View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Johannesburg Institutional Repository

Codes for Correcting Three or More Adjacent Deletions or Insertions

Ling Cheng
University of the Witwatersrand
Johannesburg, South Africa
Email: ling.cheng@wits.ac.za

Theo G. Swart and Hendrik C. Ferreira
University of Johannesburg
Auckland Park 2006, South Africa
Email: {tgswart,hcferreira}@uj.ac.za

Khaled A. S. Abdel-Ghaffar
University of California
Davis, CA 95616, USA
E-mail: ghaffar@ece.ucdavis.edu

***Abstract*—Codes are presented that can correct the deletion or the insertion of a predetermined number of adjacent bits greater than or equal to three. This extends the constructions of codes beyond those proposed by Levenshtein fifty years ago to correct one or two adjacent deletions or insertions.**

I. Introduction

Loss of synchronization due to uncertainties in timing is a problem in communication and storage systems. It typically manifests itself in the deletion or the insertion of spurious symbols. Should the receiver fails to recover synchronization, a catastrophic failure in decoding the received symbols follows. In 1965, Levenshtein [5] has demonstrated that a class of codes constructed earlier by Varshamov and Tenengol'ts [7] to correct a single asymmetric error can also be used to correct a single deletion or insertion. Two years later, Levenshtein presented asymptotic upper bounds on the sizes of codes that can correct the deletion or the insertion of exactly a given number of adjacent deletions and insertions [6]. In the same paper, he constructed asymptotically optimal codes that can correct up to two adjacent deletions or insertions. In this paper, we present three classes of codes that can correct exactly a given number, greater or equal to three, of adjacent deletions and insertions.

Let $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ be a binary sequence of length N transmitted over a channel. If the received sequence is $\mathbf{r} = (r_1, r_2, \ldots, r_{N-s})$, where

$$r_\ell = \begin{cases} c_\ell & \text{for } \ell < \ell^*, \\ c_{\ell+s} & \text{for } \ell \geq \ell^*, \end{cases} \tag{1}$$

then we say that the channel caused s adjacent deletions in $\mathbf{c}$ starting at index ℓ^*, where $1 \leq \ell^* \leq N - s$. On the other hand, if the received word is $\mathbf{r} = (r_1, r_2, \ldots, r_{N+s})$, where

$$r_\ell = \begin{cases} c_\ell & \text{for } \ell < \ell^*, \\ c_{\ell-s} & \text{for } \ell \geq \ell^* + s, \end{cases}$$

and r_ℓ, for $\ell^* \leq \ell < \ell^* + s$, can assume any value in $\{0, 1\}$, then we say that the channel caused s adjacent insertions starting at index ℓ^*, where $1 \leq \ell^* \leq N + 1$.

In this paper, we consider codes that can correct a single deletion or insertion of exactly (rather than at most) s adjacent symbols. It should be emphasized that codes correcting exactly s adjacent symbols may fail to correct fewer number of adjacent symbols. For example, $(0, 1, 0, 1)$ and $(1, 0, 1, 0)$ can appear together as codewords in a code that corrects two adjacant deletions but not in a code that corrects a single deletion. We assume that the beginning and the end of each received sequence corresponding to a transmitted codeword are known, which allows for independent decoding of the codewords. (This assumption, which is commonly assumed in the literature, can be achieved by inserting periodic markers between codewords.) A code is s-adjacent deletion (insertion, resp.) correcting if the receiver can retrieve the transmitted codeword provided that the channel causes s adjacent bits to be deleted (inserted, resp.). This is equivalent to saying that no sequence can be obtained from two distinct codewords in the code by deleting (inserting, resp.) s adjacent bits from (in, resp.) each. A classical result by Levenshtein [5] states that a code is deletion correcting if and only if it is insertion correcting. Based on this, it suffices to consider only s-adjacent deletion correcting codes since these codes are also s-adjacent insertion correcting codes in the sense that they can correct either the deletion or the insertion of s-adjacent bits.

In 1967, Levenshtein [6] has shown that the number of codewords in an s-adjacent deletion correcting code of length N is asymptotically upper bounded by $2^{N-s+1}/N$. He also constructed codes that meet this upper bound for $s = 2$. The Varshamov-Tenengol'ts codes meet the bound for $s = 1$. In this paper, we consider s-adjacent deletion correcting codes for $s \geq 3$.

The paper is organized as follows. In Section II, we review two classes of codes that form the main ingredients in the constructions of s-adjacent deletion correcting codes presented in Section III. The paper ends with a conclusion in Section IV.

II. Two Useful Codes

In this section, we briefly review two classes of codes that are used in the following section to construct s-adjacent deletion correcting codes. The first class of codes is the well-known Varshamov-Tenengol'ts codes that were shown by Levenshtein to be capable of correcting a single deletion or insertion. The second class of codes can also correct a single deletion or insertion of a bit if it is known to the receiver that it is localized within two adjacent bits.

A. Varshamov-Tenengol'ts Codes

Recall that the Varshamov-Tenengol'ts code [7], denoted by $\mathrm{VT}_a(n)$, where $0 \le a \le n$, consists of all binary vectors $(x_1, x_2, \ldots, x_n)$ satisfying

$$\sum_{j=1}^{n} j x_j \equiv a \pmod{n+1}. \tag{2}$$

If a codeword $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ is transmitted and a single bit, say c_j, is deleted, then the receiver can retrieve the transmitted codeword. This is done by identifying the run of identical bits in the transmitted codeword that has been shortened (or deleted if the run has unit length). For example, (0011100001) is a codeword in $\mathrm{VT}_0(10)$. If it is received as (001100001), then the receiver can determine that the deleted bit belongs to the first run of 1's and by inserting a 1 in this run, it recovers the transmitted codeword.

The cardinality of $\mathrm{VT}_a(n)$ was determined by Ginzburg [4]. In particular, it is shown that the choice $a = 0$ maximizes the cardinality of the codes, i.e., $|\mathrm{VT}_0(n)| \ge |\mathrm{VT}_a(n)|$ for all $0 \le a \le n$. From [4], we have

$$|\mathrm{VT}_0(n)| = \frac{1}{2(n+1)} \sum_{\substack{d \mid n+1 \\ d \text{ odd}}} \phi(d) 2^{(n+1)/d} \ge \left\lceil \frac{2^n}{n+1} \right\rceil, \tag{3}$$

where ϕ is Euler's function. On the other hand, Lemma 1 in [2] implies that

$$|\mathrm{VT}_0(n)| < \frac{2^{n+1}}{n+1}. \tag{4}$$

B. Substitution-Transposition Codes

In this paper, we will make use of a code proposed in [1] in the context of detecting substitutions and transpositions of bits. For a fixed $a \in \{0, 1, 2\}$, the code of length n, which will be denoted here by $\mathrm{ST}_a(n)$ (rather than $C_n(t)$ as in equation (5) in [1]), is given by

$$\mathrm{ST}_a(n) = \{(c_1, c_2, \ldots, c_n) : \sum_{j=1}^{n} b_j c_j \equiv a \pmod{3}\}, \tag{5}$$

where

$$b_j = \begin{cases} 1 & \text{if } j \text{ is odd,} \\ 2 & \text{if } j \text{ is even.} \end{cases} \tag{6}$$

It is shown in [1] that the choice $a = 0$ maximizes the cardinality of the codes, i.e., $|\mathrm{ST}_0(n)| \ge |\mathrm{ST}_a(n)|$ for all $a \in \{0, 1, 2\}$ and that

$$\mathrm{ST}_0(n) = \left\lceil \frac{2^n}{3} \right\rceil = \begin{cases} \frac{2^n+1}{3} & \text{if } n \text{ is odd,} \\ \frac{2^n+2}{3} & \text{if } n \text{ is even.} \end{cases} \tag{7}$$

The code $\mathrm{ST}_a(n)$ is shown in [1] to be able to detect a a substitution of a bit, i.e., changing 0 to 1 or vice versa, or a transposition of two adjacent and different bits, i.e., changing 01 to 10 or vice versa. However, in the present work we are interested in deletions rather than substitutions and transpositions. We show next that the code can determine a deleted bit from a codeword $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ if it is known to the receiver that the deleted bit is either c_{j_0} or c_{j_0-1} (provided that $j_0 \ge 2$). Indeed, if this is not the case, then there are two distinct codewords, $\mathbf{c}' = (c'_1, c'_2, \ldots, c'_n)$ and $\mathbf{c}'' = (c''_1, c''_2, \ldots, c''_n)$, in $\mathrm{ST}_a(n)$ such that deleting c'_{j_0} from $\mathbf{c}'$ results in the same sequence as deleting c''_{j_0-1} from $\mathbf{c}''$. This implies that $c'_j = c''_j$ for $j < j_0 - 1$ or $j > j_0$ and $c'_{j_0-1} = c''_{j_0}$. Hence,

$$\begin{aligned} \sum_{j=1}^{n} b_j c'_j - \sum_{j=1}^{n} b_j c''_j &= \sum_{j=1}^{n} b_j (c'_j - c''_j) && (8) \\ &= b_{j_0-1}(c'_{j_0-1} - c''_{j_0-1}) + b_{j_0}(c'_{j_0} - c''_{j_0}) && (9) \\ &= b_{j_0-1}(c''_{j_0} - c''_{j_0-1}) + b_{j_0}(c'_{j_0} - c''_{j_0}) && (10) \\ &\equiv b_{j_0}(c'_{j_0} + c''_{j_0-1} - 2c''_{j_0}) \pmod 3 && (11) \end{aligned}$$

where we used the facts that $c'_j = c''_j$ for $j < j_0 - 1$ or $j > j_0$ in (9), $c'_{j_0-1} = c''_{j_0}$ in (10), and $b_{j-1} \equiv -b_j \pmod 3$, as given in (6), in (11). Since $\mathbf{c}'$ and $\mathbf{c}''$ are codewords in $\mathrm{ST}_a(n)$ satisfying (5), it follows that the left hand side in (8) vanishes modulo 3. As b_{j_0} being either 1 or 2, and hence nonzero modulo 3, (11) implies that

$$2c''_{j_0} \equiv c'_{j_0} + c''_{j_0-1} \pmod 3.$$

Given that $c'_{j_0}, c''_{j_0-1}, c''_{j_0} \in \{0, 1\}$, the above congruency yields $c'_{j_0} = c''_{j_0-1} = c''_{j_0}$. Hence, $c'_j = c''_j$ for all $1 \le j \le n$ contradicting the assumption that $\mathbf{c}'$ and $\mathbf{c}''$ are distinct. We conclude that $\mathrm{ST}_a(n)$ indeed can be used to recover a deleted bit from a codeword if it is known that the deleted bit is confined to one of two known adjacent positions. In particular, suppose a codeword in $\mathrm{ST}_a(n)$ is transmitted and it is known that the transmitted bit of index j_0 or $j_0 - 1$ (provided that $j_0 \ge 2$) is deleted resulting in the received word $\mathbf{r} = (r_1, r_2, \ldots, r_{n-1})$. Then the transmitted codeword is $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, where

$$c_j = \begin{cases} r_j & \text{for } j < j_0 - 1, \\ r_{j-1} & \text{for } j > j_0, \end{cases} \tag{12}$$

and either c_{j_0-1} is the solution of

$$c_{j_0-1} \equiv b_{j_0-1}^{-1} \left(a - \sum_{j=1}^{j_0-2} b_j r_j + \sum_{j=j_0-1}^{n-1} b_j r_j \right) \pmod 3 \tag{13}$$

if such a congruency has a solution $c_{j_0-1} \in \{0, 1\}$, in which case $c_{j_0} = r_{j_0-1}$, or c_{j_0} is the solution of

$$c_{j_0} \equiv b_{j_0}^{-1} \left(a - \sum_{j=1}^{j_0-1} b_j r_j + \sum_{j=j_0}^{n-1} b_j r_j \right) \pmod 3 \tag{14}$$

if such a congruency has a solution $c_{j_0} \in \{0, 1\}$, in which case $c_{j_0-1} = r_{j_0-1}$. Our argument that $\mathrm{ST}_a(n)$ can recover the deleted bit if it is confined to one of two known adjacent positions implies that exactly one of the two possibilities determined by (13) and (14) holds. Notice that if $j_0 = 1$, then the deleted bit is c_1 which can be determined from (14).

III. Constructions

In this section we construct s-adjacent deletion correcting codes of length N, a multiple of s. We give the basic framework of the construction methods before describing them.

A. Basics

Let $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ be a codeword in a code C of length $N = sn$. The bit c_ℓ in $\mathbf{c}$ can be indexed by a pair of positive integers as $c_{i,j}$ by mapping its index ℓ to (i, j) where

$$i = \ell - (\lceil \ell/s \rceil - 1)s \text{ and } j = \lceil \ell/s \rceil. \tag{15}$$

Clearly, $1 \leq i \leq s$ and $1 \leq j \leq n$. Then, the codeword $\mathbf{c}$ can be represented as an $s \times n$ matrix

$$\mathbf{c} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{s,1} & c_{s,2} & \cdots & c_{s,n} \end{bmatrix}, \tag{16}$$

the bits of which are transmitted over the channel column by column starting with the first column. Suppose that the channel caused the deletion of s adjacent bits starting at index ℓ^*. Then, the received word is $\mathbf{r} = (r_1, r_2, \ldots, r_{N-s})$ satisfying (1). By applying the mapping of indices specified in (15), the bit r_ℓ in the received sequence can be identified with $r_{i,j}$, where $1 \leq i \leq s$ and $1 \leq j < n$. With this indexing, the received word can be represented as an $s \times (n-1)$ matrix

$$\mathbf{r} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n-1} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{s,1} & r_{s,2} & \cdots & r_{s,n-1} \end{bmatrix}, \tag{17}$$

the bits of which are received column by column starting with the first column. Let (i^*, j^*) be the mapping of the index ℓ^* as specified in (15). Then, from (1), we have

$$r_{i,j} = \begin{cases} c_{i,j} & \text{for } j < j^* \text{ or } j = j^* \text{ and } i < i^*, \\ c_{i,j+1} & \text{for } j > j^* \text{ or } j = j^* \text{ and } i \geq i^*. \end{cases} \tag{18}$$

We notice that each row in the matrix form of $\mathbf{r}$ as given in (17) is obtained from the corresponding row in the matrix form of $\mathbf{c}$ as given in (16) by a single deletion of a bit. This is the foundation of all the constructions presented next.

B. Construction 1

If we choose each row in $\mathbf{c}$ to be a codeword in a Varshamov-Tenengol'ts code of length n, then each row in $\mathbf{c}$ can be recovered from the corresponding row in $\mathbf{r}$. Notice that the code constructed in this way of length $N = sn$, which we denote by $C_1(s, n)$, is obtained by interleaving s single deletion correcting codes. The construction of s-adjacent deletion correcting codes using s Varshamov-Tenengol'ts codes is a straightforward application of the interleaving concept. To maximize the size of the code, we choose each row in $\mathbf{c}$ to be in $\mathrm{VT}_0(n)$ since the choice $a = 0$ maximizes the size of $\mathrm{VT}_a(n)$. The number of codewords in $C_1(s, n)$ is then equal to $|\mathrm{VT}_0(n)|^s$. In Table I, we list the values of $|\mathrm{VT}_0(n)|$ for $2 \leq n \leq 12$. Using (3) and (4), we have

$$\left(\frac{2^n}{n+1}\right)^s \leq |C_1(s, n)| < \left(\frac{2^{n+1}}{n+1}\right)^s. \tag{19}$$

Example 1: Suppose we want to construct a code, $C_1(4, 6)$, of length $N = 4 \times 6$ which is 4-adjacent deletion correcting. Each codeword in the code can be represented as a 4×6 matrix as in (16), where each row in the matrix is a codeword in $\mathrm{VT}_0(6)$, There are exactly ten codewords in $\mathrm{VT}_0(6)$. Hence, the total number of codewords in $C_1(4, 6)$ is $10^4 = 10,000$.

Notice that the construction of $C_1(s, n)$ does not take into account that the deleted bits from $\mathbf{c}$ in (16) are consecutive in the sense that if the deletion starts at index ℓ^*, then, from (18), the s bits c_{i,j^*} with $i^* \leq i \leq s$ and c_{i,j^*+1} with $1 \leq i \leq i^* - 1$ are deleted. In particular, if one can deduce that the bit $c_{1,j}$ is deleted from the first row in (16), then only $c_{i,j}$ or $c_{i,j-1}$ (if $j \geq 2$) is deleted from the i-th row, $2 \leq i \leq s$. This information can be used to obtain more efficient constructions as shown next.

C. Construction 2

We present a code, denoted by $C_2(s, n)$ of length $N = sn$, the codewords of which are represented by $s \times n$ matrices as in (16). Let $a_2, \ldots, a_s \in \{0, 1, 2\}$. We choose the first row in the codewords of $C_2(s, n)$ to be the vector $(0, 1, 0, 1, \ldots)$ of length n composed of alternating 0's and 1's. The i-th row, for $2 \leq i \leq s$, in $C_2(s, n)$, is selected from $\mathrm{ST}_{a_i}(n)$ in (5). Hence, $C_2(s, n)$ has exactly $\prod_{i=2}^{s} |\mathrm{ST}_{a_i}(n)|$ codewords. To maximize the size of $C_2(s, n)$, we choose $a_2 = a_3 = \cdots = a_s = 0$. With this choice, it follows from (7) that

$$|C_2(s, n)| = \left\lceil \frac{2^n}{3} \right\rceil^{s-1}. \tag{20}$$

To show that $C_2(s, n)$ is an s-adjacent deletion correcting code, note that if s adjacent bits are deleted from a codeword $\mathbf{c}$, then a bit is deleted from each row in its matrix representation in (16) to give the received word $\mathbf{r}$ as given in its matrix representation in (17). Since the first row in $\mathbf{c}$ is $(0, 1, 0, 1, \ldots)$, then deleting a bit other than the first or the last from it results in a vector with exactly one pair of two identical and consecutive bits in the first row $(r_{1,1}, r_{1,2}, \ldots, r_{1,n-1})$ of $\mathbf{r}$. On the other hand, deleting the first or last bit from the first row in $\mathbf{c}$ results in the vectors $(1, 0, 1, \ldots)$ and $(0, 1, 0, \ldots)$, respectively, of length $n - 1$ of alternating 0's and 1's. From this, the receiver can determine the index of the bit in the first row of $\mathbf{c}$ that has been deleted. Suppose that j_0 is this index. Then, either c_{i,j_0} or c_{i,j_0-1} (if $j_0 \geq 2$) is deleted from the i-th row, $2 \leq i \leq s$, in $\mathbf{c}$. Since this is a codeword in $\mathrm{ST}_a(n)$, it can be recovered using (12)–(14).

Example 2: We construct a code, $C_2(4, 6)$, of length $N = 4 \times 6 = 24$ which is 4-adjacent deletion correcting. The first row of the codewords in $C_2(4, 6)$ is $(0, 1, 0, 1, 0, 1)$ and the remaining rows are codewords in $\mathrm{ST}_0(6)$. The code $C_2(4, 6)$ has $22^3 = 10,648$ codewords as $\mathrm{ST}_0(6)$ is of size 22 according to (7). As an example,

$$\mathbf{c} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

is a codeword in $C_2(4, 6)$, which is the matrix representation of the sequence

$$(0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1).$$

Suppose that the received sequence is

$$(0,1,0,1,1,0,0,0,0,1,1,1,0,1,0,0,1,0,1,1),$$

which can be placed in a matrix form as

$$\mathbf{r} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Since the receiver knows that the first row in $\mathbf{c}$ is $(0,1,0,1,0,1)$, it concludes from the first row of $\mathbf{r}$ that the fourth bit has been deleted from the first row of $\mathbf{c}$. Therefore, either the third or fourth bits have been deleted from each of the following rows in $\mathbf{c}$. Using (12)–(14), these three rows can be retrieved.

D. Construction 3

This construction gives a code, denoted by $C_3(s,n)$, of length $N = sn$, the codewords of which are also represented by $s \times n$ matrices as in (16). Let $a_1, a_2, a_3, \ldots, a_s$ be such that $0 \leq a_1, a_2 \leq n$ and $0 \leq a_3, \ldots, a_s \leq 2$. A codeword $\mathbf{c}$ in $C_3(s,n)$ is specified as follows. The first and second rows, $(c_{1,1}, c_{1,2}, \ldots, c_{1,n})$ and $(c_{2,1}, c_{2,2}, \ldots, c_{2,n})$, are selected to be in the Varshamov-Tenengol'ts codes $\mathrm{VT}_{a_1}(n)$ and $\mathrm{VT}_{a_2}(n)$, respectively, but we demand that $(c_{1,j}, c_{2,j}) \neq (c_{1,j-1}, c_{2,j-1})$ for all $2 \leq j \leq n$. Let P_{a_1,a_2} be the set of pairs of codewords in the Varshamov-Tenengol'ts codes $\mathrm{VT}_{a_1}(n)$ and $\mathrm{VT}_{a_2}(n)$ satisfying this property. The i-th row, for $3 \leq i \leq s$, is selected from $\mathrm{ST}_{a_i}(n)$.

To show that $C_3(s,n)$ is an s-adjacent deletion correcting code, note that if s adjacent bits are deleted from a codeword $\mathbf{c}$, then a bit is deleted from each row in its matrix representation in (16) to give the received word $\mathbf{r}$ as given in its matrix representation in (17). Since the first row in $\mathbf{c}$ is a codeword in the Varshamov-Tenengol'ts code $\mathrm{VT}_{a_1}(n)$, then the receiver can determine the run in $(c_{1,1}, c_{1,2}, \ldots, c_{1,n})$ that contains the deleted bit. Similarly, from the second row in $\mathbf{c}$, being a codeword in $\mathrm{VT}_{a_2}(n)$, the receiver can determine the run in $(c_{2,1}, c_{2,2}, \ldots, c_{2,n})$ that contains the deleted bit. Recall that $(c_{1,j}, c_{2,j}) \neq (c_{1,j-1}, c_{2,j-1})$ for all $2 \leq j \leq n$, and if j_1 and j_2 are the indices of the deleted bits in the first and second rows of $\mathbf{c}$, respectively, then $j_2 = j_1$ or $j_2 = j_1 - 1$. Based on this, the receiver can determine at most two possibilities for the indices of the deleted bit in the i-th row of $\mathbf{c}$, $3 \leq i \leq s$. The receiver can then recover this row as it belongs to $\mathrm{ST}_{a_i}(n)$. Hence, $C_3(s,n)$ is an s-adjacent deletion correcting code.

Example 3: Again, we construct a code, $C_3(4,6)$, of length $N = 4 \times 6 = 24$ which is 4-adjacent deletion correcting. The first and second rows of the codewords in $C_3(4,6)$ are selected to be codewords in $\mathrm{VT}_0(6)$ and $\mathrm{VT}_3(6)$, respectively. From (3), there are exactly 10 codewords in $\mathrm{VT}_0(6)$, which are listed below:

$$\begin{aligned} &\mathbf{c}_0^1 = (0,0,0,0,0,0), && \mathbf{c}_0^6 = (0,1,1,1,1,0), \\ &\mathbf{c}_0^2 = (1,0,0,0,0,1), && \mathbf{c}_0^7 = (1,0,1,1,0,1), \\ &\mathbf{c}_0^3 = (0,1,0,0,1,0), && \mathbf{c}_0^8 = (1,1,1,1,1,1), \\ &\mathbf{c}_0^4 = (0,0,1,1,0,0), && \mathbf{c}_0^9 = (1,1,0,1,0,0), \\ &\mathbf{c}_0^5 = (1,1,0,0,1,1), && \mathbf{c}_0^{10} = (0,0,1,0,1,1). \end{aligned}$$

and nine codewords in $\mathrm{VT}_3(6)$, which are listed below:

$$\begin{aligned} &\mathbf{c}_3^1 = (1,1,0,0,0,0), && \mathbf{c}_3^6 = (0,1,1,0,1,0), \\ &\mathbf{c}_3^2 = (0,0,1,0,0,0), && \mathbf{c}_3^7 = (0,1,0,1,1,1), \\ &\mathbf{c}_3^3 = (0,0,0,1,0,1), && \mathbf{c}_3^8 = (1,1,1,0,1,1), \\ &\mathbf{c}_3^4 = (1,0,1,0,0,1), && \mathbf{c}_3^9 = (1,1,1,1,0,0), \\ &\mathbf{c}_3^5 = (1,0,0,1,1,0). \end{aligned}$$

If we choose $\mathbf{c}_0^4 = (0,0,1,1,0,0)$ as the first row of a codeword in $C_3(4,6)$, then the second row should be $(c_1, c_2, c_3, c_4, c_5, c_6)$ where $c_1 \neq c_2$, $c_3 \neq c_4$, and $c_5 \neq c_6$, since the first and second bits, the third and fourth bits, and the fifth and sixth bits are equal in $\mathbf{c}_0^4$. It follows that the second row is $\mathbf{c}_3^4$, $\mathbf{c}_3^5$, or $\mathbf{c}_3^6$. By checking all pairs of codewords in $\mathrm{VT}_0(6)$ and $\mathrm{VT}_3(6)$ in this way, we obtain $P_{0,3}(6)$ which consists of the following 26 pairs for the first two rows in the codewords of $C_3(4,6)$:

$$\begin{aligned} &(\mathbf{c}_0^3, \mathbf{c}_3^2), && (\mathbf{c}_0^3, \mathbf{c}_3^3), && (\mathbf{c}_0^3, \mathbf{c}_3^4), && (\mathbf{c}_0^3, \mathbf{c}_3^5), \\ &(\mathbf{c}_0^3, \mathbf{c}_3^6), && (\mathbf{c}_0^3, \mathbf{c}_3^7), && (\mathbf{c}_0^3, \mathbf{c}_3^8), && (\mathbf{c}_0^4, \mathbf{c}_3^4), \\ &(\mathbf{c}_0^4, \mathbf{c}_3^5), && (\mathbf{c}_0^4, \mathbf{c}_3^6), && (\mathbf{c}_0^5, \mathbf{c}_3^4), && (\mathbf{c}_0^5, \mathbf{c}_3^5), \\ &(\mathbf{c}_0^5, \mathbf{c}_3^6), && (\mathbf{c}_0^7, \mathbf{c}_3^2), && (\mathbf{c}_0^7, \mathbf{c}_3^3), && (\mathbf{c}_0^7, \mathbf{c}_3^4), \\ &(\mathbf{c}_0^7, \mathbf{c}_3^5), && (\mathbf{c}_0^7, \mathbf{c}_3^6), && (\mathbf{c}_0^7, \mathbf{c}_3^7), && (\mathbf{c}_0^7, \mathbf{c}_3^8), \\ &(\mathbf{c}_0^9, \mathbf{c}_3^4), && (\mathbf{c}_0^9, \mathbf{c}_3^5), && (\mathbf{c}_0^9, \mathbf{c}_3^6), && (\mathbf{c}_0^{10}, \mathbf{c}_3^4), \\ &(\mathbf{c}_0^{10}, \mathbf{c}_3^5), && (\mathbf{c}_0^{10}, \mathbf{c}_3^6). \end{aligned}$$

For the third and fourth rows, we can choose any codewords in $\mathrm{ST}_0(6)$. From (7), there are exactly twenty two codewords in $\mathrm{ST}_0(6)$. Hence, the code $C_3(4,6)$ is composed of $26 \times 22 \times 22 = 12,584$ codewords.

As an example of a codeword in $C_3(4,6)$, we can take the first row to be $\mathbf{c}_0^3$, the second row to be $\mathbf{c}_3^2$, and the third and fourth rows to be $(0,0,1,0,0,1)$ and $(1,0,1,0,1,0)$, respectively, which are codewords in $\mathrm{ST}_0(6)$. Hence, the codeword in a matrix form is

$$\mathbf{c} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

and as a sequence is

$$(0,0,0,1,1,0,0,0,0,1,1,1,0,0,0,0,1,0,0,1,0,0,1,0).$$

Suppose that the received sequence is

$$(0,0,0,1,1,0,0,0,0,1,1,0,1,0,0,1,0,0,1,0),$$

which can be placed in a matrix form as

$$\mathbf{r} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

As the first row in $\mathbf{r}$ is obtained by a single deletion from a codeword in $\mathrm{VT}_0(6)$, we can retrieve the first row in $\mathbf{c}$ to be $(0,1,0,0,1,0)$. The deleted bit from this row is the third or fourth bit. The second row in $\mathbf{r}$ is obtained by a single deletion from a codeword in $\mathrm{VT}_3(6)$. Hence, we can retrieve the second

TABLE I
SIZES OF $\mathrm{VT}_0(n)$ AND $P_{a_1,a_2}(n)$ MAXIMIZED OVER ALL $0 \le a_1, a_2 \le n$, AND A CHOICE OF (a_1, a_2) ACHIEVING THIS MAXIMUM.

n	$\lvert\mathrm{VT}_0(n)\rvert$	$\max \lvert P_{a_1,a_2}(n)\rvert$	(a_1, a_2)
2	2	2	$(0,1)$
3	2	4	$(1,3)$
4	4	6	$(0,2)$
5	6	13	$(0,0)$
6	10	26	$(0,3)$
7	16	60	$(2,6)$
8	30	136	$(0,4)$
9	52	305	$(0,0)$
10	94	726	$(0,5)$
11	172	1868	$(3,9)$
12	316	4468	$(0,6)$

row in $\mathbf{c}$ to be $(0,0,1,0,0,0)$. The deleted bit from this row is either the fourth, fifth, or sixth bit. Since if the j-th bit is deleted from the first row, then either the j-th or the $j-1$-st bit is deleted from the second row, it follows that the deleted bit in the first row is the fourth bit and the deleted bit from the second row is also the fourth bit. The deleted bit from the third row is, therefore, either the third or fourth bit. Using (12)–(14), we can retrieve the third row in $\mathbf{c}$ from the third row in $\mathbf{r}$ as $(0,0,1,0,0,1)$. Similarly, the deleted bit from the fourth row is either the third or fourth bit and using (12)–(14), we can retrieve the fourth row in $\mathbf{c}$ from the fourth row in $\mathbf{r}$ as $(1,0,1,0,1,0)$.

We notice that in this example, we have chosen the first row in the codewords of $C_3(4,6)$ to be in $\mathrm{VT}_0(6)$ and the second row to be in $\mathrm{VT}_3(6)$. Although $\mathrm{VT}_0(6)$ is of size 10 while $\mathrm{VT}_3(6)$ is of size 9, choosing both rows from $\mathrm{VT}_0(6)$ yields only 16 pairs of valid pairs in $P_{0,0}(6)$ instead of the 26 pairs in $P_{0,3}(6)$ we used in the example.

Table I gives the cardinality of $\mathrm{VT}_0(n)$, which is the maximum cardinality of $\mathrm{VT}_a(n)$ over all $0 \le a \le n$, and the cardinality of P_{a_1,a_2} maximized over all choices of a_1 and a_2, where $0 \le a_1, a_2 \le n$. The table also gives a pair (a_1, a_2) achieving this maximum. From this table, we can determine the largest size of an s-adjacent deletion correcting code, $C_3(s,n)$, $2 \le n \le 12$, which is given by

$$|C_3(s,n)| = \max |P_{a_1,a_2}(n)| \left\lceil \frac{2^n}{3} \right\rceil^{s-2}. \tag{21}$$

We can lower bound the maximum cardinality of P_{a_1,a_2} as follows. Notice that there are in total $4 \times 3^{n-1}$ pairs of sequences $(c_{1,1}, c_{1,2}, \ldots, c_{1,n})$ and $(c_{2,1}, c_{2,2}, \ldots, c_{2,n})$ such that $(c_{1,j}, c_{2,j}) \neq (c_{1,j-1}, c_{2,j-1})$ for all $2 \le j \le n$. Indeed, there are four choices for the pair $(c_{1,1}, c_{2,1})$. For each choice, there are three choices for the pair $(c_{1,2}, c_{2,2}) \neq (c_{1,1}, c_{2,1})$. In general, for $2 \le j \le n$, there are three choices for $(c_{1,j}, c_{2,j})$ that are different from $(c_{1,j-1}, c_{2,j-1})$. This gives $4 \times 3^{n-1}$ as the number of pairs of sequences $(c_{1,1}, c_{1,2}, \ldots, c_{1,n})$ and $(c_{2,1}, c_{2,2}, \ldots, c_{2,n})$ such that $(c_{1,j}, c_{2,j}) \neq (c_{1,j-1}, c_{2,j-1})$ for all $2 \le j \le n$. As each sequence of length n belongs to $\mathrm{VT}_a(n)$ for a unique value of a, $0 \le a \le n$, we have

$$\sum_{a_1=0}^{n} \sum_{a_2=0}^{n} |P_{a_1,a_2}(n)| = 4 \times 3^{n-1},$$

which gives an average value of $4 \times 3^{n-1}/(n+1)^2$ for $|P_{a_1,a_2}(n)|$. Since the maximum of this cardinality is at least equal to this average, we get

$$\max_{0 \le a_1,a_2 \le n} |P_{a_1,a_2}(n)| \ge \frac{4 \times 3^{n-1}}{(n+1)^2}. \tag{22}$$

IV. CONCLUSION

In this paper, we presented three construction methods for codes capable of correcting three or more adjacent deletions. The first is a straightforward application of the interleaving principle using Varshamov-Tenengol'ts codes. The second construction, based on substitution-transposition codes, is the simplest to implement and decode. The third construction uses a combination of both codes.

As a comparison between the three constructions, we consider the code rate defined by $\frac{\log_2 |C(s,n)|}{sn}$ for an s-adjacent deletion correcting code $C(s,n)$ of length sn where $s = \alpha n$ for some constant $\alpha > 0$. Levenshtein's upper bound [6] implies that the code rate is at most $1 - \frac{1}{n} + O\left(\frac{\log_2 n}{n^2}\right)$. From (19) and (20), the rates of codes based on the first and second constructions are $1 - \frac{\log_2 n}{n} + O\left(\frac{1}{n}\right)$ and $1 - \frac{1+\alpha \log_2 3}{\alpha n} + O\left(\frac{1}{n^2}\right)$, respectively. From (21) and (22), the rates of codes based on the third construction are at least $1 - \frac{2 - \log_2 3 + \alpha \log_2 3}{\alpha n} + O\left(\frac{\log_2 n}{n^2}\right)$.

We should mention that in this paper we assumed that *exactly* s adjacent deletions occur. This is a crucial assumption since it allows the receiver to write the received sequence as a matrix in which each row suffers from a single deletion. All constructions rely on this assumption. Definitely, it is interesting to construct codes that can correct *at most* s adjacent deletions. In [3], Bours constructed codes that can do that and also correct substitution errors using a product code construction. However, their rates are much lower than the codes presented here.

REFERENCES

[1] K. A. S. Abdel-Ghaffar, "Detecting substitutions and transpositions of characters," *Comput. J.*, vol. 41, no. 4, pp. 270–277, 1998.

[2] K. A. S. Abdel-Ghaffar and H. C. Ferreira, "Systematic encoding of the Varshamov-Tenengol'ts codes and the Constantin-Rao codes" *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 340–345, Jan. 1998.

[3] P. A. H. Bours, "Codes for correcting insertion and deletion errors," Ph.D. dissertation, Eindhoven University of Technology, June 1994 [online] Available at http://alexandria.tue.nl/extra3/proefschrift/PRF10A/9421971.pdf.

[4] B. D. Ginzburg, "A number-theoretic function with an application in the theory of coding," *Problemy Kibernetiki*, (in Russian), vol. 19, pp. 249–252, 1967. English translation in *Systems Theory Research*, vol. 19, pp. 255–259, 1970.

[5] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Doklady Akademii Nauk SSSR*, (in Russian) vol. 163, pp. 845–848, 1965. English translation in *Soviet Physics-Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[6] V. I. Levenshtein,"Asymptotically optimum binary code with correction for losses of one or two adjacent bits," *Problemy Kibernetiki*, (in Russian), vol. 19, pp. 293–298, 1967. English translation in *Systems Theory Research*, vol. 19, pp. 298–304, 1970.

[7] R. P. Varshamov and G. M. Tenengol'ts, "Codes which correct single asymmetric errors." *Automat. Telemekh.*, (in Russian), vol. 26, no. 2, pp. 286–290, 1965. English translation in *Automat. Rem. Contr.*, vol. 26, no. 2, pp. 286–290, 1965.