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

Daniel Cullina
Dep. of Electrical & Computer Eng.
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801
cullina@illinois.edu

Negar Kiyavash
Dep. of Industrial and Enterprise Systems Eng.
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801
kiyavash@illinois.edu

***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 [5]. These bounds easily generalize to the q-ary case [11]. He showed that the Varshamov Tenengolts (VT) codes, which had been designed to correct a single asymmetric error [12], [13], 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 constructing codes [2]. 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 [4].

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 [6]. He also gave upper and lower bounds on the number of substrings using the number of runs in the starting string [5]. Calabi and Hartnett gave a tight bound on the number of substrings of each length [1]. Hirschberg extended the bound to larger alphabets [3]. Swart and Ferreira gave a formula for the number of distinct substrings produced by two deletions for any starting string [10]. 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 [9]. Liron and Langberg improved and unified existing bounds and constructed tightness examples [8]. 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 1). Despite this equivalence, this packing argument produces different upper bounds for channel 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 to $\binom{n}{s}(q-1)^s$ [6]. There are q^{n+s} possible outputs, so

$$C_{q,s,n} \lesssim \frac{q^{n+s}}{\binom{n}{s}(q-1)^s}. \qquad (1)$$

The s-deletion case is slightly more complicated because different inputs have different degrees. For instance, the input strings consisting of a single symbol repeated n times have only a single possible output: the string with that symbol repeated $n-s$ time. Consequently, using the minimum degree over all of the inputs yields a worthless bound. Using the following argument [5], Levenshtein showed that

$$C_{q,s,n} \lesssim \frac{q^{n}}{\binom{n}{s}(q-1)^s}. \qquad (2)$$

The average degree of an input is asymptotic to $\left(\frac{q-1}{q}\right)^s \binom{n}{s}$ and most inputs have a degree close to that. The inputs can be divided into two classes: those with degree at least $1-\epsilon$ times the average degree and those with smaller degree. For an appropriately chosen ϵ that goes to zero as n goes to infinity, the vast majority of inputs fall into the former class. Call members of the former class the typical inputs. The minimum degree argument can be applied to bound the number of typical inputs that can appear in a code. There are q^{n-s} possible outputs, so the number of typical inputs in a code is asymptotically at most (2). We have no information about what the fraction of the atypical inputs can appear in a code, but the total number of atypical inputs is small enough to not affect the asymptotics of the upper bound.

The bounds (1) and (2) have the same growth rates, but the bound on deletion correcting codes is a factor of q^s better than the bound on insertion correcting codes, despite the fact that any s-deletion correcting code is an s-insertion correcting code and vice versa. Note that there is no possible improvement to the insertion channel bound from dividing the inputs into typical and atypical classes.

We extend this bounding strategy to channels that perform both deletions and insertions. We obtain a generalized upper bound that includes Levenshtein's bound as a special case. Recall that Levenshtein's bound is known to be tight for one deletion and alphabet size two. The new bound improves upon the Levenshtein's bound whenever the number of deletions is greater than the alphabet size.

The rest of the paper is organized as follows. In Section II, we present some notation and basic results on deletion and insertion channels. In Section III, we construct a class of well-behaved edges in the channel graph. In Section IV, we prove a lower bound on the degree of each input vertex and use it to establish our main result: an upper bound on the size of a q-ary s-deletion correcting code. Due to space contraints, we omit most of our proofs.

II. Preliminaries

A. Notation

Let $\mathbb{N}$ be the set of nonnegative integers. Let $[n]$ be the set of nonnegative integers less than n, $\{0,1,\ldots,n-1\}$. Let $[q]^n$ be the set of q-ary strings of length n. Let $[q]^*$ be the set of q-ary strings of all lengths. More generally, for a set S, let S^n be the set of lists of elements S of length n 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 $a(n) \lesssim b(n)$ denote that $\lim_{n\to\infty} \frac{a(n)}{b(n)} \leq 1$. We will use the following asymptotic equality frequently: for fixed c, $\binom{n}{c} \sim \frac{n^c}{c!}$.

B. Deletion distance

The substring relation is a partial ordering of $[q]^*$. Consequently for strings x and y, we write $x \leq y$ if 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 $S_{a,0}(x) = \{z \in [q]^{n-a} : z \leq x\}$, the set of substrings of x that can be produced by a deletions. Define $S_{0,b}(x) = \{w \in [q]^{n+b} : w \geq x\}$, the set of superstrings of x that can be produced by b insertions. Define $S_{a,b}(x) = \bigcup_{z\in S_{a,0}(x)} S_{0,b}(z)$.*

The a-deletion b-insertion channel takes a string of length n, finds a substring of length $n-a$, and outputs a superstring of that substring of length $n-a+b$. Consequently, 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 graph completely describes the behavior of the $(l+a)$-symbol a-deletion b-insertion channel.

Definition 2. *Let $B_{q,l,a,b}$ be a bipartite graph with left vertex set $[q]^{l+a}$ and right vertex set $[q]^{l+b}$. Vertices are adjacent if they have a common substring of length l.*

If x is a left vertex of $B_{q,l,a,b}$, then its neighborhood is $S_{a,b}(x)$. When two inputs share common outputs they can potentially be confused by the receiver.

Definition 3. *A q-ary n-symbol a-deletion b-insertion correcting code is a set $C \subset [q]^n$ such that for any two distinct strings $x, y \in C$, $S_{a,b}(x) \cap S_{a,b}(y)$ is empty.*

Definition 4. *For $x \in [q]^n$ and $y \in [q]^m$, define the deletion distance between them to be $d_L(x,y) = n+m-2l$, where l is the length of their longest common substring.*

Lemma 1. *For $a,b,n \in \mathbb{N}$, $x,y \in [q]^n$, $S_{a,b}(x) \cap S_{a,b}(y) = \varnothing$ if and only if $d_L(x,y) > 2(a+b)$. Consequently a set $C \subset [q]^n$ is a q-ary n-symbol a-deletion b-insertion correcting code if and only if for all distinct $x,y \in C$, $d_L(x,y) > 2(a+b)$.*

III. Constructing edges

To execute the strategy described in section I-A, we need a lower bound on the degree of each channel input. This is a lower bound on the degree of each left vertex of $B_{q,l,a,b}$. To obtain this bound, we first construct a subset of the edges of $B_{q,l,a,b}$ that is easier to work with than the complete edge set.

One way to get information about the size of a target set T is to find a construction function $f : P \to T$, where P is an easily counted parameter set. If f is injective, then $|P| =$

$$
\begin{array}{l}
\textsc{Construct}(11,(L,1,102),(R,2,21211)) \\
\quad \textsc{Insert}(L,1,102) \quad = \begin{array}{c} 2102 \\ 102 \end{array} \\
\quad \textsc{Insert}(R,2,21211) = \begin{array}{c} 21211 \\ 121211 \end{array} \\
= \begin{array}{ccc} 11 & 2102 & 21211 \\ 11 & 102 & 121211 \end{array} = \begin{array}{c} 11210221211 \\ 11102121211 \end{array}
\end{array}
$$

Fig. 1. An example of the construction procedure for a pair of strings. The INSERT function is applied to each triple ($\mathrm{LR} \times ([q] \setminus \{0\}) \times [q]^*$) to produce a pair of string segments. CONSTRUCT concatenates these.

$|f(P)|$ and $|P| \leq |T|$. We can demonstrate the injectivity of f with a deconstruction function $g : T \to P$ that is a left inverse of f. This means that $g(f(p)) = p$ for all $p \in P$. If the function g is given a constructible member of T, g recovers the construction parameters that produce it. In this section we apply this method to the edge set of $B_{q,l,a,b}$.

To construct an edge $(x, y) \in E(B_{q,l,a,b})$, start with a string $z \in [q]^l$ which will be a substring of both endpoints of the edge. Let $s = a+b$. Partition z into $s+1$ nonempty 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.

Each way to partition z corresponds to 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: $M(t, l, k) = \left\{ \lambda \in (\mathbb{N} \setminus [k])^t \,\middle|\, \sum_{i \in [t]} \lambda_i = l \right\}$.*

A standard argument shows that $|M(t, l, k)| = \binom{l-kt+t-1}{t-1}$.

The parameter set for the construction described above is $[q]^l \times M(s+1, l, 1) \times \binom{[s]}{a} \times [q]^s$ where $\binom{[s]}{a}$ is the family of a element subsets of $[s]$. The size of this set is $\binom{l-1}{s}\binom{s}{a}q^{l+s}$.

It is clear that there are many edges that this construction produces multiple times. We will show that if the following two restrictions are added to construction procedure, each edge will be produced at most once:

- Each inserted symbol must differ from the leftmost symbol in the interval to its right.
- Each interval of z must be nonalternating.

The first restriction is well posed because the intervals are nonempty. 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. To implement this restriction, for each insertion point we pick $\delta \in [q] \setminus \{0\}$ and insert δ plus the next symbol.

Definition 6. *A string is alternating if some $u \in [q]$ appears at all even indices, some $v \in [q]$ appears at all odd indices, and $u \neq v$. Let $A_{q,n}$ be the set of nonalternating q-ary strings of length n.*

$$
\begin{array}{l}
\textsc{Deconstruct}\begin{pmatrix} 11210221211 \\ 11102121211 \end{pmatrix} \\
\quad \textsc{Match}\begin{pmatrix} 11210221211 \\ 11102121211 \end{pmatrix} = 11 \; \begin{array}{c} 210221211 \\ 102121211 \end{array} \\
\quad \textsc{Delete}\begin{pmatrix} 210221211 \\ 102121211 \end{pmatrix} \\
\qquad \textsc{Match}\begin{pmatrix} 10221211 \\ 102121211 \end{pmatrix} = 102 \; \begin{array}{c} 21211 \\ 121211 \end{array} \checkmark \\
\qquad \textsc{Match}\begin{pmatrix} 210221211 \\ 02121211 \end{pmatrix} = \quad \epsilon \; \begin{array}{c} 210221211 \\ 02121211 \end{array} \\
\quad = (L,1,102) \; \begin{array}{c} 21211 \\ 121211 \end{array} \\
\quad \textsc{Delete}\begin{pmatrix} 21211 \\ 121211 \end{pmatrix} \\
\qquad \textsc{Match}\begin{pmatrix} 1211 \\ 121211 \end{pmatrix} = \quad 121 \; \begin{array}{c} 1 \\ 211 \end{array} \\
\qquad \textsc{Match}\begin{pmatrix} 21211 \\ 21211 \end{pmatrix} = 21211 \; \begin{array}{c} \epsilon \\ \epsilon \end{array} \checkmark \\
\quad = (R,2,21211) \; \begin{array}{c} \epsilon \\ \epsilon \end{array} \\
= 11, (L,1,102), (R,2,21211)
\end{array}
$$

Fig. 2. An example of the deconstruction process. First, MATCH strips off the common prefix. The DELETE function tests whether it a longer common prefix is achieved by deleting the head of the first string or the second string. The check marks indicate the longer match. DELETE produces a triple specifying that deletion and prefix.

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

To explain the purpose of the second restriction, we must first 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 part of the next interval of z and the other was an insertion, but we do not know which is which.

To resolve this situation, apply the following heuristic. 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. Assume that the deleted symbol that resulted in the longer common prefix was the insertion and that the longer prefix was the next interval of z. 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.

We will show that this heuristic is always correct when applied to edges produced under the second restriction.

Algorithm 1 Construct an edge

```
CONSTRUCT : $[q]^* \times (\mathrm{LR} \times ([q] \setminus \{0\}) \times [q]^*)^s \to [q]^* \times [q]^*$
CONSTRUCT($w_0, t$)
    $(x, y) \leftarrow \mathrm{C}(t)$
    return $(w_0 : x, w_0 : y)$

C : $(\mathrm{LR} \times ([q] \setminus \{0\}) \times [q]^*)^s \to [q]^* \times [q]^*$
C($t$)
    if $t = \epsilon$ then
        return $(\epsilon, \epsilon)$
    else
        $(u, v) \leftarrow$ INSERT(HEAD($t$))
        $(x, y) \leftarrow$ C(TAIL($t$))
        return $(u : x, v : y)$
    end if

INSERT : $\mathrm{LR} \times ([q] \setminus \{0\}) \times [q]^* \to [q]^* \times [q]^*$
INSERT($lr, \delta, w$)
    $w' \leftarrow (\delta +$ HEAD($w$)$) : w$
    if $lr =$ LEFT then
        return $(w', w)$
    else
        return $(w, w')$
    end if
```

Algorithm 2 Deconstruct an edge

```
DECONSTRUCT : $[q]^* \times [q]^* \to [q]^* \times (\mathrm{LR} \times ([q] \setminus \{0\}) \times [q]^*)^s$
DECONSTRUCT($x, y$)
    $(w_0, x, y) \leftarrow$ MATCH($x, y$)
    return $(w_0, \mathrm{D}(x, y))$

D : $[q]^* \times [q]^* \to (\mathrm{LR} \times ([q] \setminus \{0\}) \times [q]^*)^s$
D($x, y$)
    if $x = \epsilon \vee y = \epsilon$ then
        assert $x = \epsilon \wedge y = \epsilon$
        return $\epsilon$
    else
        $(w, x, y) \leftarrow$ DELETE($x, y$)
        return $(w : \mathrm{D}(x, y))$
    end if

DELETE : $[q]^* \times [q]^* \to (\mathrm{LR} \times ([q] \setminus \{0\}) \times [q]^*) \times [q]^* \times [q]^*$
DELETE($x, y$)
    $g =$ HEAD($x$) $-$ HEAD($y$)
    $(a, b, c) \leftarrow$ MATCH(TAIL($x$)$, y$)
    $(d, e, f) \leftarrow$ MATCH($x,$ TAIL($y$))
    assert LENGTH($a$) $\neq$ LENGTH($d$)
    if LENGTH($a$) $>$ LENGTH($d$) then
        return $(($LEFT$, g, a), b, c)$
    else
        return $(($RIGHT$, (-g), d), e, f)$
    end if

MATCH : $[q]^i \times [q]^j \to [q]^k \times [q]^{i-k} \times [q]^{j-k}$
MATCH($x, y$)
    $w \leftarrow \epsilon$
    while $x \neq \epsilon \wedge y \neq \epsilon \wedge$ HEAD($x$) $=$ HEAD($y$) do
        $w \leftarrow w :$ HEAD($x$)
        $x \leftarrow$ TAIL($x$)
        $y \leftarrow$ TAIL($y$)
    end while
    return $(w, x, y)$
```

Our construction function, CONSTRUCT, is specified in Algorithm 1 and our deconstruction function, DECONSTRUCT, is specified in Algorithm 2. Example of the construction and deconstruction algorithms are provided in Figures 1 and 2.

The functions treat strings as lists of symbols. We write the empty list as ϵ. We write the concatenation of x and y as $x\,{:}\,y$. The function HEAD returns the first symbol of a nonempty list, TAIL returns everything except the head, and LENGTH returns the number of symbols in the string.

Lemma 2. *For $lr \in$ LR, $\delta \in [q] \setminus \{0\}$, and $w \in A_{q,m}$, let $(x, y) =$ INSERT(lr, δ, w). Let $u, v \in [q]^*$ such that if both are nonempty, they have different first symbols. Then* DELETE$(x : u, y : v) = ((lr, \delta, w), u, v)$.

Definition 7. *For all $q, l, a, b \in \mathbb{N}$, let $s = a + b$. Let $P_{q,l,s}$ be*

$$\bigcup_{\mathbf{c} \in M(s+1,l,2)} A_{q,c_0} \times \prod_{i=1}^{s} (\mathrm{LR} \times ([q] \setminus \{0\}) \times A_{q,c_i})$$

and let $P_{q,l,a,b}$ be the subset of $P_{q,l,s}$ with exactly a appearances of LEFT.

Lemma 3. *For all $q, l, s \in \mathbb{N}$ and $p \in P_{q,l,s}$,* DECONSTRUCT(CONSTRUCT$(p)) = p$.

Lemma 4. *For all $q, l, a, b \in \mathbb{N}$, $s = a + b$, and $p \in P_{q,l,a,b}$,* CONSTRUCT$(p) \in E(B_{q,l,a,b})$.

IV. Bounds on Input Degree and Code Size

Lemma 5. *Let $x \in [q]^n$ be a string with r runs. Let c 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*

$$\binom{r - (a+1)(c+1)}{a} \binom{n - 1 - 2a(c+1) - (b+1)c}{b} (q-1)^b.$$

Proof: For each $x \in [q]^n$, we identify a subset $P_x \subseteq P_{q,n-a,a,b}$ such that for all $p \in P_x$, CONSTRUCT$(p) = (x, y)$. From Lemma 4, all y produced this way are in $S_{a,b}(x)$. From Lemma 3, $|S_{a,b}(x)| \geq |P_x|$.

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 of deletion and the spaces selected for insertion partition x into $s+1$ intervals. To ensure that none of these intervals are alternating, we will require that all of the intervals contain at least $c + 1$ symbols.

There are many equivalent ways to extends a run by inserting a matching symbol. CONSTRUCT extends runs by adding a symbol at the right end, so we only select symbols for deletion from those at the right end of a run. We need there to be at least $c+1$ symbols between consecutive deleted symbols. It is easier to enforce the stronger condition that there are $c+1$ end of run symbols between consecutive deleted symbols. There are $\binom{r-(a+1)(c+1)}{a}$ 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 $c+1$ spaces before and after a deleted symbol. In the worst case, all of these forbidden spaces are distinct, leaving $n-1-2a(c+1)$ spaces to choose from. There must be $c+1$ 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 spaces in each of these $b+1$ intervals. Again, it is easier to enforce the stronger condition that there are at least c spaces not near a deletion in each interval. Thus there are always at least $\binom{n-1-2a(c+1)-(b+1)c}{b}$ ways to pick the spaces.

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

To apply Lemma 5 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 6. *The number of q-ary strings of length n with an alternating interval of length at least c is at most $(n-c+1)q^{n-c+1}(q-1)$.*

Lemma 7. *The number of q-ary strings of length n with $\left(\frac{q-1}{q}-\epsilon\right)(n-1)+1$ or fewer runs is at most $q^n e^{-2(n-1)\epsilon^2}$.*

Now we have all of the ingredients required to execute the strategy described in Section I-A.

Theorem 1. *For fixed $q,a,b \in \mathbb{N}$ and $s=a+b$, the number of codewords in an n-symbol q-ary a-deletion b-insertion correcting code is asymptotically at most $q^{n+b}/(q-1)^s\binom{n}{s}\binom{s}{b}$.*

Proof: There are q^{n-a+b} possible outputs, so for any $T_n \subseteq [q]^n$,

$$C_{q,s,n} \lesssim \frac{q^{n-a+b}}{\min_{x\in[q]^n\setminus T_n}|S_{a,b}(x)|} + |T_n|.$$

We form two classes of bad strings: strings with a long alternating interval, and strings with few runs. Call these classes T'_n and T''_n respectively. Let $T_n = T'_n \cup T''_n$.

A string falls into T'_n if it has an alternating subinterval of length at least c. If we let $c=(t+1)\log_q n$, then by Lemma 6 we have $|T'_n| < nq^{n-c+1}(q-1) = n^{-t}q^{n+1}(q-1)$.

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 $\left(\frac{q-1}{q}-\epsilon\right)(n-1)+1$ runs. If we let $\epsilon=\sqrt{\frac{t\log n}{2(n-1)}}$, then by Lemma 7 we have $|T''_n| \le q^n e^{-2(n-1)\epsilon^2} = q^n/n^t$. For fixed t, this ϵ 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 5 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 $\binom{\frac{q-1}{q}n-(a+1)((t+1)\log_q n+1)}{a} \sim \left(\frac{q-1}{q}\right)^a\binom{n}{a}$. The second term is asymptotic to $\binom{n-1-2a-(2a+b+1)(t+1)\log_q n}{b} \sim \binom{n}{b}$. Thus

$$\min_{x\in[q]^n\setminus T_n}|S_{a,b}(x)| \gtrsim \left(\frac{q-1}{q}\right)^a\binom{n}{a}\binom{n}{b}(q-1)^b$$

By setting $t=s+1$ we obtain an asymptotic upper bound of

$$C_{q,s,n} \lesssim \frac{q^{n-a+b}}{\frac{(q-1)^s}{q^a}\binom{n}{s}\binom{s}{b}} + O\left(\frac{q^n}{n^{s+1}}\right) \sim \frac{q^{n+b}}{(q-1)^s\binom{n}{s}\binom{s}{b}}.$$

■

This improves (2), Levenshtein's upper bound, by a factor of $\binom{s}{b}q^{-b}$. By setting b to zero we recover Levenshtein's bound. Whenever $s>q$, $\binom{s}{1}q^{-1} > \binom{s}{0}q^0 = 1$ so setting b to one in the generalized bound offers an improvement.

ACKNOWLEDGMENT

This work was supported in part by AFOSR under grants FA 9550-11-1-0016 and FA 9550-10-1-0573; and by NSF grant CCF 10-54937 CAR.

REFERENCES

[1] 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.

[2] 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)*, July 2012, p. 513 517.

[3] D. Hirschberg, "Bounds on the number of string subsequences," in *Combinatorial Pattern Matching*, 1999, p. 115122.

[4] 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

[5] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, 1966, p. 707710.

[6] ——, "Elements of coding theory," *Diskretnaya matematika i matematicheskie voprosy kibernetiki*, p. 207305, 1974.

[7] ——, "Bounds for deletion/insertion correcting codes," in *IEEE International Symposium on Information Theory Proceedings*, 2002, p. 370.

[8] 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.

[9] 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.

[10] 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.

[11] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion (corresp.)," *IEEE Transactions on Information Theory*, vol. 30, no. 5, p. 766769, 1984.

[12] R. Varshamov, "On an arithmetic function with an application in the theory of coding," *Doklady Akademii nauk SSSR*, vol. 161, p. 540543, 1965.

[13] R. Varshamov and G. Tenengolts, "Codes which correct single asymmetric errors," *Avtomatika i Telemekhanika*, vol. 26, p. 288292, 1965.