}{\frac{1}{4}\sqrt{\epsilon}m} = 8\sqrt{\epsilon}n$ bad chunks, so we have at least $(1 - 8\sqrt{\epsilon})n$ good chunks, as desired. □

Since there are at least $(1-8\sqrt{\epsilon})n$ good decoding buffers and at most $8\sqrt{\epsilon}n$ bad decoding buffers, there must be at least $(1-16\sqrt{\epsilon})n$ pairs of consecutive good decoding buffers. For any pair of consecutive good decoding buffers y_{j-1}, y_j in s, the corresponding two good chunks of δm 0s in c are consecutive unless there is at least one bad chunk in between the two good chunks, which happens for at most $8\sqrt{\epsilon}n$ pairs. Thus, there are at least $(1-24\sqrt{\epsilon})n$ pairs of consecutive good decoding buffers in s such that the corresponding good chunks of 0s in c are also consecutive.

Now suppose w is an inner codeword between two good chunks with corresponding consecutive good decoding buffers, y_{j-1}, y_j. The corresponding decoding window between the decoding buffers is u_j, mapped to from t_j, a substring of c. We claim that most such w are decoded correctly.

For all but $2\frac{2\epsilon mn}{\delta m/8}+2\frac{2\epsilon mn}{\delta m/8}+\frac{2\epsilon mn}{\delta m/4} < 2\sqrt{\epsilon}n$ choices of j, we have $\Delta(x_{j-1}, y_{j-1}) \le \frac{\delta m}{8}$, $\Delta(x_j, y_j) \le \frac{\delta m}{8}$, and $\Delta(t_j, u_j) \le \frac{\delta m}{4}$. When we have an inner codeword w and an index j such that all these are true, we have $|x_{j-1}|, |x_j| \le \frac{9\delta m}{8}$, and each of x_{j-1}, x_j shares at least $\frac{3\delta m}{4}$ characters with one of the chunks of δm 0s neighboring w. It follows that x_{j-1}, x_j each contain at most $\frac{3\delta m}{8}$ characters of w. Additionally, by the definition of a good chunk, u_j contains at most $\frac{\delta m}{4}$ characters in each of the chunks neighboring w. Thus, we have $\Delta(w, t_j) \le \frac{3\delta m}{4}$, in which case, $\Delta(w, t_j) \le \Delta(w, t_j) + \Delta(t_j, u_j) \le \delta m$. Thus, for at least $(1-24\sqrt{\epsilon})n - 2\sqrt{\epsilon}n = (1-26\sqrt{\epsilon})n$ inner words w, there exists $j \in \{1, \ldots, n'\}$ such that $\Delta(w, u_j) \le \delta m$.

Therefore, our algorithm detects at least $(1-26\sqrt{\epsilon})n$ correct pairs (i, r_i). Since our algorithm detects at most $(1+8\sqrt{\epsilon})n$ pairs total, we have at most $34\sqrt{\epsilon}n$ incorrect pairs. Thus, after removing conflicts, we have at least $(1-60\sqrt{\epsilon})n$ correct values, so our Reed Solomon decoder will succeed. □

Remark. Our decoding algorithm succeeds as long as the inner code can correct up to a δ fraction of insertions/deletions and consists of codewords such that every interval of length $\delta m/16$ has at least $1/10$ fraction of 1s. The time complexity of Theorem III.2 can be improved using a more efficient inner code, at the cost of reduction in rate.

Because of the addition of buffers, the code of Theorem III.2 may not be dense enough to use as an inner code. The inner code needs to have $1/10$ fraction of 1s for every interval of length $\delta m/16$. However, we can modify the construction of the inner concatenated code so that the inner codewords of the inner code in Theorem III.2 have at least $1/5$ fraction of 1s in every interval of length $\delta m/16$. This guarantees that the inner codewords of our two level construction have sufficiently high densities of 1s. This is summarized in the following theorem.

Theorem III.7. *There exists a constant $\epsilon_0 > 0$ such that the following holds. Let $\epsilon_0 > \epsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ that is decodable from an ϵ fraction of insertions/deletions with rate $1-\tilde{O}(\sqrt[4]{\epsilon})$ in time $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\epsilon)}$.*

IV. High Noise

Because our decoding algorithms for the $1-\epsilon$ and $1-\frac{2}{k+1}-\epsilon$ insertion/deletion constructions use the same list decoding technique, we abstract out the technical part of the decoding algorithm with the following theorem.

Theorem IV.1. *Let C be a code over alphabet of size k and length $N = nm$ obtained by concatenating a Reed-Solomon C_{out} of length n with an inner code C_{in} of length m. Suppose C_{out} has rate r and is over $\mathbb{F}_q$ with $n = q$. Suppose $C_{in} : [n] \times \mathbb{F}_q \to [k]^m$ can correct a $1-\delta$ fraction of insertions and deletions in $O(t(n))$ for some function t. Then, provided C is (combinatorially) decodable under up to $1-\delta-4r^{1/4}$ fraction of insertions and deletions, it is in fact decodable in time $O(N^3 \cdot (t(N) + \mathrm{polylog}\, N))$.*

Proof. Let $\gamma = 4r^{1/4}$. Consider the following algorithm, which takes as input a string s that is the result of changing a codeword c under a fraction $\le (1-\delta-\gamma)$ of insertions/deletions.

1) $\mathcal{J} \leftarrow \emptyset$.

2) For each $0 \le j \le \lceil \frac{2n}{\gamma} \rceil$ $1 \le j' \le \lceil \frac{4}{\gamma} \rceil$, do the following.

a) Let $\sigma_{j,j'}$ denote the substring from indices $\frac{\gamma m}{2} j$ to $\frac{\gamma m}{2}(j + j')$.

b) By brute force search over $\mathbb{F}_q \times \mathbb{F}_q$, find all pairs (α, β) such that $\Delta(\mathrm{Enc}_{C_{in}}((\alpha, \beta)), \sigma_{j,j'}) \le (1-\delta)m$. If exactly one such pair (α, β) exists, then add (α, β) to $\mathcal{J}$.

3) Find the list, call it $\mathcal{L}$, of all polynomials $p \in \mathbb{F}_q[X]$ of degree less than rn such that $|\{(\alpha, p(\alpha)) | \alpha \in \mathbb{F}_q\} \cap \mathcal{J}| \ge \frac{\gamma n}{2}$.

4) Find the unique polynomial in $\mathcal{L}$, if any, such that the insertion/deletion distance between its encoding under C and s is at most $(1-\gamma-\delta)N$.

Correctness. Break the codeword $c \in [k]^{nm}$ of the concatenated code C into n inner blocks, with the ith block $b_i \in [k]^m$ corresponding to the inner encoding of the ith symbol $(\alpha_i, f(\alpha_i))$ of the outer Reed-Solomon code. For some fixed canonical way of forming s out of c, let s_i be the block formed out of b_i, so that $s_1, \ldots, s_n$ partition the string s. Call an index i *good* if it can be obtained from b_i by at most $(1-\delta-\frac{\gamma}{2})m$ insertions or deletions, and *bad* otherwise. The number of bad indices is at most $\frac{(1-\delta-\gamma)mn}{(1-\delta-\gamma/2)m} \le (1-\frac{\gamma}{2})n$, so the number of good indices is at least $\frac{\gamma n}{2}$.

For any good index a, there exists some $\sigma_{j,j'}$ such that s_a is a substring of $\sigma_{j,j'}$ and $0 < |\sigma_{j,j'}| - |s_a| < \frac{\gamma m}{2}$. Since a is good, the insertion/deletion distance between b_a and s_a is at most $(1-\delta-\gamma/2)m$, and the insertion/deletion distance between s_a and $\sigma_{j,j'}$ is less than $\gamma m/2$, so the insertion/deletion distance between b_a and $\sigma_{j,j'}$ is at most $(1-\delta)m$. Since C_{in} can handle up to $(1-\delta)m$ insertions and deletions, it follows that b_a is the unique codeword of C_{in} such that $\Delta(\mathrm{Enc}_{C_{in}}(b_a), \sigma_{j,j'}) \le (1-\delta)m$. Since b_a is the encoding of $(\alpha_a, f(\alpha_a))$ under C_{in}, we conclude that for any good index a, the pair $(\alpha_a, f(\alpha_a))$ will be included in $\mathcal{J}$. In particular, $\mathcal{J}$ will have at least $\gamma n/2$ such pairs, so the correct f will be in $\mathcal{L}$.

We now check that step 3 of the algorithm will succeed. We have $|\mathcal{J}| \leq \frac{2n}{\gamma} \cdot \frac{4}{\gamma} = \frac{8n}{\gamma^2}$, and Sudan's list decoding algorithm will give a list of degree-less-than-rn polynomials over $\mathbb{F}_q$ such that $(\alpha, p(\alpha)) \in \mathcal{J}$ for more than $\sqrt{2(rn)|\mathcal{J}|}$ values of $\alpha \in \mathbb{F}_q$ [5]. Furthermore, this list will have at most $\sqrt{2|\mathcal{J}|/(rn)}$ elements. For our choice of γ, we have $\gamma n/2 > \sqrt{\frac{16rn^2}{\gamma^2}} \geq \sqrt{2(rn)|\mathcal{J}|}$, so the list decoding will succeed.

By above, there will be at least one polynomial in $\mathcal{L}$ such that the longest common subsequence of its encoding with s has length at least $(\gamma + \delta)m$, namely the correct polynomial f. Since we assumed C can decode up to a $1 - \delta - \gamma$ fraction of insertons/deletions, all other polynomials in $\mathcal{L}$ will have longest common subsequence with s smaller than $(\gamma + \delta)m$. Thus our algorithm returns the correct f.

RUNTIME. We have $O(n) \leq O(N)$ intervals $\sigma_{j,j'}$ to check, and each one brute forces over n^2 terms of $\mathbb{F}_q \times \mathbb{F}_q$. Encoding takes time $O(t(n)) \leq O(t(N))$ by assumption and computing the longest common subsequence takes $O(m^2) = O(\log^2 N)$ time, so in total the second step of the algorithm takes $O(N^3(t(N) + \log^2 N))$ time. Since $|\mathcal{J}| \leq O(N)$ for sufficiently large N, the Reed-Solomon list decoding algorithm can be performed in time $O(N^2)$, see for instance [6]. There are a constant number of polynomials to check at the end, and each one takes $O(N^2)$ time using the longest common subsequence algorithm. Thus, the overall runtime of the algorithm is $O(N^3(t(N) + \text{polylog}\, N))$. □

A. Decoding against $1 - \epsilon$ insertions/deletions

Lemma IV.2. *Suppose have a code C which is the concatenation of an outer code C_{out} of length n with an inner code C_{in} of length m. Suppose further that for some $\Delta, \delta \in (0, 1)$, we have $\text{LCS}(C_{out}) \leq \Delta n, \text{LCS}(C_{in}) \leq \delta m$. Then $\text{LCS}(C) \leq (\Delta + 2\delta)nm$.*

Lemma IV.3 ($\theta = 1/3$ case of Corollary 2.6 of [3])**.** *Let $1/2 > \epsilon > 0$, and k be a positive integer. For every m, there exists a code $C \subseteq [k]^m$ of rate $R = \epsilon/3$ that can correct a $1 - \epsilon$ fraction of insertions/deletions in time $k^{O(m)}$, provided $k \geq 64/\epsilon^3$.*

Theorem IV.4. *For any $\epsilon > 0$, there exists a family of codes over an alphabet of size $\text{poly}(1/\epsilon)$ and rate $\Omega(\epsilon^5)$ that can be efficiently decoded from a $1 - \epsilon$ fraction of insertions/deletions. Furthermore, this code is constructible, encodable, and decodable in time $N^{\text{poly}(1/\epsilon)}$.*

Proof. Let $n = q, m = 24 \log q/\epsilon$, and $k = O(1/\epsilon^3)$. By Lemma IV.3, we can construct by brute force a code $C_1 : n \times \mathbb{F}_q \to [k]^m$ that can be decoded from $1 - \epsilon/4$ fraction of worst-case insertions and deletions. We can concatenate C_1 with an outer Reed-Solomon code of rate $(\epsilon/8)^4$.

The rate of the inner code is $\Omega(\epsilon)$, and the rate of the outer code is $\Omega(\epsilon^4)$, so the total rate is $\Omega(\epsilon^5)$.

By Lemma IV.2, $\text{LCS}(C) \leq (\epsilon/8)^4 + 2(\epsilon/4) < \epsilon$, so C is capable of decoding up to $1 - \epsilon$ fraction of insertions and deletions. Encoding in C_1 is done by brute force in time $N^{\text{poly}(1/\epsilon)}$, so by Theorem IV.1, C is capable of decoding up to $1 - \epsilon/4 - 4((\epsilon/8)^4)^{1/4} > 1 - \epsilon$ fraction of worst-case insertions and deletions in time $O(N^3(N^{\text{poly}(1/\epsilon)} + \text{poly} \log N)) = N^{\text{poly}(1/\epsilon)}$, as desired. □

Remark. Our construction only requires that the inner code can be decoded from $1 - \epsilon/4$ fraction of worst-case insertions and deletions. By using the concatenated code of Theorem IV.4 as the inner code of the same construction (thus giving us two levels of concatenation), we can reduce the time complexity significantly, at the cost of a polynomial reduction in other parameters of the code, as summarized below.

Theorem IV.5. *For any $\epsilon > 0$, there exists a family of constant rate codes over an alphabet of size $\text{poly}(1/\epsilon)$ and rate $\Omega(\epsilon^9)$ that can be decoded from a $1 - \epsilon$ fraction of insertions/deletions. Furthermore, this code is constructible, encodable, and decodable in time $\text{poly}(N) \cdot (\log N)^{\text{poly}(1/\epsilon)}$.*

B. Decoding against $1 - \frac{2}{k+1} - \epsilon$ insertions/deletions

First, we sumarize an existence result from [2].

Lemma IV.6 (Theorem 12 of [2])**.** *Fix an integer $k \geq 2$ and $\gamma > 0$. Then there are infinitely many N for which there is a concatenated Reed Solomon code $C \subseteq [k]^N$ that has outer rate at least $\gamma/2$, has total rate at least $(\gamma/k)^{O(\gamma^{-3})}$, is decodable under $1 - \frac{2}{k+1} - \gamma$ fraction of insertions and deletions, has an inner code decodable under $1 - \frac{2}{k+1} - \gamma/4$ insertions and deletions, and is constructible in time $O(N \log^2 N)$.*

Theorem IV.7. *Fix an integer $k \geq 2$ and $\epsilon > 0$. For infinitely many and sufficiently large N, there is an explicit code $C \subseteq \{0, 1\}^N$ with rate $r(k, \epsilon) = (\epsilon/k)^{O(\epsilon^{-12})}$ over a size k alphabet that can be decoded from a $1 - \frac{2}{k+1} - \epsilon$ fraction of worst-case insertions and deletions in time $O_{k,\epsilon}(N^3\, \text{polylog}(N))$. Furthermore, this code is constructible in time $O_{k,\epsilon}(N \log^2 N)$.*

Proof. Consider the codes C given by Lemma IV.6 with $\gamma = 2(\epsilon/5)^4$. C has outer rate at least $\gamma/2 = (\epsilon/5)^4$ and total rate at least $(\gamma/k)^{O(\gamma^{-3})}$. Furthermore, C can decode up to $1 - \frac{2}{k+1} - \gamma$ fraction of insertions/deletions, and the inner code of C can decode $1 - \frac{2}{k+1} - \gamma/4$ fraction of insertions/deletions. Thus, by Theorem IV.1, C can efficiently decode up to $1 - \frac{2}{k+1} - \gamma/4 - 4(\gamma/2)^{1/4} > 1 - \frac{2}{k+1} - \epsilon$ fraction of insertions/deletions. □

REFERENCES

[1] L. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Trans. Inform. Theory*, 45(7):2552–2557, 1999.

[2] B. Bukh and V. Guruswami. An improved bound on the fraction of correctable deletions. *Proc. of SODA*, pages 1893-1901, 2016.

[3] V. Guruswami and C. Wang. Deletion codes in the high-noise and high-rate regimes. *Proceeding of RANDOM*, pages 867-880, 2015.

[4] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Dokl. Akad. Nauk*, 163(4):845–848, 1965. English translation in Soviet Physics Doklady, 10(8):707-710, 1966.

[5] M. Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.

[6] R. M. Roth and G. Ruckenstein. Efficient decoding of reed-solomon codes beyond half the minimum distance. *IEEE Trans. Inform. Theory*, 46(1):246–257, 2000.