An Improvement to Levenshtein’s Upper Bound on the Cardinality of Deletion Correcting Codes

Daniel Cullina, Student Member, IEEE and Negar Kiyavash, Senior Member, IEEE

Abstract—We consider deletion correcting codes over a q-ary alphabet. It is well known that any code capable of correcting s deletions can also correct any combination of s total insertions and deletions. To obtain asymptotic upper bounds on code size, we apply a packing argument to channels that perform different mixtures of insertions and deletions. Even though the set of codes is identical for all of these channels, the bounds that we obtain vary. Prior to this work, only the bounds corresponding to the all-insertion case and the all-deletion case were known. We recover these as special cases. The bound from the all-deletion case, due to Levenshtein, has been the best known for more than forty-five years. Our generalized bound is better than Levenshtein’s bound whenever the number of deletions to be corrected is larger than the alphabet size.

I. INTRODUCTION

DELETION channels output only a subsequence of their input while preserving the order of the transmitted symbols. Deletion channels are related to synchronization problems, a wide variety of problems in bioinformatics, and the communication of information over packet networks. This paper concerns channels that take a fixed-length input string of symbols drawn from a q-ary alphabet and delete a fixed number of symbols. In particular, we are interested in upper bounds on the cardinality of the largest possible s-deletion correcting codebook.

Levenshtein derived asymptotic upper and lower bounds on the sizes of binary codes for any number of deletions [2]. These bounds easily generalize to the q-ary case [3]. He showed that the Varshamov Tenengolts (VT) codes, which had been designed to correct a single asymmetric error [4], [5], could be used to correct a single deletion. The VT codes establish the asymptotic tightness of the upper bound in the case of a binary alphabet and a single deletion.

Since then, a wide variety of code constructions, which provide lower bounds, have been proposed for the deletion channel and other closely related channels. One recent construction uses constant Hamming weight deletion correcting codes [6]. In contrast, progress on upper bounds has been rare. Levenshtein eventually refined his original asymptotic bound (and the parallel nonbinary bound of Tenengolts) into a nonasymptotic version [7]. Kulkarni and Kiyavash recently proved a better upper bound for an arbitrary number of deletions and any alphabet size [8].

Another line of work has attacked some related combinatorial problems. These include characterization of the sets of superstrings and substrings of any string. Levenshtein showed that the number of superstrings does not depend on the starting string [9]. He also gave upper and lower bounds on the number of substrings using the number of runs in the starting string [2]. Calabi and Hartnett gave a tight bound on the number of substrings of each length [10]. Hirschberg extended the bound to larger alphabets [11]. Swart and Ferreira gave a formula for the number of distinct substrings produced by two deletions for any starting string [12]. Mercier et al showed how to generate corresponding formulas for more deletions and gave an efficient algorithm to count the distinct substrings of any length of a string [13]. Liron and Langberg improved and unified existing bounds and constructed tightness examples [14]. Some of our intermediate results contribute to this area.

A. Upper bound technique

To derive our upper bounds, we use a packing argument that can be applied to any combinatorial channel. Any combinatorial channel can be represented by a bipartite graph. Channel inputs correspond to left vertices, channel outputs correspond to right vertices, and each edge connects an input to an output that can be produced from it. If two channel inputs share a common output, they cannot both appear in the same code. The degree of an input vertex in the graph is the number of possible channel outputs for that input. If the degree of each input is at least r and there are N possible outputs, any code contains at most N/r codewords.

Any code capable of correcting s deletions is also capable of correcting any combination of s total insertions and deletions (See Lemma 2). Despite this equivalence, this packing argument produces different upper bounds for channels that perform different mixtures of insertions and deletions. Let \( C_{q,s,n} \) be the size of the largest q-ary n-symbol s-deletion correcting code. Prior to this work, the bounds on \( C_{q,s,n} \) coming from the s-insertion channel and the s-deletion channel were known.

For the s-insertion channel, each q-ary n-symbol input has the same degree. For fixed q and s, the degree is asymptotic...
be the set of q-ary strings of length n. Let \([q]^n\) be the set of q-ary strings of all lengths. More generally, for a set \(S\), let \(S^n\) be the set of length-\(n\) lists of elements of \(S\) and let \(S^*\) be the set of lists of elements of \(S\) of any length.

We will need the following asymptotic notation: let \(a(n) \sim b(n)\) denote that \(\lim_{n \to \infty} \frac{a(n)}{b(n)} = 1\) and \(a(n) \preceq b(n)\) denote that \(\lim_{n \to \infty} \frac{a(n)}{b(n)} \leq 1\). We will use the following asymptotic equality frequently: for fixed \(c\), \((n^c) \sim \frac{n^c}{\log n}\).

We will encounter several functions that map an element of some set \(X\) to a subset of \(X\). For such a function \(f\) and a subset \(S \subseteq X\), we will use a standard abuse of notation and write \(f(S)\) to mean \(\bigcup_{x \in S} f(x)\).

### B. Deletion and insertion channels

The substring relation is a partial ordering of \([q]^*\). Consequently for strings \(x\) and \(y\), we write \(x \preceq y\) when \(x\) is a substring of \(y\).

We formalize the problem of correcting deletions and insertions by defining the following sets.

**Definition 1.** For \(x \in [q]^n\), define \(D_a(x) = \{z \in [q]^{n-a} : z \preceq x\}\), the set of substrings of \(x\) that can be produced by \(a\) deletions. Define \(I_b(x) = \{w \in [q]^{n+b} : w \succeq x\}\), the set of superstrings of \(x\) that can be produced by \(b\) insertions. Define \(S_{a,b}(x) = I_b(D_a(x))\).

For each input \(x\) to an \(n\)-symbol \(a\)-deletion \(b\)-insertion channel \(S_{a,b}(x)\) is the set of possible outputs. The following well known fact about insertions and deletions shows that the sequencing of insertion and deletion errors does not matter.

**Lemma 1.** For all \(i, a, b \in \mathbb{N}\) and \(x \in [q]^{i+a}\), \(I_b(D_a(x)) = D_a(I_b(x))\). For all \(x \in [q]^{i+a}\) and \(y \in [q]^{i+b}\), \(D_a(x) \cap D_b(y) \neq \emptyset\) if and only if \(I_b(x) \cap I_a(y) \neq \emptyset\).

**Proof:** It is easy to see that \(D_1(I_1(x)) = I_1(D_1(x))\), \(D_1(I_1(x)) = I_1(D_1(x))\), and \(I_1(I_1(x)) = I_1(D_1(x))\). Together these imply \(D_2(I_2(x)) = I_2(D_2(x))\). If \(D_a(x) \cap D_b(y)\) is nonempty, then equivalently

\[\exists z \in D_a(x) : z \subseteq D_b(y)\]

\[\iff \exists z \in D_a(x) : y \subseteq I_b(z)\]

\[\iff \exists y \in I_b(D_a(x))\]

\[\iff \exists y \in D_a(I_b(x))\]

\[\iff \exists w \in I_b(x) : y \subseteq D_a(w)\]

\[\iff \exists w \in I_b(x) : w \subseteq I_b(y)\]

so equivalently \(I_b(x) \cap I_a(y)\) is nonempty.

When two inputs share common outputs they can potentially be confused by the receiver.

**Definition 2.** A q-ary \(n\)-symbol \(a\)-deletion \(b\)-insertion correcting code is a set \(C \subseteq [q]^n\) such that for any two distinct strings \(x, y \in C\), \(S_{a,b}(x) \cap S_{a,b}(y) = \emptyset\).

**Lemma 2.** For \(a, b, n \in \mathbb{N}\) and \(x, y \in [q]^n\). Then \(S_{a,b}(x) \cap S_{a,b}(y) \neq \emptyset\) if and only if \(D_{a+b}(x) \cap D_{a+b}(y) \neq \emptyset\). Consequently a set \(C \subseteq [q]^n\) is a q-ary n-symbol a-deletion b-insertion correcting code if and only if it is an \((a+b)\)-deletion correcting code.
Fig. 1. The strings used in the proof of Lemma 2. Their lengths are given on the left and the lines indicate their substring relationships.

Proof: If $S_{a,b}(x) \cap S_{a,b}(y)$ is nonempty, then equivalently

$$\exists w \in I_b(D_u(x)) \cap I_b(D_v(y))$$

$$\iff \exists u \in D_u(x), v \in D_v(y) : \exists w \in I_b(u) \cap I_b(v)$$

$$\iff \exists u \in D_u(x), v \in D_v(y) : \exists z \in D_b(u) \cap D_b(v)$$

$$\iff \exists z \in D_b(D_u(x)) \cap D_b(D_v(y))$$

so equivalently $D_{a+b}(x) \cap D_{a+b}(y)$ is nonempty. The middle equivalence uses Lemma 1. The relationships between the strings in this proof are illustrated in Figure 1.

III. CONSTRUCTING EDGES

Now we will execute the strategy described in section I-A. The following graph completely describes the behavior of the channel that takes a $q$-ary input string of length $l + a$ and performs $a$ deletions and $b$ insertions.

**Definition 3.** Let $B_{q,l,a,b}$ be a bipartite graph with left vertex set $[q]^{l+a}$ and right vertex set $[q]^{l+b}$. The neighborhood of each left vertex $x$ is $S_{a,b}(x)$.

To obtain our upper bound on code size, we will need a lower bound on the degree of each left vertex of $B_{q,l,a,b}$. The goal of this section is an intermediate result: an asymptotically tight lower bound on the number of edges that matches the lower bound asymptotically. In Section IV, we obtain our lower bound on input degree by working with the edges of $B_{q,l,a,b}$ that are in the image of CONSTRUCT rather than the complete set of edges.

A. The construction and deconstruction procedures

Vertices in $B_{q,l,a,b}$ are adjacent if and only if they have a common substring of length $l$. Because of this, to construct an edge $(x, y) \in E(B_{q,l,a,b})$, we start with a string $z \in [q]^l$. Let $s = a + b$. Partition $z$ into $s + 1$ intervals. To produce $x$, select $a$ of the $s$ boundaries between intervals and insert one new symbol into $z$ at each. To produce $y$, insert one new symbol into $z$ at each of the other $b$ boundaries. Figure 2 gives an example.

This construction procedure is capable of producing every edge in $B_{q,l,a,b}$, but many edges can be produced in multiple ways. We will show that if two restrictions are added to construction procedure, the deconstruction procedure will always be able to recover the construction parameters. This proves that each edge can be produced in at most one way. At the end of the section, we will show that the number of edges that cannot be produced at all under the restrictions is asymptotically negligible.

The first restriction is that each interval must be nonempty and each inserted symbol must differ from the leftmost symbol in the interval to its right. This restriction is needed because inserting a new symbol anywhere within a run of that same symbol has the same effect. Under the restriction, a run in $z$ can only be extended by inserting a matching symbol at the right end.

The second restriction is that each interval of $z$ must be nonalternating.

**Definition 4.** A string is alternating if some $u \in [q]$ appears at all even indices, some $v \in [q]$ appears at all odd indices, and $u \neq v$. A string is nonalternating if it is not alternating. Let $A_{q,n}$ be the set of nonalternating $q$-ary strings and let $A_{q,n}^*$ be those of length $n$.

The empty string and all strings of length one are trivially alternating, so the shortest nonalternating strings have length two. For each length $n \geq 2$, each of the $q$ choices for $u$ and $q - 1$ choices for $v$ results in a unique alternating string, so $|A_{q,n}| = q^n - q(q - 1)$.

Now we describe the deconstruction procedure. Start with an edge $(x, y)$. Beginning at the left, find the longest matching prefix of $x$ and $y$ and delete it from both. This prefix is the first interval of $z$. Now the first symbols of $x$ and $y$ differ. One of these symbols is an insertion, but we do not know which one.

To distinguish these two cases, apply the following heuristic. Provisionally delete the first symbol of $x$ and determine the length of the longest common prefix of $y$ and the rest of $x$. Then do the same with the roles of $x$ and $y$ reversed. Take the longer common prefix to be the next interval of $z$ and the deleted symbol that resulted in this prefix to be the insertion.

After removing this prefix, either the first symbols of $x$ and $y$ again differ or $x$ and $y$ are both the empty string. Apply this heuristic until the latter case is achieved.

B. Formalization of construction and deconstruction

In this section we will define our construction and deconstruction functions more precisely and prove that the
\textsc{Construct}(11, (X, 1, 102) : (Y, 2, 21211) : (X, 2, 021))

\begin{align*}
&= 11 \quad \text{:: Insert}(X, 1, 102) : (Y, 2, 21211) : (X, 2, 021)) \\
&= 11 \quad \text{:: 2102 \quad \text{:: Insert}(Y, 2, 21211) : (X, 2, 021))} \\
&= 11 \quad \text{:: 2102 \quad 21211 \quad \text{:: Insert}(X, 2, 021))} \\
&= 11 \quad \text{:: 2102 \quad 21211 \quad 021} = 1102121121021
\end{align*}

Fig. 3. An example of the construction procedure for a pair of ternary strings. The \textsc{Insert} function is applied to each triple \(((X, Y) \times ([3] \setminus \{0\}) \times A_{\mathbb{N}^*})\) to produce a pair of string segments. \textsc{Construct} concatenates these to produce the final pair.

\paragraph*{Latter inverts the former.} Pseudocode for all of the functions described in this section can be found in Appendix B.

The functions treat strings as lists of symbols. Let \(\epsilon\) be the empty list. Let \(x : y\) be the concatenation of \(x\) and \(y\) and let \((u, v) :: (x, y) = (u : x, v : y)\). Let \(x_0\) be the first symbol of a nonempty string \(x\) and let \(x_{-0}\) be the rest of the string.

To specify an edge \((x, y)\), we need the following parameters. First, we need \(s + 1\) nonalternating strings. When concatenated together, these will form the common substring of \(x\) and \(y\). Second, for each of the \(s\) gaps we pick an element of \(\{X, Y\}\). This specifies which endpoint of the edge will receive the inserted symbol. Finally, to specify each inserted symbol we pick \(\delta \in [q] \setminus \{0\}\). The inserted symbol will be equal to \(\delta\) plus the first symbol of the next interval modulo \(q\).

An example of the construction procedure can be found in Figure 3. The \textsc{Construct} function takes two arguments. The first is \(w_o\), the nonalternating string that will appear at the beginning of both \(x\) and \(y\). Because each inserted symbol depends on the following nonalternating string, we group the remaining construction parameters in triples \(\{X, Y\} \times ([q] \setminus \{0\}) \times A_{\mathbb{N}^*}\). The second argument to \textsc{Construct} is \(t\), a list of \(s\) of these triples. \textsc{Construct} uses the \textsc{Insert} function to turn \(t\) into a pair of strings, then prefixes \(w_s\) to both members of the pair. The \textsc{Insert} function starts by translating \(t_0\) into a pair of strings. If \(w\) is the string from \(t_0\), one of the output strings is \(w\) and the other is \((\delta + w_0) : w\). \textsc{Insert} recursively applies itself to \(t_{-0}\) and concatenates the translation of the first triple to the result the recursive call.

The deconstruction procedure attempts to recover the parameters to \textsc{Construct} from an edge \((x, y)\). An example of the deconstruction procedure can be found in Figure 4. The \textsc{Match} function takes two strings \(x\) and \(y\) and finds their longest common prefix. More precisely, \(\text{Match}(x, y) = (w, (u, v))\) where \(x = w : u\) and \(y = w : v\), and \(w\) is as long as possible. The \textsc{Deconstruct} function uses \textsc{Match} to remove the common prefix of the input strings, then calls \textsc{Delete} on the remaining parts of the input strings. \textsc{Delete} takes a pair of strings \(x\) and \(y\) that are either both empty or are both nonempty and differ in their first symbol. In the former case, there are no more construction parameters to recover so \textsc{Delete} simply returns \(\epsilon\). In the latter case, \textsc{Delete} uses the heuristic described in the previous section to determine which first symbol is an insertion. \textsc{Delete}(\(x, y\)) computes \text{Match}(x_{-0}, y)\) and \text{Match}(x, y_{-0})\). It assumes that whichever common prefix is longer is the nonalternating string used during construction. The information about the deletion and prefix becomes a triple. Finally, \textsc{Deconstruct} recursively applies itself to the leftovers from the chosen match.

Now we will show that \textsc{Deconstruct} is a left inverse of \textsc{Construct}. The main step is to show that \textsc{Delete} can recover the parameters given to \textsc{Insert}.
Lemma 3. For all \( t \in (\{X,Y\} \times ([q] \setminus \{0\}) \times A_{q,s})^* \), \( t = \text{DELETE(INSERT}(t)) \).

Proof: We show this by induction on the length of \( t \). For the base case, \( \text{DELETE(INSERT}(\epsilon) = \text{DELETE}(\epsilon, \epsilon) = \epsilon \).

If \( t \) is nonempty, let \((x,y) = \text{INSERT}(t)\). Either the first symbol of \( t \) is an insertion or the first symbol of \( y \) is. Without loss of generality suppose the former case, so \( t_0 = (X,\delta,w) \) where \( w = (w_0, \ldots, w_{m-1}) \) for some \( m \geq 2 \). Then \((x,y) = (\delta+w_0, \epsilon) \cdot (w,w):= \text{INSERT}(t_0)\). Note that the pair of strings produced by \( \text{INSERT}(t_0) \) are either both empty or both nonempty. If they are nonempty, they have different first symbols.

Recall that \( \text{DELETE} \) computes \( \text{MATCH}(x-0,y) \) and \( \text{MATCH}(x,y-0) \). \( \text{MATCH}(x-0,y) \) always equals \((w,\text{INSERT}(t_0))\). Let \((z,(u,v)) = \text{MATCH}(x,y-0)\).

Suppose that the length of \( z \) is at least \( m-1 \). Then \( z_0 = \delta + w_0 = w_1 \) and for \( 1 \leq i \leq m-2 \), \( z_i = w_{i-1} = w_{i+1} \). This implies that \( w \) is alternating, which is a contradiction.

Thus the length of \( z \) is always less than the length of \( w \) and \( \text{DELETE} \) correctly concludes that the first symbol of \( w \) was the insertion. The length of \( t_0 \) is less than the length of \( t \), so by the induction hypothesis \( \text{DELETE(INSERT}(t_0)) = t_0 \). Finally, \( \text{DELETE}(t) = (X,\delta,w) : t_0 = t_0 : t_0 = t \).

In order for the constructed pair of strings to form an edge in \( B_{q,t,\epsilon,a,b} \), the length of the nonalternating strings must add up to \( l \). Thus each possible vector of string lengths is a composition of \( l \) with \( s + 1 \) parts.

Definition 5. A composition of \( l \) with \( t \) parts is a list of \( t \) nonnegative integers with sum \( l \). Let \( M(t, l, k) \) be the family of compositions of \( l \) with \( t \) parts and each part of size at least \( k \):

\[
M(t, l, k) = \left\{ \lambda \in (\mathbb{N} \setminus [k])^t \mid \sum_{i \in [t]} \lambda_i = l \right\}.
\]

Each element of \( M(t, l, 0) \) can be uniquely represented by a string of \( t \) item symbols and \( t \) dividers symbols: the dividers partition the items into \( t \) groups. Thus \( |M(t, l, 0)| = (l-kt+1) \) and \( |M(t, l, k)| = |M(t, l-kt, 0)| = (l-kt+1) \).

Definition 6. For all \( q, l, a, b \in \mathbb{N} \), let \( s = a + b \). Let \( P_{q,t,\epsilon,a,b} \) be the set

\[
\bigcup_{\lambda \in M(s+1,l,2)} A_{q,\lambda} \times \prod_{i=0}^{s-1} \left( [X,Y] \times ([q] \setminus \{0\}) \times A_{q,\lambda_i} \right)
\]

and let \( P_{q,t,\epsilon,a,b} \) contain the elements of \( P_{q,t,\epsilon,a,b} \) in which \( X \) appears exactly \( a \) times and \( Y \) appears exactly \( b \) times.

The argument of this section is summarized in the following lemma.

Lemma 4. For all \( q, l, a, b \in \mathbb{N} \) and \((w_x,t) \in P_{q,t,\epsilon,a,b} \), let \((x,y) = \text{CONSTRUCT}(w_x, t)\). Then \((x,y) \in E(B_{q,t,\epsilon,a,b})\) and \( |E(B_{q,t,\epsilon,a,b})| \geq |P_{q,t,\epsilon,a,b}| \).

C. Asymptotically matching lower and upper bounds

Lemma 5. For fixed \( q, a, b \in \mathbb{N} \) and \( s = a + b \), \( |P_{q,t,\epsilon,a,b}| \geq q^l \binom{\binom{q}{a}}{s} (q-1)^s \).

Proof: In \( P_{q,t,\epsilon,a,b} \), there are \( \binom{\binom{q}{a}}{s} \) possibilities for the \( s \) elements of \( \{X,Y\} \). There are \( (q-1)^s \) possibilities for the \( s \) elements \([q] \setminus \{0\}) \). For \( \lambda_i \geq 2 \), \( |A_{q,\lambda_i}| = q^{\lambda_i} - q(q-1) \), so the number of possibilities for the \( s + 1 \) alternating strings is

\[
\sum_{\lambda \in M(s+1,l,2)} \prod_{i=0}^{s} \left( q^{\lambda_i} - q(q-1) \right)
\]

\[
\geq \sum_{\lambda \in M(s+1,l,2)} \prod_{i=0}^{s} (q^{\lambda_i} - q^2)
\]

\[
= q^l \prod_{\lambda \in M(s+1,l,2)} \prod_{i=0}^{s} (1 - q^{2-\lambda_i})
\]

\[
\geq q^l \sum_{\lambda \in M(s+1,l,2+\log_q l)} \prod_{i=0}^{s} (1 - q^{2-\lambda_i})
\]

\[
(\lambda) \geq q^l \left( \left( 1 + 2 + \log_q l \right)(s+1) \right) (1-l)^{s+1}
\]

\[
\sim q^l \binom{\binom{q}{a}}{s} (q-1)^s.
\]

In (a), we drop the terms of the sum in which for some \( i \), \( \lambda_i < 2 + \log_q l \). This allows us to apply the inequality \( q^{2-\lambda_i} \leq q^{-\log_q l} \) in (b). We conclude that for \( a \) and \( b \) fixed and \( l \),

\[
|P_{q,t,\epsilon,a,b}| \geq q^l \binom{\binom{q}{a}}{s} (q-1)^s.
\]

Our upper bound will use the following fact about insertions due to Levenshtein [9]. Each \( x \in [q] \) has the same number of superstrings of length \( n \):

\[
|I_{q}(x)| = I_{q,a,n},
\]

where

\[
I_{q,a,n} = \sum_{i=0}^{s} \binom{n}{i} (q-1)^i.
\]

For fixed \( s \) and \( q \), \( I_{q,a,n} \sim \binom{\binom{q}{a}}{s} (q-1)^s \).

Lemma 6. For all \( q, l, a, b \in \mathbb{N} \) with \( s = a + b \), the number \( k \) of edges in \( B_{q,t,\epsilon,a,b} \) satisfies

\[
|E(B_{q,t,\epsilon,a,b})| \leq q^l \prod_{\lambda \in M(s+1,l,2+\log_q l)} \prod_{i=0}^{s} (1 - q^{2-\lambda_i})
\]

\[
\sim q^l \binom{\binom{q}{a}}{s} (q-1)^s.
\]

Proof: There are \( q^l \binom{\binom{q}{a}}{s} (q-1)^s \) triples \((z,x,y) \in [q]^l \times [q]^{l+a} \times [q]^{l+b} \) such that \( z \geq x \) and \( z \geq y \). If \( x \in [q]^{l+a} \) and \( y \in [q]^{l+b} \) are adjacent in \( B_{q,t,\epsilon,a,b} \), then they have at least one common substring of length \( l \) and appear in at least one triple.

Our bounds establish the asymptotic growth of the number of edges.

Theorem 1. For fixed \( q, a, b \in \mathbb{N} \), the number \( k \) of edges in \( B_{q,t,\epsilon,a,b} \) satisfies

\[
|E(B_{q,t,\epsilon,a,b})| \sim q^l \binom{\binom{q}{a}}{s} (q-1)^s.
\]

Proof: From Lemma 4, we have \( |E(B_{q,t,\epsilon,a,b})| \geq |P_{q,t,\epsilon,a,b}| \).

Lemma 5 provides the asymptotic lower bound and Lemma 6 provides the asymptotic upper bound.
For \( x \in [q]^{n} \), \( S_{a,b}(x) \) is the neighborhood of \( x \) in \( B_{q,n-a,a,b} \). The average degree of the left vertices is asymptotic to \( q^{(s)}(q - 1)^{s}/q^n = (r-a)^{s}(q - 1)^{s}q^{-a} \).

IV. BOUNDS ON INPUT DEGREE AND CODE SIZE

**Lemma 7.** Let \( x \in [q]^{n} \) be a string with \( r \) runs. Let \( c - 1 \) be the length of the longest alternating interval of \( x \). Then \( |S_{a,b}(x)| \), the number of unique strings that can be produced from \( x \) by \( a \) deletions and \( b \) insertions, is at least

\[
\left( \frac{r - (a + 1)c - 1}{a} \right) \left( \frac{n - (2a + b + 1)c - 2}{b} \right)(q - 1)^{b}.
\]

**Proof:** To lower bound \( |S_{a,b}(x)| \), we identify a subset \( P_x \subseteq P_{q,n-a,a,b} \) such that for all \( p \in P_x \), \( \text{CONSTRUCT}(p) = (x, y) \). From Lemma 4, all \( y \) produced this way are in \( S_{a,b}(x) \) and \( |S_{a,b}(x)| \geq |P_x| \).

To produce an element of \( P_x \), we select \( a \) symbols of \( x \) for deletion, select \( b \) spaces in \( x \) for insertion, and specify the \( b \) new symbols. The symbols selected for deletion and the symbols for insertion that satisfy this condition.

There are many equivalent ways to extend a run by inserting a matching symbol. \textsc{Construct} extends a run by adding a symbol at the right end, so we only select symbols for deletion from those at the right end of a run. It is easier to ignore the symbols that do not appear at the end of a run for the purpose of spacing as well. We need there to be at least \( c \) symbols in each of the \( a + 1 \) intervals produced by the deletions, but we enforce the stronger condition that in each of these intervals there are at least \( c \) symbols that appear at the end of their run. There are \( M(a + 1, r, c) = (r-a)^{c-1} \) ways to pick the symbols for deletion that satisfy this condition.

There are \( n-1 \) potential spaces in which an insertion can be made. Insertions cannot be performed in the \( c \) spaces before and after a deleted symbol. In the worst case, all of these forbidden spaces are distinct, leaving \( n-1-2ac \) spaces to choose from. There must be \( c \) symbols between any two consecutive chosen spaces, before the first chosen space, and after the last chosen space. Thus there must be at least \( c-1 \) spaces in each of these \( b+1 \) intervals. Again, it is easier to enforce the stronger condition that there are at least \( c+1 \) spaces near a deletion in each interval. Thus there are always at least \( M(b+1, n-1-2ac, c+1) = (n-1-2ac-b+1)c-1) = (n-2a+b+1)c-2 \) ways to pick the spaces.

Finally, for each of the \( b \) insertion points, we must specify the difference between the inserted symbol and its successor. Thus, there are \( (q-1)^{b} \) choices for this step.

The following argument, very similar to Lemma 6, shows that this degree lower bound is asymptotically tight. This is a generalization of a lemma of Levenshtein [2].

**Lemma 8.** For all \( q, n, r, a, b \in \mathbb{N} \) with \( s = a + b \), if \( x \in [q]^{n} \) has \( r \) runs, then

\[
|S_{a,b}(x)| \leq \binom{r + a - 1}{a} I_{q,b,n-a+b}.
\]

**Proof:** Any substring of \( x \) can be specified by the number of symbols deleted from each run. This is a composition of \( a \) with \( r \) parts, so \( |S(a,0)(x)| \leq |M(a, r, 0)| = \binom{r-a}{r-1} = \binom{r-a}{r-1} \). Each string in \( S_{a,b}(x) \) is a superstring of one of these substrings. Each substring has exactly \( I_{q,b,n-a+b} \) superstrings of length \( n - a + b \).

If \( r = pn \) for fixed \( p \), the bounds of Lemma 7 and Lemma 8 are both asymptotic to

\[
\binom{r}{a} \binom{n}{b} (q - 1)^{b}.
\]

To apply Lemma 7 to a string, we need two statistics of that string: the number of runs and the length of the longest alternating interval. The next two lemmas concern the distributions of these statistics.

**Lemma 9.** The number of \( q \)-ary strings of length \( n \) with an alternating interval of length at least \( c \) is at most \( \binom{n}{c} q^{n-c+1} (q-1) \).

**Proof:** If some interval of length at least \( c \) is alternating, at least one of its subintervals of length exactly \( c \) is alternating. A string of length \( n \) contains \( n-c+1 \) intervals of length \( c \), so each string of interest fall into at least one of \( n-c+1 \) classes. In each class, there are \( q^{c} \) choices for the symbols in the alternating interval and \( q^{n-c} \) choices for the remaining symbols.

**Lemma 10.** The number of \( q \)-ary strings of length \( n \) with \( q^{-1}e - (n-1) + \epsilon \) or fewer runs is at most \( q^ne^{-2(n-1)\epsilon^2} \).

**Proof:** For \( x \in [q]^{n} \), let \( x' \in [q]^{n-1} \) be the string of first differences of \( x \). That is, let \( x'_i = x_{i+1} - x_i \mod q \). If \( x \) has \( r \) runs, then \( x'_i \) is nonzero at the \( r-1 \) boundaries between runs. Thus there are \( q^{n-1} \) strings with exactly \( r \) runs.

The number of strings with fewer runs is

\[
q \sum_{i=0}^{n-1} \binom{n-1}{i} \binom{q-1}{i} \left( \frac{1}{q} \right)^{n-1-i} \leq q^ne^{-2(n-1)\epsilon^2}.
\]

The upper bound comes from the application of Hoeffding’s inequality to the binomial distribution [15].

Now we can show that there are few inputs with degree significantly below the average.

**Lemma 11.** Let \( q, a, b \in \mathbb{N} \) be fixed and let \( s = a + b \). For all \( t \in \mathbb{N} \), there is a sequence of subsets \( T_n \subseteq [q]^{n} \) such that \( |T_n| \) is \( O(q^n/n^t) \) and

\[
\min_{x \in [q] \setminus T_n} |S_{a,b}(x)| \geq \frac{(q-1)^{s}}{q^a} \binom{n}{b}.
\]

**Proof:** We form two classes of bad strings: strings with a long alternating interval and strings with few runs. Call these classes \( T^n_a \) and \( T^n_b \) respectively. Let \( T^n_a = T^n_a \cup T^n_b \).

A string falls into \( T^n_a \) if it has an alternating subinterval of length at least \( c \). If we let \( c = t(n+1) \log_q n \), then by Lemma 9 we have

\[
|T^n_a| < nq^{n-c+1}(q-1) = n^{-t}q^{n+1}(q-1)
\]
which is $O(q^n/n^t)$.

Over all strings in $[q]^n$, the average number of runs is
\[
\frac{q-1}{q} (n - 1) + 1.
\]
A string falls into $T''_n$ if it has at most
\[
\left(\frac{q-1}{q} - \epsilon\right)(n-1) + 1 \text{ runs.}
\]
If we let $\epsilon = \sqrt{\frac{\log n}{2(n-1)}}$, then by Lemma 10 we have
\[
|T''_n| \leq q^n e^{-2(n-1)\epsilon^2} = q^n e^{-t\log n} = q^n/n^t.
\]
For fixed $t$, this $\epsilon$ is $o(1)$, so
\[
\left(\frac{q-1}{q} - \epsilon\right)(n-1) + 1 \sim \frac{(q-1)n}{q}.
\]
Now we can apply Lemma 7 to lower bound the degree of the strings in $[q]^n \setminus T_n$. The first multiplicative term in the lower bound is asymptotic to
\[
\left(\frac{q-1}{q} - a(1)(t+1)\log_q n - 1\right) \sim \left(\frac{q-1}{q} - a\right)(n/a).
\]
The second term is asymptotic to
\[
\left(n - (2a + b + 1)(t + 1)\log_q n - 2\right) \sim \left(n/b\right).
\]
Thus
\[
\min_{x \in [q]^n \setminus T_n} |S_{a,b}(x)| \gtrsim \frac{(q-1)^a}{q^a} \left(\frac{n}{b}\right) \left(\frac{q-1}{q}\right) \left(\frac{n}{b}\right) (q-1)^b \sim \frac{(q-1)^a}{q^a} \left(\frac{n}{b}\right) \left(\frac{s}{b}\right).
\]

Proof: We optimize over $b$ in Theorem 2. The factor $(\frac{s}{b})$ is a constant times a binomial distribution:
\[
\left(\frac{q+1}{q}\right)^a \left(\frac{s}{b}\right)^{\frac{1}{q+1}} \left(\frac{q}{q+1}\right)^{q^s-b}.
\]
The maximum is achieved by $b = \left\lfloor \frac{s+1}{q+1} \right\rfloor$. When $q+1$ divides $s$, the maximum is at least
\[
\frac{(q+1)^s}{q^n} \frac{1}{3} \frac{1}{q^s(q+1)} = \frac{(q+1)^{s+1}}{3q^n q^{s+1}}
\]
by Stirling’s approximation. See Appendix A for details.

The degree lower bound in Lemma 11 cannot be raised any further without excluding an asymptotically nonnegligible number of inputs.

**Lemma 12.** For fixed $q, a, b \in \mathbb{N}$ and fixed $\epsilon > 0$, suppose that there is a sequence of subsets $T_n \subseteq [q]^n$ such that
\[
\min_{x \in [q]^n \setminus T_n} |S_{a,b}(x)| \gtrsim (1+\epsilon) \left(\frac{q-1}{q}\right) \left(\frac{n}{s}\right) \left(\frac{s}{b}\right).\]

Then $|T_n|$ is $\Omega(q^n)$.

Proof: This is essentially an application of Markov’s inequality. By the definition of $T_n$,
\[
\sum_{x \in [q]^n \setminus T_n} |S_{a,b}(x)| \geq \min_{x \in [q]^n \setminus T_n} |S_{a,b}(x)| \geq (q^n - |T_n|)(1+\epsilon) \left(\frac{q-1}{q}\right) \left(\frac{n}{s}\right) \left(\frac{s}{b}\right).
\]
From Lemma 6,
\[
\sum_{x \in [q]^n} |S_{a,b}(x)| = |E(B_{q,n-a,a,b})| \lesssim q^{n-a} \left(\frac{n}{s}\right) \left(\frac{s}{b}\right) (q-1)^s.
\]
Chaining these inequalities together and dividing both sides by $q^{n-a} \left(\frac{n}{s}\right) \left(\frac{s}{b}\right) (q-1)^s$ yields
\[
(1 + \epsilon) \left(\frac{q-1}{q}\right) \left(\frac{n}{s}\right) \left(\frac{s}{b}\right) \leq 1.
\]
Thus $|T_n| \gtrsim \frac{1}{1+\epsilon} q^n$ and $|T_n|$ is $\Omega(q^n)$.

If the number of excluded channel inputs is $\Omega(q^n)$, the excluded inputs are the dominant contribution to the upper bound on code size. Thus our bounds on code size are the best that can be obtained via the technique of excluding atypical inputs.

V. CONCLUDING REMARKS

In this paper, we extended Levenshtein’s strategy for obtaining an upper bound on the size of deletion codes. Levenshtein’s bound arises from the deletion channel. We derived the corresponding bounds from channels that perform a mixture of deletions and insertions. This results in an improvement whenever the number of errors, $s$, is larger than the alphabet size, $q$. The best version of our bound uses a channel where the ratio of deletions to insertions is $q$ to one.

Our argument relies on the fact that the channel graphs are approximately regular in the asymptotic regime where the number of errors is fixed. A natural question is whether this argument can be extended to the regime where the number of errors is a constant fraction of the input length. However, it is not clear whether the graphs are approximately regular in the
latter regime. The argument of this paper relies on the typical spacing between errors going to infinity. Because this spacing becomes large, any interaction between two errors becomes rare. When the typical spacing does not grow with input length, interactions will not be rare and it will not be possible to simply discard the cases where they occur. Instead it will be necessary to understand the details of these interactions.

APPENDIX A

One form of Stirling’s approximation is [16]

\[ 1 \leq \frac{n!}{\sqrt{2\pi n} \left(\frac{n}{e}\right)^n} \leq e^{\frac{\alpha}{2n}}. \]

Then for \( \alpha, \beta, n \in \mathbb{N} \), consider the binomial distribution produced by \((\alpha+\beta)n\) trials and success probability \( \alpha/(\alpha+\beta) \). The most likely outcome is \( \alpha n \) successes and the probability of that outcome is:

\[
\begin{align*}
\max_i \left( \alpha+\beta \right) & = \frac{(\alpha+\beta)n}{\alpha} \\
& = \frac{(\alpha+\beta)n}{\alpha} \left( \frac{\alpha}{\alpha+\beta} \right)^i \left( \frac{\beta}{\alpha+\beta} \right)^{\beta n} \\
& \geq \frac{\sqrt{2\pi(\alpha+\beta)n}}{e^{\frac{\alpha}{2n}}} \left( \frac{\alpha}{\alpha+\beta} \right)^{\alpha n} \left( \frac{\beta}{\alpha+\beta} \right)^{\beta n} \\
& = \frac{1}{e^{\frac{\alpha}{2n}}} \sqrt{\alpha+\beta} \frac{1}{\alpha \beta n} \\
& \geq \frac{1}{3} \sqrt{\alpha+\beta} \frac{1}{\alpha \beta n}.
\end{align*}
\]

APPENDIX B

ALGORITHMS

Our construction function, CONSTRUCT, is specified in Algorithm 1 and our deconstruction function, DECONSTRUCT, is specified in Algorithm 2. The function LENGTH returns the number of symbols in the string.

REFERENCES

[1] D. Cullina and N. Kiyavash, “An improvement to Levenshtein’s upper bound on the cardinality of deletion correcting codes,” in IEEE International Symposium on Information Theory Proceedings, Jul. 2013.
[2] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in Soviet physics doklady, vol. 10, 1966, p. 707710.
[3] G. Tenengolts, “Nonbinary codes, correcting single deletion or insertion (corresp.),” IEEE Transactions on Information Theory, vol. 30, no. 5, p. 766769, 1984.
[4] R. Varshamov and G. Tenengolts, “Codes which correct single asymmetric errors,” Avtomatika i Telemehanika, vol. 26, p. 288292, 1965.
[5] R. Varshamov, “On an arithmetic function with an application in the theory of coding,” Doklady Akademi nauk SSSR, vol. 161, p. 540543, 1965.
[6] D. Cullina, A. Kulkarni, and N. Kiyavash, “A coloring approach to constructing deletion correcting codes from constant weight subgraphs,” in IEEE International Symposium on Information Theory Proceedings (ISIT), Jul. 2012, p. 513 517.
[7] V. I. Levenshtein, “Bounds for deletion/insertion correcting codes,” in IEEE International Symposium on Information Theory Proceedings, 2002, p. 370.

Algorithm 1 Construct an edge

```
CONSTRUCT :
A_{q,*} \times (\{X, Y\} \times ([q] \setminus \{0\}) \times A_{q,*})^* \rightarrow [q]^* \times [q]^*
```

CONSTRUCT(w, t)

```
return (w, w) :: INSERT(t)
```

INSERT :

```
\{{X, Y} \times ([q] \setminus \{0\}) \times A_{q,*}}^* \rightarrow [q]^* \times [q]^*
```

INSERT(t)

```
if t = e then
  return (e, e)
else
  (xy, δ, w) ← t₀
  i ← δ + w₀
  if xy = X then
    return (i, e) :: (w, w) :: INSERT(t₀)
  else
    return (i, i) :: (w, w) :: INSERT(t₀)
end if
```

[8] A. A. Kulkarni and N. Kiyavash, “Non-asymptotic upper bounds for deletion correcting codes,” IEEE Transactions on Information Theory, 2012. [Online]. Available: http://arxiv.org/abs/1211.3128
[9] V. I. Levenshtein, “Elements of coding theory,” Diskretnaya matematika i matematicheskie voprosy kibernetiki, 1978. 1979.
[10] L. Calabi and W. E. Hartnett, “Some general results of coding theory with applications to the study of codes for the correction of synchronization errors,” Information and Control, vol. 15, no. 3, p. 235249, 1969.
[11] D. Hirschberg, “Bounds on the number of string subsequences,” in Combinatorial Pattern Matching, 1999, p. 115122.
[12] T. G. Swart and H. C. Ferreira, “A note on double insertion/deletion correcting codes,” IEEE Transactions on Information Theory, vol. 49, no. 1, p. 269273, 2003.
[13] H. Mercier, M. Khabbazian, and V. Bhargava, “On the number of subsequences when deleting symbols from a string,” IEEE Transactions on Information Theory, vol. 54, no. 7, pp. 3279–3285, 2008.
[14] Y. Liron and M. Langberg, “A characterization of the number of subsequences obtained via the deletion channel,” in IEEE International Symposium on Information Theory Proceedings, 2012, p. 503507.
[15] W. Hoeffding, “Probability inequalities for sums of bounded random variables,” Journal of the American Statistical Association, vol. 58, no. 301, pp. 13–30, Mar. 1963.
[16] W. Feller, “Stirling’s formula,” in An introduction to probability theory and its applications, 3rd ed. John Wiley & Sons, 1968, vol. 1, pp. 52–54.

Daniel Cullina (M’13) received the B.S. degree in electrical engineering form the California Institute of Technology in 2010. He is presently a graduate student in the Electrical and Computer Engineering Department at the University of Illinois at Urbana-Champaign. His research interests include coding theory and combinatorics.

Negar Kiyavash (SM’13) received the B.S. degree in electrical and computer engineering from the Sharif University of Technology, Tehran, in 1999, and the M.S. and Ph.D. degrees, also in electrical and computer engineering, both from the University of Illinois at Urbana-Champaign in 2003 and 2006, respectively. She is presently an Assistant Professor in the Electrical and Computer Engineering Department at the University of Illinois at Urbana-Champaign. Her research interests are in information theory and statistical signal processing with applications to computer, communication, and multimedia security.
Algorithm 2 Deconstruct an edge

**DECONSTRUCT :**

\[ [q]^* \times [q]^* \rightarrow A_{q, \epsilon} \times (\{X, Y\} \times ([q] \setminus \{0\}) \times A_{q, \epsilon})^* \]

**DECONSTRUCT**(x, y)

\[
(w, (x, y)) \leftarrow \text{MATCH}(x, y)
\]

\[
\text{return } (w, \text{DELETE}(x, y))
\]

**DELETE :**

\[ [q]^* \times [q]^* \rightarrow ([X, Y] \times ([q] \setminus \{0\}) \times A_{q, \epsilon})^* \]

**DELETE**(x, y)

\[
\text{if } x = \epsilon \lor y = \epsilon \text{ then}
\]

\[
\text{assert } x = \epsilon \land y = \epsilon
\]

\[
\text{return } \epsilon
\]

\[
\text{else}
\]

\[
g \leftarrow x_0 - y_0
\]

\[
(a, (b, c)) \leftarrow \text{MATCH}(x_{-0}, y)
\]

\[
(d, (e, f)) \leftarrow \text{MATCH}(x, y_{-0})
\]

\[
\text{assert } \text{LENGTH}(a) \neq \text{LENGTH}(d)
\]

\[
\text{if } \text{LENGTH}(a) > \text{LENGTH}(d) \text{ then}
\]

\[
\text{return } (X, g, a) : \text{DELETE}(b, c)
\]

\[
\text{else}
\]

\[
\text{return } (Y, -g, d) : \text{DELETE}(e, f)
\]

\[
\text{end if}
\]

**end if**

**MATCH :**

\[ [q]^* \times [q]^* \rightarrow [q]^* \times ([q]^* \times [q]^*) \]

**MATCH**(x, y)

\[
w \leftarrow \epsilon
\]

\[
\text{while } x \neq \epsilon \land y \neq \epsilon \land x_0 = y_0 \text{ do}
\]

\[
w \leftarrow w : x_0
\]

\[
x \leftarrow x_{-0}
\]

\[
y \leftarrow y_{-0}
\]

\[
\text{end while}
\]

\[
\text{return } (w, (x, y))
\]