# A Note on Non-Binary Multiple Insertion/Deletion Correcting Codes

Filip Palunčić*, Theo G. Swart*, Jos H. Weber†, Hendrik C. Ferreira* and Willem A. Clarke*
*Department of Electrical and Electronic Engineering Science,
University of Johannesburg, Auckland Park, 2006, South Africa
Emails: {fpaluncic, tgswart, hcferreira, willemc}@uj.ac.za
†Delft University of Technology, IRCTR/CWPC
Mekelweg 4, 2628 CD Delft, The Netherlands
Email:j.h.weber@tudelft.nl

***Abstract*— We propose the construction of a non-binary multiple insertion/deletion correcting code based on a binary multiple insertion/deletion correcting code. In essence, it is a generalisation of Tenengol'ts' non-binary single insertion/deletion correcting code. We evaluate the cardinality of the proposed construction based on the asymptotic upper bound on the cardinality of a maximal binary multiple insertion/deletion correcting code derived by Levenshtein.**

## I. Introduction

The Levenshtein code construction is remarkable for a number of reasons. This single insertion/deletion correcting code is based on a construction originally developed by Varshamov and Tenengol'ts [1] to correct a single asymmetric error. Subsequently, Levenshtein [2] realised that this construction is also a single insertion/deletion correcting code. Levenshtein [2] showed that his code is asymptotically optimal and, later, Sloane [3] showed that the Levenshtein code was optimal for codeword length less than or equal to 9. Furthermore, Sloane [3] conjectured that the Levenshtein code is optimal for all codeword lengths.

The Levenshtein code consists of all binary vectors $\mathbf{x}_n = (x_1 x_2 \dots x_n)$ that satisfy $\sum_{i=1}^{n} i x_i \equiv a (\text{mod } m)$, for particular pairs of $a$ and $m$, where $m \geq n+1$.

Building on this important result by Levenshtein, Tenengol'ts [4] showed how the Levenshtein code can be utilised in the construction of a non-binary single insertion/deletion correcting code. Furthermore, Tenengol'ts [4] derived an asymptotic upper bound on the cardinality of an optimal non-binary code capable of correcting a single insertion/deletion error, which showed that his construction is close to asymptotic optimality.

However, progress with respect to binary multiple insertion/deletion correcting codes has been limited and slow. An important generalisation of the Levenshtein code is the Helberg code [5][1]. However, the Helberg code suffers from a low cardinality.

To construct a non-binary multiple insertion/deletion correcting code, we will use a generalisation of the approach used by Tenengol'ts [4]. Our construction is based on any binary multiple insertion/deletion correcting code, such as the Helberg code. Table I places our construction in context in comparison to other insertion/deletion correcting codes.

TABLE I
Overview of insertion/deletion correcting codes

| | Binary | Non-binary |
|---|---|---|
| Single Ins./Del. Correction | Levenshtein [2] | Tenengol'ts [4] |
| Multiple Ins./Del. Correction | Helberg [5] | Proposed Construction Method (This Paper) |

The paper is organised as follows. In Section II, we give a description of Tenengol'ts' non-binary single insertion/deletion correcting code. In Section III, we give the generalisation of Tenengol'ts' construction for a non-binary multiple insertion/deletion correcting code. Since we believe that the Helberg code has a low cardinality, we will evaluate in Section IV the cardinality of the proposed code using a conjectured cardinality of a binary multiple insertion/deletion correcting code based on the asymptotic upper bound on the cardinality of such codes [2]. Finally, we conclude the paper in Section V.

## II. Tenengol'ts Code

Let $\mathcal{A}_q = \{0, 1, \dots, q-1\}$ be an alphabet of size $q$. Let $\mathbf{x}_n = (x_1 x_2 \dots x_n)$ be a $q$-ary vector of length $n$, i.e. $x_i \in \mathcal{A}_q$ for $1 \leq i \leq n$. For the Tenengol'ts construction, the following mapping is important. Let $f(\mathbf{x}_n) = (f(x_1) f(x_2) \dots f(x_n))$, where $f : \mathcal{A}_q^n \to \mathcal{A}_2^n$, $f(x_1)$ can be 0 or 1 and

$$f(x_i) = \begin{cases} 1, & \text{if } x_i \geq x_{i-1} \\ 0, & \text{if } x_i < x_{i-1} \end{cases} \tag{1}$$

for $2 \leq i \leq n$. We will refer to the binary sequence $f(\mathbf{x}_n)$ as a *relational sequence*. Then the Tenengol'ts code consists of all $q$-ary vectors $\mathbf{x}_n = (x_1 x_2 \dots x_n)$ that satisfy

$$\sum_{i=1}^{n} x_i \equiv \beta (\text{mod } q) \tag{2}$$

$$\sum_{i=1}^{n} (i-1) f(x_i) \equiv \gamma (\text{mod } n), \tag{3}$$

[1]Note that it has been proven that the Helberg code is indeed a multiple insertion/deletion correcting code in [6].

for fixed values of $\beta$ and $\gamma$. The purpose of (2) is to determine the value of the symbol that was deleted or inserted. The constraint in (3) essentially corresponds to the Levenshtein code. To see why this is so, consider the mapping $g(\mathbf{x}_n) = (g(x_1)g(x_2)\dots g(x_{n-1}))$, where $g : \mathcal{A}_q^n \to \mathcal{A}_2^{n-1}$ and

$$g(x_i) = \begin{cases} 1, & \text{if } x_{i+1} \geq x_i \\ 0, & \text{if } x_{i+1} < x_i. \end{cases} \tag{4}$$

Therefore, $g(\mathbf{x}_n)$ corresponds to the last $n-1$ elements of $f(\mathbf{x}_n)$. Let $\mathcal{L}(n)$ denote a Levenshtein codebook consisting of codewords of length $n$ for $m = n+1$. Then, the constraint in (3) can be stated equivalently as $g(\mathbf{x}_n) \in \mathcal{L}(n-1)$.

To understand how the Tenengol'ts code works, consider the case where a single deletion occurs. The first thing to note is that a single deletion in $\mathbf{x}_n$ results in a single deletion in $f(\mathbf{x}_n)$. To see why, consider the sequence

$$\dots x_{j-1} \overset{f(x_j)}{\widehat{\phantom{x}}} x_j \overset{f(x_{j+1})}{\widehat{\phantom{x}}} x_{j+1} \dots$$

Assume that the symbol $x_j$ is deleted. If $f(x_j) = f(x_{j+1})$, then $x_{j-1}$ and $x_{j+1}$ have the same relation to each other as $f(x_j)$ (i.e. if $f(x_j) = f(x_{j+1}) = 1$, then $x_{j-1} \leq x_j$ and $x_j \leq x_{j+1}$, and so $x_{j-1} \leq x_{j+1}$). Therefore, in the relational sequence $f(\mathbf{x}_n)$, a single bit is deleted. If $f(x_j) \neq f(x_{j+1})$, then the relation between $x_{j-1}$ and $x_{j+1}$ can correspond to either $f(x_j)$ or $f(x_{j+1})$. Again, in the relational sequence, there is a single bit that is deleted.

Let $\mathbf{x}'_{n-1}$ correspond to $\mathbf{x}_n$ after a single deletion. Since $f(\mathbf{x}'_{n-1})$ corresponds to $f(\mathbf{x}_n)$ after a single deletion, using (3) we can reconstruct $f(\mathbf{x}_n)$. Using (2), we can determine the value of the deleted symbol. Then, all that we have to determine is the exact position of the deleted $q$-ary symbol based on the recovered relational sequence. The position of the deleted $q$-ary symbol corresponds to the run where the bit was deleted from the relational sequence $f(\mathbf{x}_n)$ or to the bit preceding that run (a run refers to a maximal length substring consisting of consecutive like symbols). Since a run in the relational sequence corresponds to a sequence of monotonically increasing (decreasing) symbols in $\mathbf{x}_n$, the precise location of the deleted symbol can be determined.

## III. Generalisation of the Tenengol'ts Code

In this section, we will show how Tenengol'ts' approach can be extended to multiple insertion/deletion correcting codes. Such a construction is based upon a binary multiple insertion/deletion correcting code. Let $s$ denote the maximal insertion/deletion correcting capability of a code. A codebook is referred to as an $s$ insertion/deletion correcting codebook if it is capable of correcting $s$ or fewer insertion/deletion errors. Let $\mathcal{C}_s(n)$ denote a binary $s$ insertion/deletion correcting codebook consisting of codewords of length $n$.

We wish to construct a $q$-ary code $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \dots, \beta_s))$, which is capable of correcting $s$ or fewer insertion/deletion errors. We will consider the case where $s$ or fewer deletion errors occur. Any code capable of correcting $s$ deletion errors is also an $s$ insertion/deletion correcting code [2].

Corresponding to the constraint in (3) for the Tenengol'ts code, we require that for $\mathbf{x}_n \in \mathcal{T}_s(q, n, (\beta_1, \beta_2, \dots, \beta_s))$, that $g(\mathbf{x}_n) \in \mathcal{C}_s(n-1)$. Therefore, in the relational sequence we are able to correct $s$ or fewer binary insertion/deletion errors.

We also need to generalise the constraint in (2). To achieve this, we will use the following set of $s$ congruencies:

$$\begin{aligned} \sum_{i=1}^{n} x_i &\equiv \beta_1 (\text{mod } p), \\ \sum_{i=1}^{n} x_i^2 &\equiv \beta_2 (\text{mod } p), \\ &\vdots \\ \sum_{i=1}^{n} x_i^s &\equiv \beta_s (\text{mod } p). \end{aligned} \tag{5}$$

These constraints are used to determine the values of the $s$ or fewer deleted symbols.

*Lemma 1:* The set of congruencies in (5) for a fixed value of $(\beta_1, \beta_2, \dots, \beta_s)$ can uniquely determine the values of $s$ or fewer randomly deleted symbols provided that $p$ is prime and $p > \max(q-1, s)$.

*Proof:* Consider some $\mathbf{x}_n = (x_1 x_2 \dots x_n)$ that corresponds to an arbitrary, but fixed $(\beta_1, \beta_2, \dots, \beta_s)$ in (5). Assume that the symbols $x_{i_1}, x_{i_2}, \dots, x_{i_s}$, where $1 \leq i_1 < i_2 < \dots < i_s \leq n$, are deleted from $\mathbf{x}_n$, resulting in $\mathbf{x}'_{n-s} = (x'_1 x'_2 \dots x'_{n-s})$. Let $q_1 = x_{i_1}, q_2 = x_{i_2}, \dots, q_s = x_{i_s}$. Furthermore, let

$$\sum_{i=1}^{n-s} {x'}_i^j \equiv \beta'_j (\text{mod } p),$$

for $1 \leq j \leq s$. Let $\beta''_j = \beta_j - \beta'_j$. Then

$$\beta''_j \equiv q_1^j + q_2^j + \dots + q_s^j (\text{mod p}). \tag{6}$$

Therefore, we have $s$ congruency equations in $s$ unknowns. It is known that the equations in (6) have a unique solution provided that $p$ is prime and $p > \max(q-1, s)$, which is precisely the set $\{q_1, q_2, \dots, q_s\}$ (see for example [7], [8]).

Suppose that $s' < s$ deletions occur. Then, to determine the values of the $s'$ symbols deleted, the first $s'$ congruency equations are sufficient. ∎

Now consider the set $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \dots, \beta_s))$, which is defined as

$$\begin{aligned} \hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \dots, \beta_s)) = \Big\{ & \mathbf{x}_n = (x_1 x_2 \dots x_n) \in \mathcal{A}_q^n : \\ & g(\mathbf{x}_n) \in \mathcal{C}_s(n-1), \\ & \sum_{i=1}^{n} x_i \equiv \beta_1 (\text{mod } p), \\ & \sum_{i=1}^{n} x_i^2 \equiv \beta_2 (\text{mod } p), \\ & \vdots \\ & \sum_{i=1}^{n} x_i^s \equiv \beta_s (\text{mod } p) \Big\}, \end{aligned} \tag{7}$$

where $p$ is prime and $p > \max(q-1, s)$. It turns out that $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ is not an $s$ insertion/deletion correcting code for $s \neq 1$. To understand why, consider the following example.

*Example 1:* Consider the case $s = 2$, $q = 6$ and $n = 7$. Therefore, $p = 7$ is the minimum possible value of $p$, as $p$ needs to be a prime number. Assume that the codeword $\mathbf{x}_7 = (3244015) \in \hat{\mathcal{T}}_2(6, 7, (5, 1))$ is transmitted, for which $f(\mathbf{x}_7) = (\times 011011)$ (the sign $\times$ is used to represent a "don't care" condition, either a 0 or a 1). Assume that the second and second last symbols are deleted, and so $\mathbf{x}'_5 = (34405)$ is received. Then, $f(\mathbf{x}'_5) = (\times 1101)$. At receiver, we can recover $f(\mathbf{x}_7)$ as $g(\mathbf{x}_7) \in \mathcal{C}_2(6)$. Furthermore, we know that the symbols 2 and 1 were deleted due to the constraints in (5). However, we do not know which symbol came first in the original codeword $\mathbf{x}_7$. This is where ambiguity originates from. Let us attempt to perform decoding symbol by symbol. Since, at the receiver, we know that $f(\mathbf{x}_7) = (\times 011011)$, we know that the second bit and a bit in the last run were deleted. The question is: which symbol is to be associated with which deleted bit in the relational sequence? In this particular case, there are two options:

1) The deleted symbol 1 is associated with the bit 0 deleted from the relational sequence and the deleted symbol 2 is associated with the bit 1 deleted from the relational sequence. Decoding symbol by symbol we obtain the following result. Firstly, the symbol 1: then $f(\mathbf{x}'_6) = (\times 01101)$. Therefore the symbol could only have been deleted from the first or second position of the original codeword. Then $\mathbf{x}'_6 = (314405)$. Secondly, the symbol 2: then $f(\mathbf{x}_7) = (\times 011011)$. Therefore the symbol could only have been deleted from fifth, sixth or seventh position of the original codeword. Then $\mathbf{x}_7 = (3144025)$.
2) The deleted symbol 2 is associated with the bit 0 deleted from the relational sequence and the deleted symbol 1 is associated with the bit 1 deleted from the relational sequence. Decoding symbol by symbol we obtain the following result. Firstly, the symbol 2: then $f(\mathbf{x}'_6) = (\times 01101)$. Therefore the symbol could only have been deleted from the first or second position of the original codeword. Then $\mathbf{x}'_6 = (324405)$. Secondly, the symbol 1: then $f(\mathbf{x}_7) = (\times 011011)$. Therefore the symbol could only have been deleted from fifth, sixth or seventh position of the original codeword. Then $\mathbf{x}_7 = (3244015)$.

Therefore, we obtain two possible codewords, both in $\hat{\mathcal{T}}_2(6, 7, (5, 1))$, and have no way of knowing which was the originally transmitted codeword.

For the general case of the set $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$, it turns out there are never more than $s!$ of such possible codewords. This is the purpose of the following lemma.

*Lemma 2:* If $s$ symbols are deleted from a codeword $\mathbf{x}_n \in \hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ for an arbitrary, but fixed $(\beta_1, \beta_2, \ldots, \beta_s)$ to obtain $\mathbf{x}'_{n-s}$, then from $\mathbf{x}'_{n-s}$ one can recover at most $s!$ codewords (including the original codeword) that are in $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$.

*Proof:* Assume that $s$ symbols, $\{q_1, q_2, \ldots, q_s\}$, are deleted from a codeword $\mathbf{x}_n \in \hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$, for an arbitrary, but fixed $(\beta_1, \beta_2, \ldots, \beta_s)$, resulting in $\mathbf{x}'_{n-s}$. Therefore, $f(\mathbf{x}'_{n-s})$ is obtained from $f(\mathbf{x}_n)$ through $s$ deletions. Since $g(\mathbf{x}_n) \in \mathcal{C}_s(n-1)$, $g(\mathbf{x}_n)$ can be recovered from $g(\mathbf{x}'_{n-s})$, which implies that $f(\mathbf{x}_n)$ can also be recovered. Furthermore, due to the constraints in (5), the set $\{q_1, q_2, \ldots, q_s\}$ can also be recovered, although the ordering of this set within the original codeword $\mathbf{x}_n$ is unknown.

Each deleted symbol from the set $\{q_1, q_2, \ldots, q_s\}$ can be associated with a particular bit of the $s$ bits deleted from $f(\mathbf{x}_n)$. There are at most $s!$ such possible associations.

To prove the lemma, we will use the approach of decoding symbol by symbol. In other words, select a random symbol, and insert the bit associated with this symbol into $f(\mathbf{x}'_{n-s})$, thereby obtaining $f(\mathbf{x}'_{n-s+1})$. Insert the symbol into $\mathbf{x}'_{n-s}$ to obtain $\mathbf{x}'_{n-s+1}$ that corresponds to the above relational sequence. Then, select another random deleted symbol, and repeat the above process until all $s$ deleted symbols have been used and $\mathbf{x}_n$ is obtained. Note that this $\mathbf{x}_n$ need not be the same as the original codeword.

Now we want to show that for a particular association between a deleted symbol from $\mathbf{x}_n$ and a deleted bit from $f(\mathbf{x}_n)$ there is at most one possible correction. Again consider the sequence

$$\ldots x_{j-1} \overset{f(x_j)}{\frown} x_j \overset{f(x_{j+1})}{\frown} x_{j+1} \ldots$$

Assume that the bit $x_j$ is deleted. If $f(x_j) = f(x_{j+1})$, then $x_{j-1}$ and $x_{j+1}$ will have the same relation as $f(x_j)$ and so the position of the deleted symbol in $\mathbf{x}_n$ will correspond to one of the positions in the run where the corresponding bit has been deleted in $f(\mathbf{x}_n)$. If, on the other hand, $f(x_j) \neq f(x_{j+1})$, then the relation between $x_{j-1}$ and $x_{j+1}$ can correspond to either $f(x_j)$ or $f(x_{j+1})$. For example, if $f(x_j) = 0$, $f(x_{j+1}) = 1$ and $x_{j-1} > x_{j+1}$, then the position of the deleted symbol in $\mathbf{x}_n$ actually corresponds to the bit just before the run where the corresponding bit has been deleted in $f(\mathbf{x}_n)$. If the deleted symbol in $\mathbf{x}_n$ is associated with the correct[2] deleted bit in $f(\mathbf{x}_n)$, then, due to Tenengol'ts [4], we know that there is only one possible correction. On the other hand, if the deleted symbol in $\mathbf{x}_n$ is associated with an incorrect[3] deleted bit in $f(\mathbf{x}_n)$, then there is at most one possible correction. This follows because a run in $f(\mathbf{x}_n)$ corresponds to a monotonically increasing (decreasing) sequence in $\mathbf{x}_n$.

Finally, we need to show that, for a particular set of associations between deleted symbols from $\mathbf{x}_n$ and deleted bits from $f(\mathbf{x}_n)$, the order in which the symbols are corrected from $\mathbf{x}'_{n-s}$ up to $\mathbf{x}_n$ do not alter the final sequence $\mathbf{x}_n$ obtained. Consider some $q$-ary codeword $\mathbf{x}_n$ and assume that after $s$ deletions we obtain $\mathbf{x}'_{n-s}$. It is obvious that no matter in which order the symbols are deleted, the same vector $\mathbf{x}'_{n-s}$ will be

[2]By "correct", we mean that the particular deleted symbol in $\mathbf{x}_n$ is that symbol that is actually responsible for the corresponding deleted bit in $f(\mathbf{x}_n)$.

[3]By "incorrect", we mean the opposite of "correct" in the previous footnote.

obtained. Since this is the reverse process of correction (where we insert the deleted symbols) symbol by symbol, it follows that the same $\mathbf{x}_n$ will be obtained from $\mathbf{x}'_{n-s}$, irrespective of the order in which the symbols are inserted. ∎

The above lemma shows that if $s$ symbols are deleted from $\mathbf{x}_n \in \hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$, decoding $\mathbf{x}'_{n-s}$ symbol by symbol will lead to at most $s!$ codewords that are also in $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$. Denote these $\sigma \leq s!$ codewords by the set $\{\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \ldots, \mathbf{x}_n^{(\sigma)}\}$, amongst which is the original codeword $\mathbf{x}_n$.

*Example 2:* Consider again $\mathbf{x}_7 = (3244015)$ from Example 1. Assume that the fourth and sixth symbols are deleted. Then $\mathbf{x}'_5 = (32405)$ and $f(\mathbf{x}'_5) = (\times 0101)$. At the receiver, we know that $f(\mathbf{x}_7) = (\times 011011)$ and that the symbols 4 and 1 were deleted. First associate the deleted symbol 1 with the bit deleted from the second run in $f(\mathbf{x}_7)$ and the symbol 4 with the bit deleted from the last run. Then,

$$1 : f(\mathbf{x}'_6) = (\times 01101) \rightarrow \mathbf{x}'_6 = (312405),$$

and

$$4 : f(\mathbf{x}_7) = (\times 011011) \rightarrow \mathbf{x}_7 = (3124045).$$

Now, reverse the order of correction. Then,

$$4 : f(\mathbf{x}'_6) = (\times 01011) \rightarrow \mathbf{x}'_6 = (324045),$$

and

$$1 : f(\mathbf{x}_7) = (\times 011011) \rightarrow \mathbf{x}_7 = (3124045).$$

Now, associate the deleted symbol 4 with the bit deleted from the second run in $f(\mathbf{x}_7)$ and 1 with the bit deleted from the last. Then,

$$4 : f(\mathbf{x}'_6) = (\times 01101) \rightarrow \mathbf{x}'_6 = (324405),$$

and

$$1 : f(\mathbf{x}_7) = (\times 011011) \rightarrow \mathbf{x}_7 = (3244015).$$

Then, reverse the order of correction, so that

$$1 : f(\mathbf{x}'_6) = (\times 01011) \rightarrow \mathbf{x}'_6 = (324015),$$

and

$$4 : f(\mathbf{x}_7) = (\times 011011) \rightarrow \mathbf{x}_7 = (3244015).$$

For a given association between the deleted symbols from $\mathbf{x}_7$ and deleted bits from $f(\mathbf{x}_7)$, the same decoded codeword is obtained irrespective of the ordering in which the symbols are corrected. For this example, we have that $\mathbf{x}_7^{(1)} = (3124045)$ and $\mathbf{x}_7^{(2)} = (3244015)$.

We wish to construct $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ from $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ such that $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ is an $s$ insertion/deletion correcting code. This will be achieved through a purging process. This is done in the following manner:

1) Select any $\mathbf{x}_n \in \hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$. For all possible combinations of $s$ deletions in $\mathbf{x}_n$, determine the sets $\mathcal{D} = \{\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \ldots, \mathbf{x}_n^{(\sigma)}\}$, where $\sigma \leq s!$. We will refer to the set $\mathcal{D}$ as a *decoding set*. For all such sets $\mathcal{D}$, purge the codewords in $\mathcal{D}$ from $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ except $\mathbf{x}_n$.
2) Select a codeword $\mathbf{y}_n \in \hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ such that $\mathbf{y}_n$ is not equal to any of the previously selected codewords or any of the previously purged codewords. Then repeat the purging process as in Step 1.
3) Repeat Step 2 until there are no codewords left in $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ that can be selected.

It should be noted that during the purging process at Step 2, it can never happen that a previously selected codeword is purged. To see why, let $\mathbf{y}_n$ represent the currently selected codeword and let $\mathbf{x}_n$ represent some previously selected codeword. For all possible combinations of $s$ deletions in $\mathbf{x}_n$, the codewords in $\mathcal{D} = \{\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \ldots, \mathbf{x}_n^{(\sigma)}\}$, except $\mathbf{x}_n$, have been purged at some previous step. The codeword $\mathbf{y}_n$ is selected such that $\mathbf{y}_n \neq \mathbf{x}_n$ and $\mathbf{y}_n \notin \mathcal{D}$, for all possible sets $\mathcal{D}$. Therefore, it follows that, if $\mathcal{D}'$ represents some decoding set after $s$ deletions in $\mathbf{y}_n$, then $\mathbf{x}_n \notin \mathcal{D}'$ for all such possible sets.

*Theorem 1:* The set $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ is an $s$ insertion/deletion correcting code.

*Proof:* In order to prove that $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ is an $s$ insertion/deletion correcting code, we will prove that $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ is an $s$ deletion correcting code. The result then follows because it is known, due to Levenshtein [2], that any $s$ deletion (or insertion) correcting code is also an $s$ insertion/deletion correcting code.

Consider some $\mathbf{x}_n \in \mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$, for an arbitrary, but fixed vector $(\beta_1, \beta_2, \ldots, \beta_s)$. Assume that $s$ random deletions occur in $\mathbf{x}_n$, thereby giving $\mathbf{x}'_{n-s}$. Let $\mathcal{Q} = \{q_1, q_2, \ldots, q_s\}$ represent the values of the $s$ deleted symbols. We need to show that we can uniquely decode $\mathbf{x}_n$ from $\mathbf{x}'_{n-s}$.

We can recover $f(\mathbf{x}_n)$ from $f(\mathbf{x}'_{n-s})$ as $g(\mathbf{x}_n) \in \mathcal{C}_s(n-1)$. Furthermore, we can determine the set $\mathcal{Q}$, as shown in Lemma 1. By decoding symbol by symbol, we obtain a decoding set $\mathcal{D} = \{\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \ldots, \mathbf{x}_n^{(\sigma)}\}$, where $\sigma \leq s!$, $\mathcal{D} \subset \hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ and $\mathbf{x}_n^{(i)} = \mathbf{x}_n$ for some $1 \leq i \leq \sigma$ (see Lemma 2). Due to the purging process used to obtain $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ from $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$, we know that $(\mathcal{D} \backslash \{\mathbf{x}_n\}) \cap \mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s)) = \emptyset$. Therefore, we can uniquely decode $\mathbf{x}_n$. ∎

It is clear that $\mathcal{T}_1(q, n, \beta)$ is in fact the Tenengol'ts code. Therefore, $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ is a generalisation of Tenengol'ts' construction.

The use of a purging process in constructing $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ from $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ is a drawback of the construction of the code $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$, especially for larger $n$. It is possible that there exists an additional constraint that creates $\mathcal{T}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$ from $\hat{\mathcal{T}}_s(q, n, (\beta_1, \beta_2, \ldots, \beta_s))$[4].

## IV. Some Thoughts Regarding Cardinality

Let $M_s(q, n)$ denote the cardinality of a maximal $q$-ary code with codewords of length $n$ capable of correcting $s$

[4]Up till now, the attempts by the authors to find such a constraint have been futile.

insertion/deletion errors. Levenshtein [9] showed that, for a fixed $s$ and $q$, and as $n \to \infty$[5]

$$\frac{(s!)^2 q^{n+s}}{(q-1)^{2s} n^{2s}} \lesssim M_s(q,n) \lesssim \frac{s! q^n}{(q-1)^s n^s}. \tag{8}$$

Therefore, (8) gives an asymptotic lower and upper bound on $M_s(q,n)$.

The cardinality of $\mathcal{T}_s(q,n)$[6] is dependent on the cardinality of the code $\mathcal{C}_s(n-1)$. Since the cardinality of the Helberg code is low[7], we will give a conjecture regarding the cardinality of $\mathcal{C}_s(n)$ based on an upper bound of a maximal binary multiple insertion/deletion correcting code derived by Levenshtein [2], upon which we will examine the cardinality of $\mathcal{T}_s(q,n)$. Needless to say, these results are hypothetical, but instructive nevertheless.

*Conjecture 1:* There exists a sequence of codes that are constructed by partitioning $\mathcal{A}_2^n$ into $c(n+1)^s$ codebooks, where $c$ is some fixed positive integer, such that each codebook is capable of correcting $s$ insertion/deletion errors.

The above conjecture essentially describes a code that would be a generalisation of Levenshtein's code. Note that the above conjecture implies that $|\mathcal{C}_s(n)| \geq 2^n/c(n+1)^s$, where $|\mathcal{C}_s(n)|$ is the cardinality of $\mathcal{C}_s(n)$[8], and that according to (8), $M_s(2,n) \lesssim s!2^n/n^s$. The bounds that are derived next are based on the above conjecture.

If $q$ is a prime number, then

$$|\hat{\mathcal{T}}_s(q,n)| \geq \frac{q^n}{cq^s n^s}. \tag{9}$$

During the purging process used to construct $\mathcal{T}_s(q,n)$ from $\hat{\mathcal{T}}_s(q,n)$, for some selected codeword $\mathbf{x}_n \in \hat{\mathcal{T}}_s(q,n)$, at most $(s!-1)\binom{n}{s}$ codewords are purged. Therefore

$$|\mathcal{T}_s(q,n)| > \frac{|\hat{\mathcal{T}}_s(q,n)|}{n^s}, \tag{10}$$

since $s!\binom{n}{s} < n^s$. Hence, for $q$ prime

$$|\mathcal{T}_s(q,n)| > \frac{q^n}{cq^s n^{2s}}. \tag{11}$$

Then the ratio of the asymptotic lower bound in (8) to the expression on the right-hand side of (11) is $c(s!)^2 q^{2s}/(q-1)^{2s}$. Therefore, for the case where $q$ is prime, we see that the lower bound on the cardinality of $\mathcal{T}_s(q,n)$ in (11) is within a constant factor from the asymptotic lower bound on $M_s(q,n)$ in (8), for a fixed $s$ and $q$.

[5]The notation $a(n) \lesssim b(n)$ denotes that $\lim_{n\to\infty} a(n)/b(n) \leq 1$ (see [2]).

[6]In this section, we will use $\mathcal{T}_s(q,n)$ as a shortened notation of $\mathcal{T}_s(q,n,(\beta_1,\beta_2,\ldots,\beta_s))$. Similarly, also $\hat{\mathcal{T}}_s(q,n)$ for $\hat{\mathcal{T}}_s(q,n,(\beta_1,\beta_2,\ldots,\beta_s))$.

[7]Note that it is shown in [10] that $2^{\lceil n/s \rceil + 1}$ is an upper bound on the cardinality of the Helberg code. Furthermore, for a fixed $s$ and sufficiently large $n$, this loose upper bound is less than the lower asymptotic bound from (8) for $q = 2$.

[8]Note that in [11] it is shown that there exists a $\mathcal{C}_2(n)$ such that $|\mathcal{C}_2(n)| \geq 2^n/(n+1)^2$ (i.e. where $c = 1$) for $n \leq 12$.

For the case where $q$ is not a prime,

$$|\mathcal{T}_s(q,n)| > \frac{q^n}{c(2q)^s n^{2s}}. \tag{12}$$

This follows from Bertrand's Postulate [12], which states that there exists at least one prime number $p$ such that $q < p \leq 2q$.

## V. Conclusion

In this paper we have described a code construction which is a generalisation of Tenengol'ts' non-binary single insertion/deletion correcting code. Such a construction is based on a binary multiple insertion/deletion correcting code. We have proven that such a construction is a non-binary multiple insertion/deletion correcting code. Furthermore, we have shown that the cardinality of this construction is at least within a constant factor of the asymptotic lower bound on the cardinality of a maximal non-binary multiple insertion/deletion correcting code provided that there exists a near optimal binary multiple insertion/deletion correcting code.

## References


[1] R. R. Varshamov and G. M. Tenengol'ts, "Codes which correct single asymmetric errors," *Automation and Remote Control*, vol. 26, no. 2, pp. 286–290, 1965, originally published in *Avtomatika i Telemekhanika*, vol. 26, no. 2, pp. 288–292, 1965 (in Russian).

[2] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics-Doklady*, vol. 10, no. 8, pp. 707–710, Feb. 1966, originally published in *Doklady Akademmi Nauk SSSR*, vol. 162, no. 4, pp. 845–848, 1965 (in Russian).

[3] N. J. A. Sloane, "On single-deletion-correcting codes," in *Codes and Designs*, K. T. Arasu and A. Seress, Eds. Walter de Gruyter, Berlin: Ray-Chaudhuri Festschrift, 2002, pp. 273–291. [Online]. Available: http://www2.research.att.com/∼njas/doc/dijen.ps

[4] G. M. Tenengol'ts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Inform. Theory*, vol. 30, no. 5, pp. 766–769, Sept. 1984.

[5] A. S. J. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Trans. Inform. Theory*, vol. 48, no. 1, pp. 305–308, Jan. 2002.

[6] K. A. S. Abdel-Ghaffar, F. Paluncić, H. C. Ferreira, and W. A. Clarke, "Some notes on the Helberg code," submitted to *IEEE Trans. Inf. Theory*.

[7] L. Dolecek and V. Anantharam, "Repetition error correcting sets: Explicit constructions and prefixing methods," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 4, pp. 2120–2146, Jan. 2010.

[8] R. R. Varshamov, "A class of codes for asymmetric channels and a problem from the additive theory of numbers," *IEEE Trans. Inform. Theory*, vol. 19, no. 1, pp. 92–95, Jan. 1973.

[9] V. I. Levenshtein, "Bounds for deletion/insertion correcting codes," in *Proceedings of the 2002 IEEE International Symposium on Information Theory*, Lausanne, Switzerland, June 30–July 5, 2002, p. 370.

[10] F. Paluncić, K. A. S. Abdel-Ghaffar, H. C. Ferreira, and W. A. Clarke, "A multiple insertion/deletion correcting code for run-length limited sequences," submitted to *IEEE Trans. Inf. Theory*.

[11] T. G. Swart and H. C. Ferreira, "A note on double insertion/deletion correcting codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 1, pp. 269–273, Jan. 2003.

[12] G. H. Hardy and E. M. Wright, *An Introduction to The Theory of Numbers*, 5th ed. London: Oxford University Press, 1979.