Electronic Colloquium on Computational Complexity, Report No. 117 (2015)

An improved bound on the fraction of correctable deletions

Boris Bukh* Venkatesan Guruswami†

Abstract

We consider codes over fixed alphabets against worst-case symbol deletions. For any fixed $k \geqslant 2$, we construct a family of codes over alphabet of size k with positive rate, which allow efficient recovery from a worst-case deletion fraction approaching $1 - \frac{2}{k+1}$. In particular, for binary codes, we are able to recover a fraction of deletions approaching $1/3$. Previously, even non-constructively the largest deletion fraction known to be correctable with positive rate was $1 - \Theta(1/\sqrt{k})$, and around 0.17 for the binary case.

Our result pins down the largest fraction of correctable deletions for k-ary codes as $1 - \Theta(1/k)$, since $1 - 1/k$ is an upper bound even for the simpler model of erasures where the locations of the missing symbols are known.

Closing the gap between $1/3$ and $1/2$ for the limit of worst-case deletions correctable by binary codes remains a tantalizing open question.

1 Introduction

This work concerns error-correcting codes capable of correcting *worst-case* deletions. Specifically, consider a fixed alphabet $[k] \stackrel{\text{def}}{=} \{1, 2, \ldots, k\}$, and suppose we transmit a sequence of n symbols from $[k]$ over a channel that can adversarially *delete* an arbitrary fraction p of symbols, resulting in a subsequence of length $(1-p)n$ being received at the other end. The location of the deleted symbols are unknown to the receiver. The goal is to design a code $C \subseteq [k]^n$ such that every $c \in C$ can be uniquely recovered from any of its subsequences caused by up to pn deletions. Equivalently, for $c \neq c' \in C$, the length of the longest common subsequence of c, c', which we denote by $\mathrm{LCS}(c, c')$, must be less than $(1-p)n$.

In this work, we are interested in the question of correcting as large a fraction p of deletions as possible with codes of positive rate (bounded away from 0 for $n \to \infty$). That is, we would like $|C| \geqslant \exp(\Omega_k(n))$ so that the code incurs only a constant factor redundancy (this factor could depend on k, which we think of as fixed).

Denote by $p^*(k)$ the limit superior of all $p \in [0,1]$ such that there is a positive rate code family over alphabet $[k]$ that can correct a fraction p of deletions. The value of $p^*(k)$ is not known for any value of k. Clearly, $p^*(k) \leqslant 1 - 1/k$ — indeed, one can delete all but n/k occurrences of the most frequent symbol in a word to leave one of k possible subsequences, and therefore only trivial codes with k codewords can correct a fraction $1 - 1/k$ of deletions. This trivial limit remains the best known upper bound on $p^*(k)$. We note that

*Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Email: bbukh@math.cmu.edu. Supported in part by U.S. taxpayers through NSF grant DMS-1201380.

†Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213. Email: guruswami@cmu.edu. Supported in part by NSF grants CCF-1422045 and CCF-0963975.

ISSN 1433-8092

this upper bound holds even for the simpler model of erasures where the locations of the missing symbols are known at the receiver (this follows from the so-called Plotkin bound in coding theory).

Whether the trivial upper bound $p^*(k) \leqslant 1-1/k$ can be improved, or whether there are in fact codes capable of correcting deletion fractions approaching $1-1/k$ is an outstanding open question concerning deletion codes and the combinatorics of longest common subsequences. Perhaps the most notable of these is the $k=2$ (binary) case. The current best lower bound on $p^*(2)$ is around 0.17. This bound comes from the random code, in view of the fact that the expected LCS of two random words in $\{0,1\}^n$ is at most $0.8263n$ [7]. As the LCS of two random words in $\{0,1\}^n$ is at least 0.788, one cannot prove any lower bound on $p^*(2)$ better than 0.22 using the random code. Kiwi, Loebl, and Matoušek [6] showed that, as $k\to\infty$, we have $\mathbb{E}[\mathrm{LCS}(c,c')] \sim \frac{2}{\sqrt{k}}n$ for two random words $c,c' \in [k]^n$. This was used in [5] to deduce $p^*(k) \geqslant 1-O(1/\sqrt{k})$.

The above discussion only dealt with the *existence* of deletion codes. Turning to explicit and efficiently decodable constructions, Schulman and Zuckerman [10] constructed constant-rate binary codes which are efficiently decodable from a *small* constant fraction of worst-case deletions. This was improved in [5]; in the new codes, the rate approaches 1. Specifically, it was shown that one can correct a fraction $\zeta > 0$ of deletions with rate about $1-O(\sqrt{\zeta})$. In terms of correcting a larger fraction of deletions, codes that are efficiently decodable from a fraction $1-\gamma$ of errors over a $\mathrm{poly}(1/\gamma)$ sized alphabet were also given in [5].

Our focus in this work is exclusively on the worst-case model of deletions. For random deletions, it is known that reliable communication at positive rate is possible for deletion fractions approaching 1 even in the binary case. We refer the reader interested in coding against random deletions to the survey by Mitzenmacher [8].

1.1 Our results

Here we state our results informally, omitting the precise computational efficiency guarantees, and omitting the important technical properties of constructed codes related to the "span" of common subsequences (see Section 2 for the definition). The precise statements are in Subsection 4.2 and in Section 5.

Our first result is a construction of codes which are combinatorially capable of correcting a larger fraction of deletions than was previously known to be possible.

Theorem 1 (Informal)**.** *For all integers $k \geqslant 2$, $p^*(k) \geqslant \frac{k-1}{k+1}$. Furthermore, for any desired $\varepsilon > 0$, there is an efficiently constructible family of k-ary codes of rate $r(k,\varepsilon) > 0$ such that the* LCS *of any two distinct codewords is less than fraction $\frac{2}{k+1}+\varepsilon$ of the code length. In particular, there are explicit binary codes that can correct a fraction $1/3-\varepsilon$ of deletions, for any fixed $\varepsilon > 0$.*

Note that, together with the trivial upper bound $p^*(k) \leqslant 1-1/k$, the result pins down the asymptotics of $1-p^*(k)$ to $\Theta(1/k)$ as $k\to\infty$.

In our second result we construct codes with the above guarantee together with an efficient algorithm to recover from deletions:

Theorem 2 (Informal)**.** *For any integer $k \geqslant 2$ and any $\varepsilon > 0$, there is an efficiently constructible family of k-ary codes of rate $r(k,\varepsilon) > 0$ that can be decoded in polynomial (in fact near-linear) time from a fraction $1-\frac{2}{k+1}-\varepsilon$ of deletions.*

1.2 Our techniques

All our results are based on code concatenations, which use an outer code over a large alphabet with desirable properties, and then further encode the codeword symbols by a judicious inner code.

The innermost code consists of words of the form $(1^A 2^A \dots \mathrm{k}^A)^{L/A}$. Informally, we think of these words as oscillating with amplitude A (this can be made precise via Fourier transform for example, but we won't need it in our analysis). The crucial property, that was observed in [3], is that two such words have a long common subsequence only if they amplitudes are close. This was property was also exploited in [2] to show a certain weak limitation of deletion codes, namely that in any set of $t \geqslant k+2$ words in $[k]^n$, some two of them have an LCS at least $\frac{n}{k} + c(k,t)n^{1-1/(t-k-2)}$.

The effective use of these codes as inner codes in a concatenation scheme relies on a property stronger than absence of long common subsequences between codewords. Informally, the property amounts to absence of long common subsequences between subwords of codewords. For the precise notion, consult the definition of a *span* in the next section and the statement of Theorem 3 in the following section. Using this, we are able to show that if the outer code has a small LCS value, then the LCS of the concatenated code approaches a fraction $\frac{2}{k+1}$ of the block length.

For the outer code, the simplest choice is the random code. This gives the existential result (Theorem 8). Using the explicit construction of codes to correct a large fraction of deletions over fixed alphabets from [5] gives us a polynomial (in fact near-linear) time deterministic construction (Theorem 10). While the outer code from [5] is also efficiently decodable from deletions, it is not clear how to exploit this to decode the concatenated code efficiently.

To obtain codes that are also efficiently decodable, we employ another level of concatenation, using Reed–Solomon codes at the outermost level, and the above explicit concatenated code itself as the inner code. The combinatorial LCS property of these codes is established similarly, and is in fact easier, as we may assume (by indexing each position) that all symbols in an outer codeword are distinct, and therefore the corresponding inner codewords are distinct. To decode the resulting concatenated code, we try to decode the inner code (by brute-force) for many different contiguous subwords of the received subsequence. A small fraction of these are guaranteed to succeed in producing the correct Reed–Solomon symbol. The decoding is then completed via *list decoding* of Reed–Solomon codes. The approach here is inspired by the algorithm for list decoding binary codes from a deletion fraction approaching $1/2$ in [5]. Our goal here is to recover the correct message *uniquely*, but by virtue of the combinatorial guarantee, there can be at most one codeword with the received word as a subsequence, so we can go over the (short) list and identify the correct codeword. Note that list decoding is used as an intermediate algorithmic primitive even though our goal is unique decoding; this is similar to [4] that gave an algorithm to decode certain low-rate concatenated codes up to half the Gilbert–Varshamov bound via a list decoding approach.

2 Preliminaries

A *word* is a sequence of symbols from a finite alphabet. For the problems of this paper, only the size of the alphabet and the length of the word are important. So, we will often use $[k]$ for a canonical k-letter alphabet, and consider the words indexed by $[n]$. In this case, a word of length n over alphabet $[k]$ will be denoted $[k]^n$. We treat symbols in a word as *distinguishable*. So, if x denotes the second 1 in the word 21011 and we delete the subword 10, the variable x now refers to the first 1 in the word 211.

A subsequence in a word w is any word obtained from w by deleting one or more symbols. In contrast, a *subword* is a subsequence made of several consecutive symbols of w. The *span* of a subsequence w' in a word w is the length of the smallest subword containing the subsequence. We denote it by $\operatorname{span}_w w'$, or simply by $\operatorname{span} w'$ when no ambiguity can arise.

A *common subsequence* between words w_1 and w_2 is a pair of subsequences w'_1 in w_1 and w'_2 in w_2 that are equal as words, i.e., $\operatorname{len} w'_1 = \operatorname{len} w'_2$ and $w'_1[i] = w'_2[i]$ for all i.

For words w_1, w_2, we denote by $\mathrm{LCS}(w_1, w_2)$ the length of the *longest common subsequence* of w_1 and w_2, i.e., the largest j for which there is a common subsequence between w_1 and w_2 of length j.

A *code* C of block length n over the alphabet $[k]$ is simply a subset of $[k]^n$. We will also call such codes as k-ary codes, with binary codes referring to the $k = 2$ case. The *rate* of C equals $\frac{\log|C|}{k \log n}$.

For a code $C \subseteq [k]^n$, its *"LCS value"* is defined as the

$$\mathrm{LCS}(C) \stackrel{\text{def}}{=} \max_{c_1 \neq c_2 \in C} \mathrm{LCS}(c_1, c_2) .$$

Note that a code $C \subseteq [k]^n$ is capable of recovering from t worst-case deletions if and only if $\mathrm{LCS}(C) < n - t$.

We define *span of a common subsequence* (w'_1, w'_2) as

$$\operatorname{span}(w'_1, w'_2) \stackrel{\text{def}}{=} \operatorname{span} w'_1 + \operatorname{span} w'_2.$$

The span will play an important role in our analysis of $\mathrm{LCS}(C)$ of the codes C we construct, by virtue of the fact that if $\operatorname{span}(w'_1, w'_2) \geqslant b \cdot \operatorname{len} w'_1$ for every common subsequence of $w_1, w_2 \in [k]^n$, then $\mathrm{LCS}(w_1, w_2) \leqslant \frac{2n}{b}$. Our result will be based on a construction for which we can take $b \approx k + 1$ for long enough common subsequences of any distinct pair of codewords.

Concatenated codes. Our results heavily use the simple but useful idea of code concatenation. Given an *outer* code $C_{\text{out}} \subseteq [Q]^n$, and an injective map $\tau \colon [Q] \to [q]^m$ defining the encoding function of an *inner* code C_{in}, the concatenated code $C_{\text{concat}} \subseteq [q]^{nm}$ is obtained by composing these codes as follows. If $(c_1, c_2, \ldots, c_n) \in [Q]^n$ is a codeword of C_{out}, the corresponding codeword in C_{concat} is $(\tau(c_1), \ldots, \tau(c_n)) \in [q]^{nm}$. The words $\tau(c_i) \in C_{\text{in}}$ will be referred to as the *inner blocks* of the concatenated codeword, with the i'th block corresponding to the i'th outer codeword symbol.

3 Alphabet reduction for deletion codes

Fix k to be the alphabet size of the desired deletion code. We shall show how to turn words over K-letter alphabet, for $K \gg k$, without large common subsequence into words over k-letter alphabet without large common subsequence. More specifically, for any $\varepsilon > 0$ and large enough integer $K = K(\varepsilon)$, we give a method to transform a deletion code $C_1 \subseteq [K]^n$ with $\mathrm{LCS}(C_1) \ll \varepsilon n$ into a deletion code $C_2 \subseteq [k]^N$ with $\mathrm{LCS}(C_2) \leqslant \left(\frac{2}{k+1} + \varepsilon\right)N$. The transformation lets us transform a *crude* dependence between the alphabet size of the code C_1 and its LCS value (i.e., between K and ε), into a *quantitatively strong* one, namely $\mathrm{LCS}(C_2) \approx \frac{2}{k+1}N$. The code C_2 will in fact be obtained by concatenating C_1 with an inner k-ary code with K codewords, and therefore have the same size as C_1. The block length N of C_2 will be much larger than n, but the ratio N/n will be bounded as a function of k, K, and ε. The rate of C_2 will thus only be a constant factor smaller than that of C_1.

Specifically, we will prove the following:

Theorem 3. *Let $C_1 \subseteq [K]^n$ be a code with $\mathrm{LCS}(C_1) = \gamma n$, and let $k \geqslant 2$ be an integer. Then there exists an integer $T = T(K,\gamma,k)$ satisfying $T \leqslant 32 \cdot (2k/\gamma)^K$, and an injective map $\tau \colon [K] \to [k]^T$ such that the code $C_2 \subseteq [k]^N$ for $N = nT$ obtained by replacing each symbol in codewords of C_1 by its image under τ has the following property: if s is a common subsequence between two distinct codewords $c, c' \in C_2$, then*

$$\operatorname{span} s \geqslant (k+1) \operatorname{len} s - 4\gamma k N \ .$$

In particular, since $\operatorname{span} s \leqslant 2N$, *we have* $\mathrm{LCS}(C_2) \leqslant \left(\frac{2+4\gamma k}{k+1}\right) N < \left(\frac{2}{k+1} + 4\gamma\right) N$.

Thus, one can construct codes over a size k alphabet with LCS value approaching $\frac{2}{k+1}$ by starting with an outer code with LCS value $\gamma \to 0$ over any fixed size alphabet, and concatenating it with a constant-sized map. The span property will be useful in concatenated schemes to get longer, efficiently decodable codes. The key to the above construction is the inner map, described next.

3.1 The construction

We now describe the way to encode symbols from the alphabet $[K]$ as words over $[k]$ that underlies Theorem 3. Let L be constant to be chosen later. For an integer A dividing L, define word of "amplitude A" to be

$$f_A \stackrel{\text{def}}{=} (1^A 2^A \ldots \mathsf{k}^A)^{L/A}. \tag{1}$$

The crucial property of these words is that f_A and f_B have no long common subsequence if B/A is large (or small); for the proof see one of [3, 2]. In the present work, we will need an asymmetric version of this observation — we will need to analyze common subsequences in subwords of f_A and f_B.

Let $R \geqslant 2$ be an integer to be chosen later. Let $R_l = R^{l-1}$. For a word w over alphabet $[K]$ denote by $\hat{w}$ the word obtained from w via the substitution

$$l \in [K] \quad \mapsto \quad f_{R_l}. \tag{2}$$

to each symbol of w. Note that $\operatorname{len} \hat{w} = kL \operatorname{len} w$. If a symbol $x \in \hat{w}$ is obtained by expanding symbol $y \in w$, then we say that y is a *parent* of x.

3.2 Analysis

Lemma 4. *For a natural number P, let f_A^∞ be the (infinite) word*

$$(1^A 2^A \ldots k^A)^*$$

Let A, B be natural numbers, and suppose $s = (w_1', w_2')$ is a common subsequence between f_A^∞ and f_B^∞. Then

$$\operatorname{span} s \geqslant \left(k + 1 - \frac{kA}{B}\right) \operatorname{len} s - 2(A+B).$$

Proof. The words f_A^∞ and f_B^∞ are concatenations of *chunks*, which are subwords of the form l^A and l^B respectively. A chunk in f_A^∞ is *spanned* by subsequence w_1' if the span of w_1' contains at least one symbol of the chunk. Similarly, we define chunks spanned by w_2' in f_B^∞. We will estimate how many chunks are spanned by w_1' and by w_2'.

As a word, a common subsequence is of the form $k_1^{p_1} k_2^{p_2} \cdots k_s^{p_s}$ where $k_l \neq k_{l+1}$ and the exponents are positive. The subsequence $k_l^{p_l}$ spans at least $k \left\lceil \frac{p_l - A}{A} \right\rceil + 1$ chunks in f_A^∞. Similarly, $k_l^{p_l}$ spans at least $k \left\lceil \frac{p_l - B}{B} \right\rceil + 1$ chunks in f_B^∞. Therefore the total number of symbols in chunks spanned by $k_l^{p_l}$ in both f_A^∞ and in f_B^∞ is at least

$$\phi(p_l) \stackrel{\text{def}}{=} A \left(k \left\lceil \frac{p_l - A}{A} \right\rceil + 1 \right) + B \left(k \left\lceil \frac{p_l - B}{B} \right\rceil + 1 \right)$$

We then estimate $\phi(p_l)$ according to whether $p_l \leqslant B$:

$$\phi(p_l) \geqslant \begin{cases} k(p_l - A) + B & \text{if } p_l \leqslant B, \\ k(p_l - A) + k(p_l - B) + B & \text{if } p_l > B. \end{cases}$$

In both cases we have

$$\phi(p_l) \geqslant \left(k + 1 - \frac{kA}{B} \right) p_l.$$

Note that the chunks spanned by $k_l^{p_l}$ are distinct from chunks spanned by $k_{l'}^{p_{l'}}$ for $l \neq l'$. So, the total number of symbols in all chunks spanned by subsequence s in both f_A^∞ and f_B^∞ is least

$$\sum_l \phi(p_l) \geqslant \left(k + 1 - \frac{kA}{B} \right) \operatorname{len} s.$$

The total span of s might be smaller since the first and the last chunks in each of f_A^∞ and f_B^∞ might not be fully spanned. Subtracting $2(A+B)$ to account for that gives the stated result. $\square$

Let (w_1', w_2') be a common subsequence between $\hat{w}_1$ and $\hat{w}_2$. We say that the i'th symbol in (w_1', w_2') is *well-matched* if the parents of $w_1'[i]$ and of $w_2'[i]$ are equal. A common subsequence is *badly-matched* if none of its symbols are well-matched.

Lemma 5. *Suppose w_1, w_2 are words over alphabet $[K]$ and $s = (w_1', w_2')$ is a badly-matched common subsequence between $\hat{w}_1$ and $\hat{w}_2$. Then*

$$\operatorname{span} w_1' + \operatorname{span} w_2' \geqslant \left(k + 1 - \frac{k}{R} - \frac{4R^{K-1}}{Lk} \right) \operatorname{len} s - 16R^{K-1}.$$

Proof. We subdivide the common subsequence s into subsequences $s_1, \dots, s_r$ such that, for each $i = 1, \dots, r$ and each $j = 1, 2$, the symbols matched by s_i in w_j' belong to the expansion of the same symbol in w_j. We choose the subdivision to be the coarsest with this property. That implies that pairs of symbols of w_1 and w_2 matched by s_i and by s_{i+1} are different. In particular, expansions of at least $r-4$ symbols of w_1 and w_2 are fully contained in the spans of w_1' and w_2', and so

$$Lk(r-4) \leqslant \operatorname{span} s.$$

By the preceding lemma we then have

$$\operatorname{span} s \geqslant \left(k + 1 - \frac{k}{R} \right) \operatorname{len} s - 4rR^{K-1} \geqslant \left(k + 1 - \frac{k}{R} \right) \operatorname{len} s - 4R^{K-1} \left(\frac{\operatorname{span} s}{Lk} + 4 \right).$$

The lemma then follows from the collecting together the two terms involving $\operatorname{span} w_1' + \operatorname{span} w_2'$, and then dividing by $1 + 4R^{K-1}/Lk$. $\square$

The next step is to drop the assumption in Lemma 5 that the common subsequence is badly-matched. By doing so we incur an error term involving $\mathrm{LCS}(w_1, w_2)$.

Lemma 6. *Suppose w_1, w_2 are words over alphabet $[K]$ and $s = (w'_1, w'_2)$ is a common subsequence between $\hat{w}_1$ and $\hat{w}_2$. Then*

$$\operatorname{span} s \geqslant \left(k + 1 - \frac{k}{R} - \frac{4R^{K-1}}{Lk}\right) \operatorname{len} s - 2Lk(k+1) \cdot \mathrm{LCS}(w, w') - 16R^{K-1}.$$

Proof. Without loss, the subsequence s is locally optimal, i.e., every alteration of s that increases $\operatorname{len} s$ also increases $\operatorname{span} s$. Define an auxiliary bipartite graph G whose two parts are the symbols in w_1 and the symbols in w_2. For each well-matched symbol in s we join the parent symbols in w_1 and w_2 by an edge.

We may assume that each vertex in G has degree at most 2. Indeed, suppose a symbol $x \in w$ is adjacent to three symbols $y_1, y_2, y_3 \in w'$ with y_2 being in between y_1 and y_3. Then we alter s by first removing all matches between x and y_1, y_2, y_3, and then completely matching x with y_2. The alteration does not increase $\operatorname{span} s$, and the result is a common subsequence that is at least as long as s, and whose auxiliary graph has fewer edges. We can then repeat this process until no vertex has degree exceeding 2.

Consider a maximum-sized matching in G. On one hand, it has at most $\mathrm{LCS}(w, w')$ edges. On the other hand, since the maximum degree of G is at most 2, the maximum-sized matching has at least $|E(G)|/2$ edges. Hence, $|E(G)| \leqslant 2\,\mathrm{LCS}(w, w')$.

Remove from s all well-matched symbols to obtain a common subsequence s'. The new subsequence satisfies

$$\operatorname{len} s' \geqslant \operatorname{len} s - Lk \cdot |E(G)| \geqslant \operatorname{len} s - 2Lk \cdot \mathrm{LCS}(w, w').$$

It is also clear that s' is a badly-matched common subsequence. From the previous lemma

$$\operatorname{span} s' \geqslant \left(k + 1 - \frac{k}{R} - \frac{4R^{K-1}}{Lk}\right) \operatorname{len} s - 2Lk(k+1) \cdot \mathrm{LCS}(w, w') - 16R^{K-1}.$$

Since $\operatorname{span} s \geqslant \operatorname{span} s'$, the lemma follows. □

We are now ready to prove Theorem 3 by picking parameters suitably.

Proof of Theorem 3. Recall that we are starting with a code $C_1 \subseteq [K]^n$ with $\mathrm{LCS}(C_1) = \gamma n$. Given $\varepsilon > 0$ and an integer $k \geqslant 2$, pick parameters

$$R = \left\lceil \frac{2k}{\gamma} \right\rceil \quad \text{and} \quad L = 16R^{K-1} \left\lceil \frac{1}{\gamma k} \right\rceil$$

in the construction (1) and (2). Define $T = kL$ and $\tau \colon [K] \to [k]^T$ as $\tau(l) = f_{R_l}$, and let $C_2 \subseteq [k]^N$, where $N = nkL$, be the code obtained as in the statement of Theorem 3. Note that $T \leqslant 32 \cdot (2k/\gamma)^K$ by our choice of parameters.

By Lemma 6, we can conclude that any common subsequence s of two distinct codewords of C_2 satisfies

$$\operatorname{span} s \geqslant (k + 1 - \gamma) \operatorname{len} s - 2(k+1)\gamma N - \gamma N\,.$$

Since $\operatorname{len} s \leqslant N$ and $k \geqslant 2$, the right hand side is at least $(k+1)\operatorname{len} s - 4k\gamma N$, as desired. □

4 Existence and construction of good deletion codes

In this section, we will plug in good "outer" deletion codes over large alphabets into Theorem 3 to derive codes over alphabet $[k]$ that correct a fraction $\approx (k-1)/(k+1)$ of deletions.

4.1 Existential claims

We start with "outer" codes over large alphabets guaranteed to exist by the probabilistic method. We use $h(\cdot)$ to denote the binary entropy function. A similar statement appears in [5], but we include the short proof for completeness.

Lemma 7. *Suppose $\gamma, r > 0$ and integer $K \geqslant 2$ satisfy*

$$2r \log K + 2h(\gamma) - \gamma \log K < 0.$$

Then, for all large n, there exists a code with K^{rn} codewords in $[K]^n$ such that $\mathrm{LCS}(w, w') < \gamma n$ for all distinct w, w' in the code.

Proof. Let $w_1, \dots, w_{K^{rn}}$ be a sequence of words sampled from $[K]^n$ independently at random *without replacement*. For any $i < j$ the joint distribution of (w_i, w_j) is same as of two words independently sampled from $[K]^n$ conditioned on them being distinct. Hence, by the union bound we have

$$Pr[\mathrm{LCS}(w_i, w_j) > \gamma n] \leqslant \binom{n}{\gamma n}^2 K^{-\gamma n}.$$

By the second application of the union bound we thus have

$$\Pr[\exists w, w' \in \mathscr{C}_0,\ \mathrm{LCS}(w, w') \geqslant \gamma n] \leqslant K^{2rn} \binom{n}{\gamma n}^2 K^{-\gamma n} < 1.$$

As this probability is less than 1, there is a choice of $w_1, \dots, w_{Mn}$ such that pairwise LCS is less than γn. $\square$

Using the above existential bound in Theorem 3, we now deduce the following.

Theorem 8 (Existence of deletion codes)**.** *Fix an integer $k \geqslant 2$. Then for every real number $\varepsilon > 0$, there is $\tilde{r} = (\varepsilon/k)^{O(\varepsilon^{-3})}$ such that for infinitely many N there is a code $C \subseteq [k]^N$ of rate at least $\tilde{r}$ and $\mathrm{LCS}(C) < \left(\frac{2}{k+1} + \varepsilon\right) N$.*

Proof. We first apply Lemma 7 with $\gamma = \varepsilon/4$ and $r = \gamma/6 = \varepsilon/24$ to get a code $C_1 \subseteq [K]^n$ for $K \leqslant O(1/\varepsilon^3)$ with $\mathrm{LCS}(C_1) \leqslant \varepsilon n/4$ and $|C_1| \geqslant K^{rn}$. Now applying Theorem 3 to C_1 yields a code $C_2 \subseteq [k]^N$ with $\mathrm{LCS}(C_2) \leqslant \left(\frac{2}{k+1} + \varepsilon\right) N$. The rate of C_2 is at least $r/T \geqslant (\varepsilon/k)^{O(\varepsilon^{-3})}$ since $T \leqslant (k/\varepsilon)^{O(K)}$. $\square$

Remark 1. The exponent $O(1/\varepsilon^3)$ in the rate can be improved to $O(1/\varepsilon^a)$ for any $a > 2$. We made the concrete choice $a = 3$ for notational convenience.

4.2 Efficient deterministic construction

Theorem 8 already shows the *existence* of positive rate codes over the alphabet $[k]$ which are capable of correcting a deletion fraction approaching $\frac{k-1}{k+1}$, giving our main combinatorial result. We now turn to explicit constructions of such codes. Given Theorem 3, all that we need is an explicit code family capable of correcting a deletion fraction approaching 1 over constant-sized alphabets, which is guaranteed by the following theorem.

Lemma 9 ([5], Thm 3.4)**.** *For every $\gamma > 0$ there exists an integer $K \leqslant O(1/\gamma^5)$ such that for infinitely many block lengths n, one can construct a code $C \subseteq [K]^n$ of rate $\Omega(\gamma^3)$ and $\mathrm{LCS}(C) \leqslant \gamma n$ in time $n(\log n)^{\mathrm{poly}(1/\gamma)}$. Further, the code C can be efficiently encoded and decoded from a fraction $(1-\gamma)$ of deletions in $n \cdot (\log n)^{\mathrm{poly}(1/\gamma)}$ time.*

Remark 2. The linear dependence on n in the decoding time can be deduced using fast ($n \cdot \mathrm{poly}(\log n)$ time) unique decoding algorithms for Reed–Solomon codes. The bounds stated in [5] are $n^{O(1)}(\log n)^{\mathrm{poly}(1/\gamma)}$ time.

Using the efficiently constructible codes of Lemma 9 in place of random codes as outer codes, we can get the constructive analog of Theorem 8 with a similar proof. We also record the statement concerning the span of common subsequences of distinct codewords of our code (which is guaranteed by Theorem 3), as we will make use of this in the next section on efficiently decodable deletion codes.

Theorem 10 (Constructive deletion codes)**.** *Fix an integer $k \geqslant 2$. Then for every real number $\varepsilon > 0$, there is $\tilde{r} = (\varepsilon/k)^{O(\varepsilon^{-3})}$ such that for infinitely many N, we can construct a code $C \subseteq [k]^N$ in time $O(N(\log N)^{\mathrm{poly}(1/\varepsilon)})$ such that (i) C has rate at least $\tilde{r}$ and (ii) $\mathrm{LCS}(C) < \left(\frac{2}{k+1} + \varepsilon\right)N$; in fact if s is a common subsequence of two distinct codewords $c, c' \in C$, then $\operatorname{span} s \geqslant (k+1)\operatorname{len} s - \varepsilon k N$.*

5 Deletion codes with efficient decoding algorithms

We have already shown how to efficiently construct codes over alphabet $[k]$ that are combinatorially capable of correcting a deletion fraction approaching $1 - \frac{2}{k+1}$. However, it is not so clear how to efficiently recover the codes in Theorem 10 from deletions. To this end, we now give an alternate explicit construction by concatenating codes with large distance for the *Hamming metric* with good k-ary deletion codes as constructed in the previous section. As a side benefit, the construction time will be improved as we will need the codes from Theorem 10 for exponentially smaller block lengths.

5.1 Concatenating Hamming metric codes with deletion codes

We state our concatenation result abstractly below, and then instantiate with appropriate codes later for explicit constructions. Recall that the relative distance (in Hamming metric) of a code C of block length n equals the minimum value of $\Delta(c,c')/n$ over all distinct codewords $c, c' \in C$, where $\Delta(x,y)$ denotes the Hamming distance between two words of the same length.

Lemma 11. *Let $\eta, \theta \in (0,1]$. Let $C_{\text{out}} \subseteq [Q]^n$ be code of relative distance in Hamming metric at least $(1-\eta)$. Let $C_{\text{in}} \subseteq [k]^m$ be a code with nQ codewords, one for each $(i,\alpha) \in [n] \times [Q]$, such that for any two distinct codewords $c_1, c_2 \in C_{\text{in}}$ and a common subsequence s of c_1, c_2, we have $\operatorname{span} s \geqslant (k+1)\operatorname{len} s - \theta k m$. Consider the code $C_{\text{concat}} \subseteq [k]^N$ for $N = nm$ obtained as follows: There will be a codeword of C_{concat} for* *each codeword c of C_{out}, obtained by replacing its i'th symbol c_i by the codeword of C_{in} corresponding to (i,c_i). Then we have*

$$\mathrm{LCS}(C) \leqslant \left(\frac{2}{k+1}+2\theta+\eta\right)N\ .$$

Proof. This proof is similar to, but simpler than the proofs of Lemmas 5 and 6. It is simpler because in the present situation a codeword of C_{in} occurs at most once inside a codeword of C_{concat}.

Let c,c' be two distinct codewords of C_{concat} and let σ be a common subsequence of c,c'. Recall that each codeword of C_{concat} can be viewed as a sequence of n (inner) blocks belonging to $[k]^m$, with the i'th block encoding (as per C_{in}) the i'th symbol of the outer codeword. Let us break σ into parts based on which of the n blocks in c,c' its common symbols come from in some canonical (say greedy) way of forming the subsequence σ from c,c'). Let $\sigma_{i,j}$ denote the portion of σ formed by using symbols from the i'th block of c and the j'th block of c'. Let E be the set of pairs (i,j) for which $\sigma_{i,j}$ is not the empty word. If we were to draw words c and c' parallel to each other, and draw the pairs in E as edges, then they would be non-crossing. Therefore, $|E| \leqslant 2n$. Also, by the construction, the only portions $\sigma_{i,j}$ that are formed out of the same codeword of C_{in} are those with $i=j$ and $c_i=c'_i$. Thus there are at most ηn such portions, by the assumed relative distance of C_{out}. Combining all this, we have

$$\begin{aligned}
\operatorname{span}\sigma &\geqslant \sum_{(i,j)\in E}\operatorname{span}\sigma_{i,j}\\
&\geqslant \left(\sum_{(i,j)\in E}\Big((k+1)\operatorname{len}\sigma_{i,j}-\theta km\Big)\right)-(k+1)(\eta n)m\\
&\geqslant (k+1)\operatorname{len}\sigma-2\theta knm-(k+1)\eta nm\ .
\end{aligned}$$

Since $\operatorname{span}\sigma \leqslant 2N$, we have $\operatorname{len}\sigma < \left(\frac{2}{k+1}+2\theta+\eta\right)N$, as desired. ☐

The construction. We now instantiate the above by concatenating Reed–Solomon codes with the codes from Theorem 10. Fix the desired alphabet size $k \geqslant 2$ and $\gamma > 0$.

Let $\mathbb{F}_q$ be a large finite field, an integer $\ell = \lceil \frac{\gamma q}{2} \rceil$. Let C_{out} be the Reed–Solomon encoding code of block length $n=q$ that maps degree $<\ell$ polynomials $f \in \mathbb{F}_q[X]$ to its evaluations at all points in $\mathbb{F}_q$. Note that its relative distance is $(q-\ell+1)/q \geqslant 1-\gamma/2$.

Let C_{in} be a k-ary code with at least q^2 codewords constructed in Theorem 10 for $\varepsilon = \gamma/4$. By the promised rate of that construction, the block length of C_{in} can be taken to be $m \leqslant (k/\gamma)^{O(\gamma^{-3})}\cdot \log q$. Our final construction will apply Lemma 11 to C_{out} and C_{in} with parameters $\eta=\gamma/2$ and $\theta=\gamma/4$, to get a code $C_{\text{concat}} \subseteq [k]^N$ for $N=qm$ with $\mathrm{LCS}(C_{\text{concat}}) \leqslant \left(\frac{2}{k+1}+\gamma\right)N$.

Let us now estimate the construction time. As a function of N, $m \leqslant O_{k,\gamma}(\log N)$, and therefore the construction time for C_{in} becomes $O_{k,\gamma}(\log N(\log\log N)^{\mathrm{poly}(1/\varepsilon)})$. Together with the $q(\log q)^2$ time to construct a representation of $\mathbb{F}_q$ and the Reed–Solomon code, we get an overall construction time of $O(N\log^2 N)$ for large enough N. We record this in the following statement.

Theorem 12 (Reed–Solomon + inner deletion codes with better construction time)**.** *Fix an integer $k \geqslant 2$. Then for every real number $\gamma > 0$, there is $r(k,\gamma) = (\gamma/k)^{O(\gamma^{-3})}$ such that for infinitely many and sufficiently large N, we can construct a code $C \subseteq [k]^N$ in time $O(N\log^2 N)$ such that (i) C has rate at least $r(k,\gamma)$ and (ii)* $\mathrm{LCS}(C) < \left(\frac{2}{k+1}+\gamma\right)N$.

5.2 Deletion correction algorithm

We now describe an efficient decoding procedure for the codes from Theorem 12. The procedure will succeed as long as the fraction of deletions is only slightly smaller than $1-\frac{2}{k+1}$. We describe the basic idea before giving the formal statement and proof. If we are given a subsequence s of length $\left(\frac{2}{k+1}+\delta\right)N$ of some codeword, then by a simple counting argument, there must be at least $\delta q/2$ inner blocks (corresponding to the inner encodings of the q indexed Reed–Solomon symbols) in which s contains at least $\left(\frac{2}{k+1}+\frac{\delta}{2}\right)m$ symbols from the corresponding inner codeword. So we can decode the corresponding Reed–Solomon symbol (by brute-force) if we knew the boundaries of this block. Since we do not know this, the idea is to try decoding all contiguous chunks of size $\left(\frac{2}{k+1}+\frac{\delta}{4}\right)m$ in s with sufficient granularity (for example, subsequences beginning at locations which are multiples of $\delta m/4$).

This might result in the decoding of several spurious symbols, but there will be enough correct symbols to *list decode* the Reed–Solomon code and produce a short list that includes the correct message. By the combinatorial guarantee, only the correct message will have an encoding containing s as a subsequence. Therefore, we can prune the list and identity the correct message by re-encoding each candidate message and checking which one has s has a subsequence. The list decoding step is similar to the one used in [5] for list decoding binary codes from a fraction of deletions approaching $1/2$. Since we have the combinatorial guarantee that the code can correct a deletion fraction $\approx 1-\frac{2}{k+1}$, a list decoding algorithm up to this radius is also automatically a unique decoding algorithm.

Theorem 13 (Explicit and efficiently decodable deletion codes)**.** *The concatenated code $C \subseteq [k]^N$ constructed in Theorem 12 can be efficiently decoded from a fraction $\left(1-\frac{2}{k+1}-O(\gamma^{1/3})\right)$ of worst-case deletions in $N^3(\log N)^{O(1)}$ time, for large enough N.*

Proof. With hindsight, let $\delta = 3\gamma^{1/3}$. Suppose we are given a subsequence s of an *unknown* codeword $c \in C$ (encoding the unknown polynomial f of degree $<\ell$), where $\operatorname{len} s \geqslant \left(\frac{2}{k+1}+\delta\right)N$. We claim that the following decoding algorithm recovers c.

1. $\mathscr{T} \leftarrow \emptyset$.
2. [Inner decodings] For each integer j, $0 \leqslant j \leqslant \frac{\operatorname{len} s}{(\delta m)/4}$, do the following:
   (a) Let σ_j be the contiguous subsequence of s of length $\left(\frac{2}{k+1}+\frac{\delta}{4}\right)m$ starting at position $j\lfloor\frac{\delta m}{4}\rfloor+1$.
   (b) By a brute-force search over $\mathbb{F}_q \times \mathbb{F}_q$, find the unique pair (α,β), if any, such that its encoding under C_{in} has σ_j as a subsequence, and add (α,β) to $\mathscr{T}$. (This pair, if it exists, is unique since $\mathrm{LCS}(C_{\text{in}}) < \left(\frac{2}{k+1}+\frac{\gamma}{4}\right)m$, and $\delta \geqslant \gamma$.)
3. [Reed–Solomon list recovery] Find the list, call it $\mathscr{L}$, of all polynomials $p \in \mathbb{F}_q[X]$ of degree $<\ell$ such that
$$\left|\{(\alpha, p(\alpha)) \mid \alpha \in \mathbb{F}_q\} \cap \mathscr{T}\right| \geqslant \frac{\delta q}{2}\,. \tag{3}$$
4. [Pruning] Find the unique polynomial $f \in \mathscr{L}$, if any, such that its encoding under C contains s as a subsequence, and output f.

CORRECTNESS. Break the codeword $c \in [k]^{nm}$ of the concatenated code C into n (inner) blocks, with the i'th block $b_i \in [k]^m$ corresponding to the inner encoding of the i'th symbol $(\alpha_i, f(\alpha_i))$ of the outer Reed–Solomon codeword. For some fixed canonical way of forming s out of c, denote by s_i the portion of s consisting of the symbols in the i'th block b_i. Call an index $i \in [n]$ *good* if $\operatorname{len} s_i \geqslant \left(\frac{2}{k+1} + \frac{\delta}{2}\right) m$. By a simple counting argument, there are at least $\delta n/2$ values of $i \in [n]$ that are good.

For each good index $i \in [n]$, one of the inner decodings in Step 2 will attempt to decode a subsequence of s_i, and therefore will find the pair $(\alpha_i, f(\alpha_i))$. Since there are at least $\frac{\delta q}{2}$ good indices, the condition (3) is met for the correct f. Using Sudan's list decoding algorithm for Reed–Solomon codes [11], one can find the list of all degree $\leqslant \ell$ polynomials $p \in \mathbb{F}_q[X]$ such that $(\alpha, p(\alpha)) \in \mathscr{T}$ for more than $\sqrt{2\ell|\mathscr{T}|}$ field elements $\alpha \in \mathbb{F}_q$. Further, this list will have at most $\sqrt{2|\mathscr{T}|/\ell}$ polynomials.

Since $|\mathscr{T}| \leqslant 4q/\delta$, if we pick δ so that $\frac{\delta q}{2} > \sqrt{8\ell q/\delta}$, the decoding will succeed. Recalling that $\ell = \lceil \frac{\gamma q}{2} \rceil$, this condition is met for our choice of δ.

RUNTIME. The number of inner decodings performed is $O(q/\delta) = O(N)$, and each inner decoding takes $q^2(\log q)^{O(1)} \leqslant N^2(\log N)^{O(1)})$ time. The set $\mathscr{T}$ has size at most $O(q/\delta) \leqslant O(N)$ for N large enough. The Reed–Solomon list decoding algorithm on $|\mathscr{T}|$ many points can be performed in $O(N^2)$ field operations, see for instance [9]. So the overall running time of the decoder is at most $N^3 \cdot \operatorname{poly}(\log N)$. □

Remark 3. The cubic runtime in the above construction arose because of the brute-force implementation of the inner decodings. One can recursively use the above concatenated codes themselves as the inner codes, in place of the codes from Theorem 10. Each of the inner decodings can now be performed in $\operatorname{poly}(\log q)$ time, for a total time of $N \cdot \operatorname{poly}(\log N)$ for Step 2. By using near-linear time implementations of Reed–Solomon list decoding [1], one can also perform Step 3 in $q \cdot \operatorname{poly}(\log q)$ time. Thus one can improve the decoding complexity to $N \cdot \operatorname{poly}(\log N)$.

References

[1] Michael Alekhnovich. Linear diophantine equations over polynomials and soft decoding of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 51(7):2257–2265, 2005. http://www.math.ias.edu/~misha/papers/diophant.ps.

[2] Boris Bukh and Jie Ma. Longest common subsequences in sets of words. *SIAM J. Discrete Math.*, 28(4):2042–2049, 2014. arXiv:1406.7017.

[3] Boris Bukh and Lidong Zhou. Twins in words and long common subsequences in permutations. *Israel J. Math.*, page (to appear). arXiv:1307.0088.

[4] Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting Gilbert-Varshamov bound for low rates. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 756–757, 2004. https://people.csail.mit.edu/indyk/gvf.ps.

[5] Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. In *Proceeding of RANDOM*, page To appear, 2015. arXiv:1411.6667.

[6] Marcos Kiwi, Martin Loebl, and Jiří Matoušek. Expected length of the longest common subsequence for large alphabets. *Advances in Mathematics*, 197:480–498, November 2004. arXiv:math/0308234.

[7] George S. Lueker. Improved bounds on the average length of longest common subsequences. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 130–131, 2003.

[8] Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009. http://www.eecs.harvard.edu/~michaelm/TALKS/DelSurvey.pdf.

[9] Ron M. Roth and Gitit Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. *IEEE Trans. Inform. Theory*, 46(1):246–257, 2000. http://www.cs.technion.ac.il/~ronny/PUB/rs.pdf.

[10] Leonard Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, November 1999. Preliminary version in SODA 1997; http://authors.library.caltech.edu/6720/1/SCHUieeetit99.pdf.

[11] Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997. http://people.csail.mit.edu/madhu/papers/1996/reeds-journ.pdf.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de