Deletion Codes in the High-noise and High-rate Regimes*

Venkatesan Guruswami and Carol Wang

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA
`guruswami@cmu.edu, wangc@cs.cmu.edu`

Abstract

The noise model of deletions poses significant challenges in coding theory, with basic questions like the capacity of the binary deletion channel still being open. In this paper, we study the harder model of *worst-case* deletions, with a focus on constructing efficiently encodable and decodable codes for the two extreme regimes of high-noise and high-rate. Specifically, we construct polynomial-time decodable codes with the following trade-offs (for any $\varepsilon > 0$):

(i) Codes that can correct a fraction $1 - \varepsilon$ of deletions with rate $\mathrm{poly}(\varepsilon)$ over an alphabet of size $\mathrm{poly}(1/\varepsilon)$;
(ii) Binary codes of rate $1 - \tilde{O}(\sqrt{\varepsilon})$ that can correct a fraction ε of deletions; and
(iii) Binary codes that can be *list decoded* from a fraction $(1/2 - \varepsilon)$ of deletions with rate $\mathrm{poly}(\varepsilon)$.

Our work is the first to achieve the qualitative goals of correcting a deletion fraction approaching 1 over bounded alphabets, and correcting a constant fraction of bit deletions with rate approaching 1 over a fixed alphabet. The above results bring our understanding of deletion code constructions in these regimes to a similar level as worst-case errors.

1998 ACM Subject Classification E.4 Coding and Information Theory

Keywords and phrases algorithmic coding theory, deletion codes, list decoding, probabilistic method, explicit constructions

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.867

1 Introduction

This work addresses the problem of constructing codes which can be efficiently corrected from a constant fraction of worst-case deletions. In contrast to erasures, the locations of deleted symbols are *not* known to the decoder, who receives only a subsequence of the original codeword. The deletions can be thought of as corresponding to errors in synchronization during communication. The loss of position information makes deletions a very challenging model to cope with, and our understanding of the power and limitations of codes in this model significantly lags behind what is known for worst-case errors.

The problem of communicating over the *binary deletion channel*, in which each transmitted bit is deleted independently with a fixed probability p, has been a subject of much study (see the excellent survey by Mitzenmacher [17] for more background and references). However, even this easier case is not well-understood. In particular, the capacity of the binary deletion channel remains open, although it is known to approach $1 - h(p)$ as p goes to 0, where $h(p)$

* Research supported in part by NSF grants CCF-0963975 and CCF-1422045. The second author is supported by an NSF Graduate Research Fellowship.

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) /
19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 867–880
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

is the binary entropy function (see [5, 6, 25] for lower bounds and [12, 13] for upper bounds), and it is known to be positive (at least $(1-p)/9$) [18]) even as $p \to 1$.

The more difficult problem of correcting from adversarial rather than random deletions has also been studied, but with a focus on correcting a constant *number* (rather than fraction) of deletions [16]. Codes that can correct a single deletion have received a fair bit of attention (see the survey [23]), but it turns out that even correcting two deletions poses significant challenges and is not well understood, with efficient codes with low redundancy discovered only very recently [2].

Coding for a constant *fraction* of adversarial deletions, which is the focus of this work, has been considered previously by Schulman and Zuckerman [21]. They construct constant-rate binary codes which are efficiently decodable from a small constant fraction of worst-case deletions and insertions, and can also handle a small fraction of transpositions. The rate of these codes are bounded away from 1, whereas existentially one can hope to achieve a rate approaching 1 for a small deletion fraction.

The central theoretical goal in error-correction against any specific noise model is to understand the combinatorial trade-off between the rate of the code and noise rate that can be corrected, and to construct codes with efficient error-correction algorithms that ideally approach this optimal trade-off. While this challenge is open in general even for the well-studied and simpler model of errors and erasures, in the case of worst-case deletions, our knowledge has even larger gaps. (For instance, we do not know the largest deletion fraction which can be corrected with positive rate for any fixed alphabet size.) Over large alphabets that can grow with the length of the code, we can include the position of each codeword symbol as a header that is part of the symbol. This reduces the model of deletions to that of erasures, where simple optimal constructions (eg. Reed-Solomon codes) are known.

Given that we are far from an understanding of the best rate achievable for any specified deletion fraction, in this work we focus on the two extreme regimes — when the deletion fraction is small (and the code rate can be high), and when the deletion fraction approaches the maximum tolerable value (and the code rate is small). Our emphasis is on constructing codes that can be efficiently encoded and decoded, with trade-offs not much worse than random/inefficient codes (whose parameters we compute in Section 2). Our results, described next, bring the level of knowledge on efficient deletion codes in these regimes to a roughly similar level as worst-case errors. There are numerous open questions, both combinatorial and algorithmic, that remain open, and it is our hope that the systematic study of codes for worst-case deletions undertaken in this work will spur further research on good constructions beyond the extremes of low-noise and high-noise.

1.1 Our results

The best achievable rate against a fraction p of deletions cannot exceed $1-p$, as we need to be able to recover the message from the first $(1-p)$ fraction of codeword symbols. As mentioned above, over large (growing) alphabets this trade-off can in fact be achieved by a simple reduction to the model of erasures. Existentially, as we show in Section 2, for any $\gamma > 0$, it is easy to show that there are codes of rate $1-p-\gamma$ to correct a fraction p of deletions over an alphabet size that depends only on γ. For the weaker model of erasures, where the receiver knows the locations of erased symbols, we know explicit codes, namely certain algebraic-geometric codes [22] or expander based constructions [1, 8], achieving the optimal trade-off (rate $1-p-\gamma$ to correct a fraction p of erasures) over alphabets growing only as a function of $1/\gamma$. For deletions, we do not know how to construct codes with such a trade-off efficiently. However, in the high-noise regime when the deletion fraction is $p = 1-\varepsilon$

for some small $\varepsilon > 0$, we are able to construct codes of rate $\mathrm{poly}(\varepsilon)$ over an alphabet of size $\mathrm{poly}(1/\varepsilon)$. Note that an alphabet of size at least $1/\varepsilon$ is needed, and the rate can be at most ε, even for the simpler model of erasures, so we are off only by polynomial factors.

▶ **Theorem** (Theorem 7). *Let $1/2 > \varepsilon > 0$. There is an explicit code of rate $\Omega(\varepsilon^2)$ and alphabet size $\mathrm{poly}(1/\varepsilon)$ which can be corrected from a $1-\varepsilon$ fraction of worst-case deletions.*

Moreover, this code can be constructed, encoded, and decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$, where N is the block length of the code.

The above handles the case of very large fraction of deletions. At the other extreme, when the deletion fraction is small, the following result shows that we achieve high rate (approaching one) even over the binary alphabet.

▶ **Theorem** (Theorem 11). *Let $\varepsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ which is decodable from an ε fraction of deletions with rate $1-\tilde{O}(\sqrt{\varepsilon})$ in time $N^{\mathrm{poly}(1/\varepsilon)}$.*

Moreover, C can be constructed and encoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

▶ Remark. For both of the above results, the construction and encoding/decoding complexity can be improved to $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$ at the expense of slightly worse parameters. See Theorems 16 and 10.

The next question is motivated by constructing binary codes for the "high noise" regime. In this case, we do not know (even non-constructively) the minimum fraction of deletions that forces the rate of the code to approach zero. (Contrast this with the situation for erasures (resp. errors), where we know the zero-rate threshold to be an erasure fraction $1/2$ (resp. error fraction $1/4$).) Clearly, if the adversary can delete half of the bits, he can always ensure that the decoder receives $0^{n/2}$ or $1^{n/2}$, so at most two strings can be communicated. Surprisingly, in the model of *list decoding*, where the decoder is allowed to output a small list consisting of all codewords which contain the received string as a subsequence, one can in fact decode from an deletion fraction arbitrarily close to $1/2$, as our third construction shows:

▶ **Theorem** (Theorem 19). *Let $0 < \varepsilon < 1/2$. There is an explicit binary code $C \subseteq \{0,1\}^N$ of rate $\tilde{\Omega}(\varepsilon^3)$ which is list-decodable from a $1/2-\varepsilon$ fraction of deletions with list size $(1/\varepsilon)^{O(\log\log(1/\varepsilon))}$.*

This code can be constructed, encoded, and list-decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

We should note that it is not known if list decoding is required to correct deletion fractions close to $1/2$, or if one can get by with unique decoding. Our guess would be that the largest deletion fraction unique decodable with binary codes is bounded away from $1/2$. The cubic dependence on ε in the rate in the above theorem is similar to what has been achieved for correcting $1/2-\varepsilon$ fraction of errors [9]. We anticipate (but have not formally checked) that a similar result holds over any fixed alphabet size k for list decoding from a fraction $(1-1/k-\varepsilon)$ of symbol deletions.

Construction approach

Our codes, like many considered in the past, including those of [3, 4, 20] in the random setting and particularly [21] in the adversarial setting, are based on concatenating a good error-correcting code (in our case, Reed-Solomon or Parvaresh-Vardy codes) with an inner deletion code over a much smaller block length. This smaller block length allows us to find and decode the inner code using brute force. The core of the analysis lies in showing that the adversary can only affect the decoding of a bounded fraction of blocks of the inner code, allowing the outer code to decode using the remaining blocks.

While our proofs only rely on elementary combinatorial arguments, some care is needed to execute them without losing in rate (in the case of Theorem 11) or in the deletion fraction we can handle (in the case of Theorems 7 and 19). In particular, for handling close to fraction 1 of deletions, we have to carefully account for errors and erasures of outer Reed-Solomon symbols caused by the inner decoder. To get binary codes of rate approaching 1, we separate inner codeword blocks with (not too long) buffers of 0's and we exploit some additional structural properties of inner codewords that necessitate many deletions to make them resemble buffers. The difficulty in both these results is unique identification of enough inner codeword boundaries so that the Reed-Solomon decoder will find the correct message. The list decoding result is easier to establish, as we can try many different boundaries and use a "list recovery" algorithm for the outer algebraic code. To optimize the rate, we use the Parvaresh-Vardy codes [19] as the outer algebraic code.

1.2 Organization

In Section 2, we consider the performance of certain random and greedily constructed codes. These serve both as benchmarks and as starting points for our efficient constructions. In Section 3, we construct codes in the high deletion regime. In Section 4, we give high-rate binary codes which can correct a small constant fraction of deletions. In Section 5, we give list-decodable binary codes up to the optimal error fraction. Some open problems appear in Section 6.

2 Existential bounds for deletion codes

A quick recap of standard coding terminology: a code C of block length m over an alphabet Σ is a subset $C \subseteq \Sigma^m$. The rate of C is defined as $\frac{\log |C|}{m \log |\Sigma|}$. The encoding function of a code is a map $E : [|C|] \to \Sigma^m$ whose image equals C (with messages identified with $[|C|]$ in some canonical way). Our constructions all exploit the simple but powerful idea of code concatenation: If $C_{\text{out}} \subseteq \Sigma_{\text{out}}^n$ is an "outer" code with encoding function E_{out}, and $C_{\text{in}} \subseteq \Sigma_{\text{in}}^m$ is an "inner" code encoding function $E_{\text{in}} : \Sigma_{\text{out}} \to \Sigma_{\text{in}}^m$, the the concatenated code $C_{\text{out}} \circ C_{\text{in}} \subseteq \Sigma_{\text{in}}^{nm}$ is a code whose encoding function first applies E_{out} to the message, and then applies E_{in} to each symbol of the resulting outer codeword.

In this section, we show the existence of deletion codes in certain ranges of parameters, without the requirement of efficient encoding or decoding. The proofs (found in the full version of this paper [11]) follow from standard probabilistic arguments, but to the best of our knowledge, these bounds were not known previously. The codes of Theorem 4 will be used as inner codes in our final concatenated constructions.

Throughout, we will write $[k]$ for the set $\{1, \ldots, k\}$. We will also use the binary entropy function, defined for $\delta \in [0, 1]$ as $h(\delta) = \delta \log \frac{1}{\delta} + (1 - \delta) \log \frac{1}{1-\delta}$. All logarithms in the paper are to base 2.

We note that constructing a large code in $[k]^m$ which can correct from a δ fraction of deletions is equivalent to constructing a large set of strings such that for each pair, their longest common subsequence (LCS) has length less than $(1 - \delta)m$.

We first consider how well a random code performs, using the following theorem from [15], which upper bounds the probability that a pair of randomly chosen strings has a long LCS.

▶ **Theorem 1** ([15], Theorem 1). *For every $\gamma > 0$, there exists $c > 0$ such that if k and $m/\sqrt{k}$ are sufficiently large, and u, v are chosen independently and uniformly from $[k]^m$, then*

$$\Pr\left[\left|\mathrm{LCS}(u,v) - 2m/\sqrt{k}\right| \geq \frac{\gamma m}{\sqrt{k}}\right] \leq e^{-cm/\sqrt{k}}.$$

Fixing γ to be 1, we obtain the following.

▶ **Proposition 2.** *Let $\varepsilon > 0$ be sufficiently small and let $k = (4/\varepsilon)^2$. There exists a code $C \subseteq [k]^m$ of rate $R = \Omega\big(\varepsilon/\log(1/\varepsilon)\big)$ which can correct a $1 - \varepsilon = 1 - 4/\sqrt{k}$ fraction of deletions.*

The following results, and in particular Corollary 6, show that we can nearly match the performance of random codes using a simple greedy algorithm.

We first bound the number of strings which can have a fixed string s as a subsequence.

▶ **Lemma 3.** *Let $\delta \in (0, 1/k)$, set $\ell = (1-\delta)m$, and let $s \in [k]^\ell$. The number of strings $s' \in [k]^m$ containing s as a subsequence is at most*

$$\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} k^{m-t}(k-1)^{t-\ell} \leq k^{m-\ell}\binom{m}{\ell}.$$

When $k = 2$, we have the estimate

$$\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} 2^{m-t} \leq \delta m \binom{m}{\ell}.$$

▶ **Theorem 4.** *Let $\delta, \gamma > 0$. Then for every m, there exists a code $C \subseteq [k]^m$ of rate $R = 1 - \delta - \gamma$ such that:*
- *C can be corrected from a δ fraction of worst-case deletions, provided $k \geq 2^{2h(\delta)/\gamma}$.*
- *C can be found, encoded, and decoded in time $k^{O(m)}$.*

Moreover, when $k = 2$, we may take $R = 1 - 2h(\delta) - \log(\delta m)/m$.

▶ Remark. The authors of [14] show a similar result for the binary case, but use the weaker bound in Lemma 3 to get a rate of $1 - \delta - 2h(\delta)$.

With a slight modification to the proof of Theorem 4, we obtain the following construction, which will be used in Section 4. The so-called "β-dense" property will help us to distinguish codewords, which have high Hamming weight, from long strings of zeroes.

▶ **Proposition 5.** *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 $s \in C$, s is "β-dense": every interval of length βm in s contains at least $\beta m/10$ ones,*
- *C can be corrected from a δ fraction of worst-case deletions, and*
- *C can be found, encoded, and decoded in time $2^{O(m)}$.*

In the high-deletion regime, we have the following corollary to Theorem 4, obtained by setting $\delta = 1 - \varepsilon$ and $\gamma = (1-\theta)\varepsilon$, and noting that $h(\varepsilon) \leq \varepsilon \log(1/\varepsilon) + 2\varepsilon$ when $\varepsilon < 1/2$.

▶ **Corollary 6.** *Let $1/2 > \varepsilon > 0$ and $\theta \in (0, 1/3]$. There for every m, there exists a code $C \subseteq [k]^m$ of rate $R = \varepsilon \cdot \theta$ which can correct a $1 - \varepsilon$ fraction of deletions in time $k^{O(m)}$, provided $k \geq 64/\varepsilon^{\frac{2}{1-\theta}}$.*

3 Coding against $1 - \varepsilon$ deletions

In this section, we construct codes for the high-deletion regime. We will use a concatenated coding approach, with an enlarged alphabet to help us determine the location of inner codewords. By choosing the parameters carefully, we are able to correct a large fraction of deletions. More precisely, we have the following theorem.

▶ **Theorem 7.** *Let $1/2 > \varepsilon > 0$. There is an explicit code of rate $\Omega(\varepsilon^2)$ and alphabet size $\mathrm{poly}(1/\varepsilon)$ which can be corrected from a $1-\varepsilon$ fraction of worst-case deletions.*

Moreover, this code can be constructed, encoded, and decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$, where N is the block length of the code.

We first define the code. Theorem 7 is then a direct corollary of Lemmas 8 and 9.

The code: Our code will be over the alphabet $\{0, 1, \ldots, D-1\} \times [k]$, where $D = 8/\varepsilon$ and $k = O(1/\varepsilon^3)$.

We first define a code C' over the alphabet $[k]$ by concatenating a Reed-Solomon code with an inner code over $[k]$ which can correct a slightly higher fraction of deletions.

More specifically, let $\mathbb{F}_q$ be a finite field. For any $n' \leq n \leq q$, the Reed-Solomon code of length $n \leq q$ and dimension n' is a subset of $\mathbb{F}_q^n$ which admits an efficient algorithm to uniquely decode from t errors and r erasures, provided $r + 2t < n - n'$ (see, for example, [24]).

In our construction, we will take $n = q = 2n'/\varepsilon$. We first encode our message to a codeword $c = (c_1, \ldots, c_n)$ of the Reed-Solomon code. For each i, we then encode the pair (i, c_i) using an inner code over some alphabet $[k]$ which can correct a $1 - \varepsilon/2$ fraction of deletions.

To obtain our final code C, we replace every symbol s in C' which encodes the ith RS coordinate by the pair $\big(i \pmod D, s\big) \in \{0, 1, \ldots, D-1\} \times [k]$. The first coordinate, $i \pmod D$, contains the location of the codeword symbol modulo D, and we will refer to it as a **header**.

In order to obtain the parameters stated in Theorem 7, we will instantiate the inner code using Corollary 6, setting $\theta = 1/3$. This gives an inner code $C_1 \colon [n] \times \mathbb{F}_q \to [k]^m$, where $m = 12 \log q/\varepsilon$ and $k = O(1/\varepsilon^3)$, which can correct a $1 - \varepsilon/2$ fraction of deletions.

▶ **Lemma 8.** *For an inner code of rate R_{in}, the rate of C is $\Omega(\varepsilon R_{in})$. In particular, the rate of C can be taken to be $\Omega(\varepsilon^2)$.*

Proof. The rate of the outer Reed-Solomon code, labeled with indices, is at least $\varepsilon/4$. Finally, the alphabet increase in transforming C' to C decreases the rate by a factor of $\frac{\log(k)}{\log(Dk)} = \Omega(1)$.

By Corollary 6, the rate of the inner code can be taken to be $\Omega(\varepsilon)$.This gives us a final rate of $\Omega(\varepsilon^2)$. ◀

▶ **Lemma 9.** *Let the inner code have block length m and be decodable from a $1 - \varepsilon/2$ fraction of worst-case deletions in time $T(m)$. Then the concatenated code C can be decoded from a $1 - \varepsilon$ fraction of worst-case deletions in time $\mathrm{poly}(N) \cdot T(m)$, where N is the block length of C.*

In particular, the concatenated code using Corollary 6 can be decoded in time $N^{O(\mathrm{poly}\, 1/\varepsilon)}$.

Proof. We apply the following algorithm to decode C.

- We partition the received word into *blocks* as follows: The first block begins at the first coordinate, and each subsequent block begins at the next coordinate whose header differs from its predecessor. This takes time $\mathrm{poly}(N)$.

- We begin with an empty set L.
  For each block which is of length between $\varepsilon m/2$ and m, we remove the headers by replacing each symbol (a, b) with the second coordinate b. We then apply the decoder from Corollary 6 to the block. If this succeeds, outputting a pair (i, r_i), then we add (i, r_i) to L. This takes time $\mathrm{poly}(N) \cdot T(m)$.
- If for any i, L contains multiple pairs with first coordinate i, we remove all such pairs from L. L thus contains at most one pair (i, r_i) for each index i. We apply the Reed-Solomon decoding algorithm to the string r whose ith coordinate is r_i if $(i, r_i) \in L$ and erased otherwise. This takes time $\mathrm{poly}(N)$.

Analysis: For any i, we will decode a correct coordinate $\big(i, c_i\big)$ if there is a block of length at least $\varepsilon m/2$ which is a subsequence of $C_1(i, c_i)$. (Here and in what follows we abuse notation by disregarding headers on codeword symbols.)

Thus, the Reed-Solomon decoder will receive the correct value of the ith coordinate unless one of the following occurs:

1. (Erasure) The adversary deletes a $\geq 1 - \varepsilon/2$ fraction of $C_1(i, c_i)$.
2. (Merge) The block containing (part of) $C_1(i, c_i)$ also contains symbols from other codewords of C_1, because the adversary has erased the codewords separating $C_1(i, c_i)$ from its neighbors with the same header.
3. (Conflict) Another block decodes to (i, r) for some r. Note that an erasure cannot cause a coordinate to decode incorrectly, so a conflict can only occur from a merge.

We would now like to bound the number of errors and erasures the adversary can cause.

- If the adversary causes an erasure without causing a merge, this requires at least $(1-\varepsilon/2)m$ deletions within the block which is erased, and no other block is affected.
- If the adversary merges t inner codewords with the same label, this requires at least $(t-1)(D-1)m$ deletions, of the intervening codewords with different labels. The merge causes the fully deleted inner codewords to be erased, and causes the t merged codewords to resolve into at most one (possibly incorrect) value. This value, if incorrect, could also cause one conflict.
  In summary, in order to cause one error and $r \leq (t-1)D + 2$ erasures, the adversary must introduce at least $(t-1)(D-1)m \geq (2 + r)(1 - \varepsilon/2)m$ deletions.

In particular, if the adversary causes s errors and r_1 erasures by merging, and r_2 erasures without merging, this requires at least

$$\geq (2s + r_1)(1 - \varepsilon/2)m + r_2(1 - \varepsilon/2)m = (2s + r)(1 - \varepsilon/2)m$$

deletions. Thus, when the adversary deletes at most a $(1 - \varepsilon)$ fraction of codeword symbols, we have that $2s + r$ is at most $(1 - \varepsilon)mn/(1 - \varepsilon/2)m < n(1 - \varepsilon/2)$. Recalling that the Reed-Solomon decoder in the final step will succeed as long as $2s + r < n(1 - \varepsilon/2)$, we conclude that the decoding algorithm will output the correct message. ◀

▶ Remark (Improving the encoding and decoding complexity). Our decoding algorithm requires only that the inner code C_1 be correctable from a $1 - \varepsilon/2$ fraction of deletions. By using the concatenated code of Theorem 7 as the inner code in our construction (that is, with two levels of concatenation), we can reduce the time complexity significantly, at the cost of a polynomial reduction in other parameters of the code. This is summarized in the following theorem.

▶ **Theorem 10.** *Let $1/2 > \varepsilon > 0$. There is an explicit code of rate $\Omega(\varepsilon^3)$ and alphabet size $\mathrm{poly}(1/\varepsilon)$ which can be corrected from a $1-\varepsilon$ fraction of worst-case deletions. Moreover, this code can be constructed, encoded, and decoded in time $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$, where N is the block length of the code.*

4 Binary codes against ε deletions

4.1 Construction overview

The goal in our constructions is to allow the decoder to approximately locate the boundaries between codewords of the inner code, in order to recover the symbols of the outer code. In the previous section, we were able to achieve this by augmenting the alphabet and letting each symbol encode some information about the block to which it belongs. In the binary case, we no longer have this luxury.

The basic idea of our code is to insert long runs of zeros, or "buffers," between adjacent inner codewords. The buffers are long enough that the adversary cannot destroy many of them. If we then choose the inner code to be dense (in the sense of Proposition 5), it is also difficult for a long interval in any codeword to be confused for a buffer. This approach optimizes that of [21], which uses an inner code of rate $1/2$ and thus has final rate bounded away from 1.

The balance of buffer length and inner codeword density seems to make buffered codes unsuited for high deletion fractions, and indeed our results only hold as the deletion fraction goes to zero.

4.2 Our construction

We now give the details of our construction. For simplicity, we will not optimize constants in the analysis.

▶ **Theorem 11.** *Let $\varepsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ which is decodable from an ε fraction of deletions with rate $1 - \tilde{O}(\sqrt{\varepsilon})$ in time $N^{\mathrm{poly}(1/\varepsilon)}$.*

Moreover, C can be constructed and encoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

The code: We again use a concatenated construction with a Reed-Solomon code as the outer code, choosing one which can correct a $12\sqrt{\varepsilon}$ fraction of errors and erasures. For each i, we replace the ith coordinate c_i with the pair (i, c_i). In order to ensure that the rate stays high, we use a RS code over $\mathbb{F}_{q^h}$, with block length $n = q$, where we will take $h = 1/\varepsilon$.

The inner code will be a good binary deletion code C_1 of block length m correcting a $\delta = 40\sqrt{\varepsilon}$ fraction of deletions. We will also require the codewords of C_1 to be β-dense, for $\beta = \delta/4$. Recall that a string of length m is β-dense if any interval of length βm contains at least $\beta m/10$ 1's. We will assume each codeword begins and ends with a 1.

Now, between each pair of adjacent inner codewords of C_1, we insert a *buffer* of δm zeros. This gives us our final code C.

In order to obtain the final parameters stated in Theorem 11, we will construct the inner code C_1 using Proposition 5. This gives a code of rate $1 - 2h(\delta) - o(1)$ satisfying the requirements of our construction.

▶ **Lemma 12.** *For an inner code of rate R_{in}, the rate of the concatenated code C is $R_{in} \cdot (1 - O(\sqrt{\varepsilon})$.*

In particular, the rate of the concatenated code using Proposition 5 is $1 - \tilde{O}(\sqrt{\varepsilon}))$.

Proof. The rate of the outer (labeled) Reed-Solomon code is $(1 - 24\sqrt{\varepsilon}) \cdot \frac{h}{h+1}$. Finally, adding buffers reduces the rate by a factor of $\frac{1}{1+\delta}$.

Combining these with our choice of δ, we get that the rate of C is $R_i(1 - \tilde{O}(\sqrt{\varepsilon}))$.

The rate of the inner code C_1 can be taken to be $1 - 2h(\delta) - o(1)$, by Proposition 5, giving a final rate of $1 - \tilde{O}(\sqrt{\varepsilon})$. ◀

▶ **Lemma 13.** *Let the inner code have block length m and be decodable from a δ fraction of worst-case deletions in time $T(m)$. Then the concatenated code C can be decoded from a ε fraction of worst-case deletions in time* $\mathrm{poly}(N) \cdot T(m)$*, where N is the block length of C.*

In particular, the concatenated code with inner code constructed using Proposition 5 can be decoded in time $N^{O(\mathrm{poly}\,1/\varepsilon)}$.

The algorithm:

- The decoder first locates all runs of at least $\delta m/2$ contiguous zeroes in the received word. These runs ("buffers") are removed, dividing the codeword into blocks of contiguous symbols which we will call *decoding windows*. Any leading zeroes of the first decoding window and trailing zeroes of the last decoding window are also removed. This takes time $\mathrm{poly}(N)$.
- We begin with an empty set L.
  For each decoding window, we apply the decoder from Proposition 5 to attempt to recover a pair (i, r_i). If we succeed, this pair is added to L. This takes time $\mathrm{poly}(N) \cdot T(m)$.
- If for any i, L contains multiple pairs with first coordinate i, we remove all such pairs from L. L thus contains at most one pair (i, r_i) for each index i. We apply the Reed-Solomon decoding algorithm to the string r whose ith coordinate is r_i if $(i, r_i) \in L$ and erased otherwise, attempting to recover from a $12\sqrt{\varepsilon}$ fraction of errors and erasures. This takes time $\mathrm{poly}(N)$.

Analysis: Notice that if no deletions occur, the decoding windows will all be codewords of the inner code C_1, which will be correctly decoded. At a high level, we will show that the adversary cannot corrupt many of these decoding windows, even with an ε fraction of deletions.

We first show that the number of decoding windows considered by our algorithm is close to n, the number of windows if there are no deletions.

▶ **Lemma 14.** *If an ε fraction of deletions have occurred, then the number of decoding windows considered by our algorithm is between $(1 - 2\sqrt{\varepsilon})n$ and $(1 + 2\sqrt{\varepsilon})n$.*

Proof. Recall that the adversary can cause at most $\varepsilon nm(1 + \delta) \leq 2\varepsilon nm$ deletions.

Upper bound: The adversary can increase the number of decoding windows only by creating new runs of $\delta m/2$ zeroes (that are not contained within a buffer). Such a new run must be contained entirely within an inner codeword $w \in C_1$. However, as w is $\delta/4$-dense, in order to create a run of zeroes of length $\delta m/2$, at least $\delta m/20 = 2\sqrt{\varepsilon}$ 1's must be deleted for each such run. In particular, at most $\sqrt{\varepsilon}n$ blocks can be added.

Lower bound: The adversary can decrease the number of decoding windows only by decreasing the number of buffers. He can achieve this either by removing a buffer, or by merging two buffers. Removing a buffer requires deleting $\delta m/2 = 20\sqrt{\varepsilon}m$ zeroes from the original buffer. Merging two buffers requires deleting all 1's in the inner codewords between them. As inner codewords are $\delta/4$-dense, this requires at least $\sqrt{\varepsilon}m$ deletions for each merged buffer. In particular, at most $2\sqrt{\varepsilon}n$ buffers can be removed. ◀

We now show that almost all of the decoding windows being considered are decoded correctly by the inner decoder.

▶ **Lemma 15.** *The number of decoding windows which are incorrectly decoded is at most* $4\sqrt{\varepsilon}n$*.*

Proof. The inner decoder will succeed on each decoding window which is a subsequence of a valid inner codeword $w \in C_1$ of length at least $(1-\delta)m$. This will happen unless:

1. The window is too short:
   - **(a)** a subsequence of w has been marked as a (new) buffer, or
   - **(b)** a ρ fraction of w has been marked as part of the adjacent buffers, combined with a $\delta - \rho$ fraction of deletions within w.
2. The window is not a subsequence of a valid inner codeword: the window contains buffer symbols and/or a subsequence of multiple inner codewords.

We first show that (1) holds for at most $3\sqrt{\varepsilon}n$ windows.

From the proof of Lemma 14, there can be at most $\sqrt{\varepsilon}n$ new buffers introduced, thus handling Case 1(a). In Case 1(b), if $\rho < \delta/2$, then there must be $\delta/2$ deletions within w. On the other hand, if $\rho \geq \delta/2$, one of two buffers adjacent to w must have absorbed at least $\delta m/4$ symbols of w, so as w is $\delta/4$-dense, this requires $\delta m/40 = \sqrt{\varepsilon}m$ deletions, so can occur in at most $2\sqrt{\varepsilon}n$ windows.

We also have that (2) holds for at most $\sqrt{\varepsilon}n$ windows, as at least $\delta m/2$ symbols must be deleted from a buffer in order to prevent the algorithm from marking it as a buffer. As in Lemma 14, this requires $20\sqrt{\varepsilon}$ deletions for each merged window, and so there are at most $\sqrt{\varepsilon n}$ windows satisfying case (2). ◀

We now have that the inner decoder outputs $(1-6\sqrt{\varepsilon})n$ correct values. After removing possible conflicts in the last step of the algorithm, we have at least $(1-12\sqrt{\varepsilon})n$ correct values, so that the Reed-Solomon decoder will succeed and output the correct message.

▶ Remark (Improving the encoding and decoding efficiency). Our decoding algorithm succeeds as long as the inner code can correct a δ fraction of deletions, and consists of codewords which are $\delta/4$-dense. As in the high deletion case, the time complexity of Theorem 11 can be improved using a more efficient inner code, at the cost of a reduction in rate.

Because of the addition of buffers, the code of Theorem 11 may not be dense enough to use as an inner code. However, we can modify the construction to obtain a dense inner code (details can be found in the full version [11]). In particular, these modifications give us the following.

▶ **Theorem 16.** *Let* $\varepsilon > 0$*. There is an explicit binary code* $C \subseteq \{0,1\}^N$ *which is decodable from an* ε *fraction of deletions with rate* $1 - \tilde{O}(\sqrt[4]{\varepsilon})$ *in time* $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$*.*

Moreover, C *can be constructed and encoded in time* $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$*.*

5 List-decoding binary deletion codes

The results of Section 4 show that we can have good explicit binary codes when the deletion fraction is low. In this section, we address the opposite regime, of high deletion fraction. As a first step, notice that in any reasonable model, including list-decoding, we cannot hope to efficiently decode from a 1/2 deletion fraction with a polynomial list size and constant rate. With block length n and $n/2$ deletions, the adversary can ensure that what is received is either $n/2$ 1's or $n/2$ 0's.

Thus, for binary codes and $\varepsilon > 0$, we will consider the question of whether it is possible to list decode from a fraction $1/2 - \varepsilon$ of deletions.

▶ **Definition 17.** We say that a code $C \subseteq \{0,1\}^m$ is list-decodable from a δ deletion fraction with list size L if every sequence of length $(1-\delta)m$ is a subsequence of at most L codewords. If this is the case, we will call C (δ, L) *list-decodable from deletions.*

▶ Remark. Although the results of this section are proven in the setting of list-decoding, it is *not* known that we cannot have unique decoding of binary codes up to deletion fraction $1/2 - \varepsilon$. See the first open problem in Section 6.

5.1 List-decodable binary deletion codes (existential)

In this section, we show that good list-decodable codes exist. This construction will be the basis of our explicit construction of list-decodable binary codes. The proof appears in the appendix.

▶ **Theorem 18.** *Let $\delta, L > 0$. Let $C \subseteq \{0,1\}^m$ consist of 2^{Rm} independently, uniformly chosen strings, where $R \leq 1 - h(\delta) - 3/L$. Then C is $\big(\delta, L\big)$ list-decodable from deletions with probability at least $1 - 2^{-m}$.*

Moreover, such a code can be constructed and decoded in time $2^{\mathrm{poly}(mL)}$.

In particular, when $\delta = 1/2 - \varepsilon$, we can construct and decode in time $2^{\mathrm{poly}(m/\varepsilon)}$ a code $C \subseteq \{0,1\}^m$ of rate $\Omega(\varepsilon^2)$ which is $\big(\delta, O(1/\varepsilon^2)\big)$ list-decodable from deletions.

5.2 List-decodable binary deletion codes (explicit)

We now use the existential construction of Theorem 18 to give an explicit construction of constant-rate list-decodable binary codes. Our code construction uses Parvaresh-Vardy codes ([19]) as outer codes, and an inner code constructed using Section 5.1.

The idea is to list-decode "enough" windows and then apply the list recovery algorithm of Theorem 20.

▶ **Theorem 19.** *Let $0 < \varepsilon < 1/2$. There is an explicit binary code $C \subseteq \{0,1\}^N$ of rate $\tilde{\Omega}(\varepsilon^3)$ which is list-decodable from a $1/2 - \varepsilon$ fraction of deletions with list size $(1/\varepsilon)^{O(\log\log \varepsilon)}$.*

This code can be constructed, encoded, and list-decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

We will appeal in our analysis to the following theorem, which can be found in [10].

▶ **Theorem 20** ([10], Corollary 5)**.** *For all integers $s \geq 1$, for all prime powers r, every pair of integers $1 < K \leq N \leq q$, there is an explicit $\mathbb{F}_r$-linear map $E\colon \mathbb{F}_q^K \to \mathbb{F}_{q^s}^N$ whose image C' is a code satisfying:*

- *There is an algorithm which, given a collection of subsets $S_i \subseteq \mathbb{F}_{q^s}$ for $i \in [N]$ with $\sum_i |S_i| \leq N\ell$, runs in $\mathrm{poly}\big((rs)^s, q, \ell\big)$ time, and outputs a list of size $O\big((rs)^s N\ell/K\big)$ that includes precisely the set of codewords $(c_1, \ldots, c_N) \in C'$ that satisfy $c_i \in S_i$ for at least αN values of i, provided*

$$\alpha > (s+1)(K/N)^{s/(s+1)}\ell^{1/(s+1)}.$$

The code: We set $s = O(\log 1/\varepsilon)$, $r = O(1)$, and $N = K\,\mathrm{poly}\big(\log(1/\varepsilon)\big)/\varepsilon$ in Theorem 20 in order to obtain a code $C' \subseteq \mathbb{F}_{q^s}^N$. We modify the code, replacing the ith coordinate c_i with the pair (i, c_i) for each i, in order to obtain a code C''. This latter step only reduces the rate by a constant factor.

Recall that we are trying to recover from a $1/2-\varepsilon$ fraction of deletions. We use Theorem 18 to construct an inner code $C_1\colon [N]\times\mathbb{F}_q^s\to\{0,1\}^m$ of rate $\Omega(\varepsilon^2)$ which recovers from a $1/2-\delta$ deletion fraction (where we will set $\delta=\varepsilon/4$). Our final code C is a concatenation of C'' with C_1, which has rate $\tilde{\Omega}(\varepsilon^3)$.

▶ **Theorem 21.** *C is list-decodable from a $1/2-\varepsilon$ fraction of deletions in time $N^{\mathrm{poly}(1/\varepsilon)}$.*

Proof. Our algorithm first defines a set of "decoding windows". These are intervals of length $(1/2+\delta)m$ in the received codeword which start at positions $1+t\delta m$ for $t=0,1,\ldots,N/\delta-(1/2+\delta)/\delta$, in addition to one interval consisting of the last $(1/2+\delta)m$ symbols in the received codeword.

We use the algorithm of Theorem 18 to list-decode each decoding window, and let $\mathcal{L}$ be the union of the lists for each window. Finally, we apply the algorithm of Theorem 20 to $\mathcal{L}$ to obtain a list containing the original message.

Correctness: Let $c=(c_1,\ldots,c_N)$ be the originally transmitted codeword of C'. If an inner codeword $C_1(i,c_i)$ has suffered fewer than a $1/2-2\delta$ fraction of deletions, then one of the decoding windows is a substring of $C_1(i,c_i)$, and $\mathcal{L}$ will contain the correct pair (i,c_i).

When $\delta=\varepsilon/4$, by a simple averaging argument, we have that an ε fraction of inner codewords have at most $1/2-2\delta$ fraction of positions deleted. For these inner codewords, $\mathcal{L}$ contains a correct decoding of the corresponding symbol of c.

In summary, we have list-decoded at most N/δ windows, with a list size of $O(1/\delta^2)$ each. We also have that an ε fraction of symbols in the outer codeword of C' is correct. Setting $\ell=O(1/\delta^3)$ in the algorithm of Theorem 20, we can take $\alpha=\varepsilon$. Theorem 20 then guarantees that the decoder will output a list of $\mathrm{poly}(1/\varepsilon)$ codewords, including the correct codeword c. ◀

6 Conclusion and open problems

In this work, we initiated a systematic study of codes for the adversarial deletion model, with an eye towards constructing codes achieving more-or-less the correct trade-offs at the high-noise and high-rate regimes. There are still several major gaps in our understanding of deletion codes, and below we highlight some of them (focusing only on the worst-case model):

1. For binary codes, what is the supremum p^* of all fractions p of adversarial deletions for which one can have positive rate? Clearly $p^*\le 1/2$; could it be that $p^*=1/2$ and this trivial limit can be matched? Or is it the case that p^* is strictly less than $1/2$?
2. Can one construct codes of rate $1-p-\gamma$ to efficiently correct a fraction p of deletions over an alphabet size that only depends on γ?

   Note that this requires a relative distance of p, and currently we only know algebraic-geometric and expander-based codes which achieve such a tradeoff between rate and relative distance.
3. Can one improve the rate of the binary code construction to correct a fraction ε of deletions to $1-\varepsilon\,\mathrm{poly}(\log(1/\varepsilon))$, approaching more closely the existential $1-O(\varepsilon\log(1/\varepsilon))$ bound?

   In the case of errors, an approach using expanders gives the analogous tradeoff (see [7] and references therein). Could such an approach be adapted to the setting of deletions?

References

1 N. Alon, J. Edmonds, and M. Luby. Linear time erasure codes with nearly optimal recovery (extended abstract). In *FOCS*, pages 512–519. IEEE Computer Society, 1995.

2 J. Brakensiek, V. Guruswami, and S. Zbarsky. Efficient low-redundancy codes for correcting multiple deletions. In preparation, 2015.

3 J. Chen, M. Mitzenmacher, C. Ng, and N. Varnica. Concatenated codes for deletion channels. In *IEEE International Symposium on Information Theory, 2003*, pages 218–218, June 2003.

4 M. Davey and D. J. C. MacKay. Reliable communication over channels with insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 47(2):687–698, Feb 2001.

5 S. Diggavi and M. Grossglauser. On information transmission over a finite buffer channel. *IEEE Transactions on Information Theory*, 52(3):1226–1237, March 2006.

6 R. Gallager. Sequential decoding for binary channels with noise and synchronization errors, October 1961. Lincoln Lab. Group Report.

7 V. Guruswami. Guest column: error-correcting codes and expander graphs. *SIGACT News*, 35(3):25–41, 2004.

8 V. Guruswami and P. Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005.

9 V. Guruswami and A. Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

10 V. Guruswami and A. Rudra. Soft decoding, dual BCH codes, and better list-decodable e-biased codes. In *Proceedings of the 2008 IEEE 23rd Annual Conference on Computational Complexity*, CCC '08, pages 163–174, Washington, DC, USA, 2008. IEEE Computer Society.

11 Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *CoRR*, abs/1411.6667, 2014.

12 A. Kalai, M. Mitzenmacher, and M. Sudan. Tight asymptotic bounds for the deletion channel with small deletion probabilities. In *ISIT*, pages 997–1001, 2010.

13 Y. Kanoria and A. Montanari. Optimal coding for the binary deletion channel with small deletion probability. *IEEE Transactions on Information Theory*, 59(10):6192–6219, Oct 2013.

14 I. Kash, M. Mitzenmacher, J. Thaler, and J. Ullman. On the zero-error capacity threshold for deletion channels. In *Information Theory and Applications Workshop (ITA), 2011*, pages 1–5, Feb 2011.

15 M. Kiwi, M. Loebl, and J. Matoušek. Expected length of the longest common subsequence for large alphabets. *Advances in Mathematics*, 197:480–498, November 2004.

16 V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*, 10(8):707–710, 1966.

17 M. Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009.

18 M. Mitzenmacher and E. Drinea. A simple lower bound for the capacity of the deletion channel. *IEEE Transactions on Information Theory*, 52(10):4657–4660, 2006.

19 F. Parvaresh and A. Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 285–294, 2005.

20 E. Ratzer. Marker codes for channels with insertions and deletions. *Annals of Telecommunications*, 60(1–2):29–44, Jan–Feb 2005.

21 L. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, November 1999.

22 K. W. Shum, I. Aleshnikov, P. V. Kumar, H. Stichtenoth, and V. Deolalikar. A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 47(6):2225–2241, 2001.

23 N. J. A. Sloane. On single-deletion-correcting codes. *CoRR*, arxiv.org/abs/math/0207197, 2002.

24 L. R. Welch and E. R. Berlekamp. Error correction of algebraic block codes. *US Patent Number 4,633,470*, December 1986.

25 K. Zigangirov. Sequential decoding for a binary channel with drop-outs and insertions. *Problemy Peredachi Informatsii*, 5(2):23–30, 1969.