An Improvement of Non-binary Code Correcting Single $b$-Burst of Insertions or Deletions

Toyohiko Saeki and Takayuki Nozaki
Dept. of Informatics, Yamaguchi University, JAPAN
Email: {g012vb,tnozaki}@yamaguchi-u.ac.jp

Abstract—This paper constructs a non-binary code correcting a single $b$-burst of insertions or deletions with a large cardinality. This paper also proposes a decoding algorithm of this code and evaluates a lower bound of the cardinality of this code. Moreover, we evaluate an asymptotic upper bound on the cardinality of codes which correct a single burst of insertions or deletions.

I. INTRODUCTION

In communication and storage systems, several symbols in a sequence are inserted or deleted for the synchronization errors. Levenshtein [1] proved that VT codes (constructed by Varshamov and Tenengolts [2] for error correction on the Z-channel) correct a single insertion or deletion. This code had been extended to non-binary single insertion or deletion [3] and to two adjacent insertion or deletion [4]. This code had been also extended to a binary [5] and a non-binary multiple insertion or deletion correcting code [6].

Cheng et al. [7] constructed a binary $b$-burst insertion or deletion correcting code, which corrects any consecutive insertion or deletion of length $b$. Schoeny et al. [8] improved this construction and showed that the resulting code has larger cardinality than the code constructed by Cheng et al. These constructions have been extended to permutation code [9], [10]. Nowadays, Schoeny et al. [11] gives a non-binary $b$-burst insertion or deletion correcting code.

In this paper, we construct a non-binary $b$-burst insertion or deletion correcting code with a larger cardinality. The key idea of the paper is to investigate the correcting capability of the non-binary shifted VT code, which is a component of non-binary $b$-burst insertion or deletion correcting codes. We also derive a lower bound of the number of codewords of the constructed non-binary $b$-burst insertion or deletion correcting code. Moreover, we show an asymptotic upper bound of the cardinality of the best non-binary $b$-burst insertion or deletion correcting code.

II. PRELIMINARIES AND PREVIOUS WORKS

This section briefly introduces previous works, i.e., insertion/deletion codes given in [2], [3], [7], [8], [11]. We use notations given in this section throughout the paper.

A. Notation and Definition

For integers $i, j$, define $[i, j] := \{ k \in \mathbb{Z} | i \leq k \leq j \}$ and $[i] := [0, i - 1]$, where $\mathbb{Z}$ stands the set of integers.

For a sequence $x = (x_1, x_2, \ldots, x_n) \in [q]^n$, we denote the subsequence of $x$ whose $s$-th symbol is deleted, by $x_{-s}$, i.e., $x_{-s} = (x_1, x_2, \ldots, x_{s-1}, x_{s+1}, \ldots, x_n)$. In this case, we say that a single deletion has occurred in $x$. If $y$ is an output of the single insertion channel with an input $x$, there exists $i$ such that $y_{-1} = x$. For a sequence $x \in [q]^n$, a symbol $\lambda \in [q]$, and an integer $s \in [1, n + 1]$, we denote $x_{r(s, \lambda)} = (x_1, x_2, \ldots, x_{s-1}, \lambda, x_s, \ldots, x_n)$.

A run of length $r$ of a sequence $x$ is a subsequence of $x$ such that $x_i = x_{i+1} = \cdots = x_{i+r-1}, x_i \neq x_{i+1} \ (\text{for} \ i > 1)$, and $x_{i+r} \neq x_{i+r} \ (\text{for} \ i + r \leq n)$.

Remark 1: For a sequence $x = (1, 0, 0, 1, 1)$, $(x_2, x_3)$ is a run of length 2 and we have

$$x_{-2} = x_{-3} = (1, 0, 1, 1, 1).$$

From this, we see that we receive the same subsequences if a symbol in the same run is deleted under the single deletion channel. In other words, in the single deletion channel, even if one can correct a deletion, one cannot detect which symbol in a run is deleted.

Similarly, we get

$$x_{r(2, 0)} = x_{r(3, 0)} = x_{r(4, 0)} = (1, 0, 0, 0, 1, 1, 1).$$

Hence, in the single insertion channel, we receive the same sequence if the symbol $\lambda$ is inserted into a run of $\lambda$.

We refer to exactly $b$ consecutive deletions as a single $b$-burst deletion. We define $x_{-[i, i+b]} := (x_1, x_2, \ldots, x_i, x_{i+b+1}, x_{i+b+2}, \ldots, x_n)$. In words, when the $b$ consecutive, namely from $i$-th to $(i + b - 1)$-th, symbols of $x$ are deleted, we denote it, by $x_{-[i, i+b-1]}$. If $y$ is an output of the single $b$-insertion channel with an input $x$, there exists an integer $i$ such that $y_{-[i, i+b-1]} = x$.

A code which corrects single $b$-burst deletions (resp. insertions) is called a single $b$-burst deletion (resp. insertion) correcting code. A code is $b$-burst insertion/deletion correcting if it corrects single $b$-burst insertions or single $b$-burst deletions. Similarly, we define the terms: single deletion correcting code, single insertion correcting code, and single insertion/deletion correcting code.

The following theorem given in [8] shows a relationship between single $b$-burst deletion correcting codes and single $b$-burst insertion correcting codes.

---

1Section II-A will give the details of definition of the notation “insertion/deletion”.
Theorem 1: [8, Theorem 1] A code is a b-burst deletion correcting code if and only if it is a b-burst insertion correcting code. This theorem holds for not only binary case but also non-binary case. Hence, when we prove a code is a b-burst insertion/deletion correcting code, we only need to prove it is a b-burst deletion correcting code.

B. Single Insertion/Deletion Correcting Code

The VT code is a single insertion/deletion correcting code. The VT code is defined by the code length n and a ∈ [n + 1] as follows:

\[ \text{VT}_a(n) = \{ x \in [2]^n | \sum_{i=1}^n ix_i \equiv a \pmod{n+1} \}. \]

Let \( [P] \) be the indicator function, which equals 1 if the proposition \( P \) is true and equals 0 otherwise. A mapping \( \sigma \) of a q-ary sequence \( (x_1, x_2, \ldots, x_n) \in [q]^n \) to a binary sequence \( (u_1, u_2, \ldots, u_{n-1}) \in [2]^{n-1} \) is defined by

\[ u_i = [x_i < x_{i+1}]. \]

We refer to the sequence \( u = \sigma(x) \) as the ascent sequence for \( x \). The non-binary VT code is a non-binary single insertion/deletion correcting code defined by the code length \( n \), \( a \in [n] \) and \( c \in [q] \) as follows:

\[ q\text{VT}_{a,c}(n, q) = \{ x \in [q]^n | \sum_{i=1}^n ix_i \equiv c \pmod{q}, \sigma(x) \in \text{VT}_a(n-1) \}. \]

C. Binary Burst Insertion/Deletion Correcting Code

This section briefly introduces the binary burst insertion/deletion correcting codes given in [7], [8]. Roughly speaking, those methods employ interleaving to construct the codes.

For simplicity, we assume that \( n \) is divided by \( b \). The \( b \times n \) matrix representation for a sequence \( x \) is given as

\[ A_b(x) = \begin{pmatrix} x_1 & x_{b+1} & \cdots & x_{n-b+1} \\ x_2 & x_{b+2} & \cdots & x_{n-b+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_b & x_{2b} & \cdots & x_n \end{pmatrix}. \] (1)

We denote the \( i \)-th row of this matrix, by \( A_b(x)_i \).

Example 1: Consider the 3-burst deletion channel with an input \( x \in [2]^{12} \). Assume that the output is \( x_{-}[6,8] \). Then, these matrix representations are

\[ A_3(x) = \begin{pmatrix} x_1 & x_4 & x_7 & x_{10} \\ x_2 & x_5 & x_8 & x_{11} \\ x_3 & x_6 & x_9 & x_{12} \end{pmatrix}, \]

\[ A_3(x_{-[6,8]}) = \begin{pmatrix} x_1 & x_4 & x_{10} \\ x_2 & x_5 & x_{11} \\ x_3 & x_9 & x_{12} \end{pmatrix}. \]

From these, we see that \( A_3(x_{-[6,8]})_i \) is a result of a single deletion to \( A_3(x)_i \). Moreover, we see that when the \( (1, i) \)-th entry of \( A_3(x) \) is deleted, the \( (j, i-1) \)-th or \( (j, i) \)-th entry is deleted for \( j \geq 2 \).

From the above example, for recovering a single b-burst deletion, one needs to correct a single deletion for each row of the matrix representation. Moreover, if one detects the position \( i \) of deletion in the first row, one needs to correct a deletion for a given two adjacent positions \( i-1, i \) in the other rows.

The code in [7, Sect.III-C] embeds a marker \( (0, 1, 0, 1, \ldots) \) in the first row of the matrix representation to detect the deletion position and employs substitution-transposition codes [12] in the other rows to correct a single deletion for a given two adjacent positions. Here, note that we are able to regard to the marker \( (0, 1, 0, 1, \ldots) \) as a codeword of a VT code with maximum run length 1.

Schoeny et al. [8] improved the construction of this code. The first row of the code in [8] is a run-length-limited VT code which is a VT code with maximum run length at most \( r \). From Remark 1, one detects the interval of deletion position with the length at most \( r \). The other rows of the code are the shifted-VT codes, which correct a single deletion for a given \( r+1 \) adjacent positions. Let \( S_{n,q}(r) \) be the set of sequences in \([q]^n\) with maximum run length at most \( r \). Then, the run-length-limited VT code and shifted-VT (SVT) code are defined as

\[ \text{RLL-VT}_a(n, r) = \text{VT}_a(n) \cap S_{n,2}(r), \]

\[ \text{SVT}_{d,e}(n, r) = \{ x \in [2]^n : \sum_{i=1}^n ix_i \equiv d \pmod{r}, \sum_{i=1}^n x_i \equiv e \pmod{2} \}, \]

for \( d \in [r] \) and \( e \in [2] \). By using those codes, the binary single b-burst correcting code is constructed as:

\[ C_{2,b} = \{ x : A_b(x)_i \in \text{RLL-VT}_a(n/b, r), \forall i \in [2,b] \ A_b(x)_i \in \text{SVT}_{d,e}(n/b, r+1) \}. \]

D. Decoding Algorithm for SVT codes

In this section, we briefly introduce the decoding algorithm for the SVT codes. The details of decoding algorithms are in [8, Appendix C].

Firstly, we consider the case of deletion correction. Assume that we employ \( \text{SVT}_{d,e}(n, r) \). Let \( y \in [2]^{n-1} \) be the received sequence. Denote the first possible deletion position, by \( k \). The inputs of the deletion decoder are those, namely \( y, (d, e, n, r) \), and \( k \). We denote the estimated codeword, by \( x \). Let \( [s, t] \) be the interval of the run which contains the inserted symbol. The outputs of the deletion decoder are a pair of the estimated codeword \( x \) and interval \([s, t]\). We denote the deletion correcting algorithm for the SVT code, by \( \text{SVT-DC}(y, d, e, n, r, k) \rightarrow (x, [s, t]) \). For example, we have \( \text{SVT-DC}(0011, 0, 0, 5, 3, 2) \rightarrow (00011, [1, 3]) \).

Secondly, we consider the case of insertion correction. Let \( y \in [2]^{n+1} \) be the received sequence. Denote the first possible insertion position, by \( k \). We denote the estimated codeword, by \( x \). Let \( [s, t] \) be the interval of the run which contains the inserted symbol. We denote the insertion correction algorithm for the SVT code, by \( \text{SVT-IC}(y, d, e, n, r, k) \rightarrow (x, [s, t]) \). For example, we have \( \text{SVT-IC}(000111, 0, 0, 5, 3, 2) \rightarrow (00011, [4, 5]) \). The notations \( \text{SVT-DC} \) and \( \text{SVT-IC} \) will be used in Section III-C.
E. Non-binary Burst Insertion/Deletion Correcting Code

This section introduces the non-binary b-burst insertion/deletion correcting code given in [11].

By a straightforward construction, one obtains the non-binary b-burst insertion/deletion correcting code. Similar to the construction of non-binary VT code, we employ the mapping \( \sigma \) given in Sect. II-B. The non-binary run-length-limited VT code and the non-binary SVT code are defined as:

\[
\text{RLL}_q\text{-VT}_{a,c}(n,r,q) := q\text{VT}_{a,c}(n,q) \cap S_{a,q}(r),
\]

\[
\text{qSVT}_{d,e,f}(n,r,q) := \{ x \in [q]^n \mid \sum_{i=1}^{n} x_i \equiv f \pmod{q}, \sigma(x) \in \text{SVT}_{d,e,f}(n-1, r) \},
\]

where \( a \in [n], c \in [q], d \in [r], e \in [2], \) and \( f \in [q] \). Schoney et al. [11] showed the following lemma:

Lemma 1 ([11, Lemma 1]): For all \( d \in [r], e \in [2], \) and \( f \in [q] \), the code \( \text{qSVT}_{d,e,f}(n,r,q) \) corrects a single insertion/deletion for a given \( r \) and \( 1 \) adjacent positions.

As the result, they constructed the following non-binary single b-burst insertion/deletion correcting code:

\[
\hat{C}_{q,b} := \{ x \mid A_b(x), \hat{x} \in \text{RLL}_q\text{-VT}_{a,b}(n/b, r, q), \forall i \in [2, b] \ A_b(x), \hat{x} \in \text{qSVT}_{d,e,f}(n/b, r + 2, q) \}.
\]

III. MAIN RESULTS

This section constructs a non-binary burst insertion/deletion correcting code with a large cardinality. Section III-A gives the main theorem and construction of the code. Section III-B proves that the code is a non-binary burst insertion/deletion correcting code. Section III-C provides the decoding algorithm for the code. Section IV will evaluate the asymptotic cardinality of the code and show a numerical example.

A. Code Construction And Main Theorem

We investigate the correcting capability of the non-binary SVT code. As a result, we obtain that the code corrects a single insertion/deletion in a longer range as the following theorem.

Theorem 2: For all \( d \in [r], e \in [2], \) and \( f \in [q] \), the code \( \text{qSVT}_{d,e,f}(n,r,q) \) corrects a single insertion/deletion for a given \( r \) adjacent positions.

Based on this result, we construct a code:

\[
C_{q,b} := \{ x \mid A_b(x), \hat{x} \in \text{RLL}_q\text{-VT}_{a,b}(n/b, r, q), \forall i \in [2, b] \ A_b(x), \hat{x} \in \text{qSVT}_{d,e,f}(n/b, r + 1, q) \}.
\]

Moreover, we show the following theorem.

Theorem 3: The code \( C_{q,b} \) corrects a single b-burst insertion/deletion.

B. Proof of Theorems

In this section, we prove Theorem 2 and 3. Now, we will derive several lemmas to prove Theorem 2. The following lemma clarifies the effect of a single deletion in a sequence to its ascent sequence.

Lemma 2: Denote \( u = \sigma(x) \). Then, \( \sigma(x,\cdot) = u_{\cdot} \) or \( \sigma(x,\cdot) = u_{\cdot} \) holds.

Proof: Denote \( w = \sigma(x,\cdot) \). Obviously, it hold that \( w_j = u_j \) for \( j \in [1, i-2] \) and \( w_j = u_{j+1} \) for \( j \in [i, n-2] \). Hence, we will show that \( w_{i-1} = u_{i-1} \) or \( w_{i-1} = u_i \) holds.

Firstly, we assume \( x_{i-1} < x_i < x_{i+1} \). Then, \( u_{i-1} = u_i \) holds. Since \( x_{i-1} < x_{i+1}, w_{i-1} = 1 \) holds. Hence, \( w_{i-1} = u_{i-1} = u_i = 1 \) holds. Secondly, we assume \( x_i < x_{i-1} \) and \( x_i \geq x_{i+1} \). Then, \( u_{i-1} = 1 \) and \( u_i = 0 \) holds. If \( x_{i-1} < x_{i+1}, w_{i-1} = 1 \) otherwise \( w_{i-1} = 0 \). Hence, \( w_{i-1} = u_{i-1} = 1 \) or \( w_{i-1} = u_i = 0 \) holds.

The other cases are proved in a similar way.

Similarly, for an insertion, we obtain the following lemma.

Lemma 3: Denote \( u = \sigma(x) \). Then, \( \sigma(x,\cdot) = u_{\cdot} \) or \( \sigma(x,\cdot) = u_{\cdot} \) holds, where \( \delta = 0 \) or 1.

The following lemma is used for the proof of Theorem 2.

Lemma 4: Consider \( x, y \in \{ z \in [q]^n \mid \sum_{i=1}^{n} z_i \equiv f \pmod{q} \} \) such that \( x \neq y \) and \( x_{\cdot} = y_{\cdot} \) for a pair of integers \( s < t \). Denote \( u = \sigma(x), v = \sigma(y), \) and \( \sigma(x,\cdot) = \sigma(y_{\cdot}) \). Then, the following hold:

1) If \( w = u_{\cdot} = v_{\cdot} \), then there exist \( i, j \in [s, t] \) such that \( u_i \neq u_j \)

2) For a pair of integers \( (\alpha, \beta) \in \{(0, 0), (0, 1), (1, 1)\} \), if \( w = u_{\cdot} = v_{\cdot} \) and \( u_{\cdot} = v_{\cdot} = \gamma \), there exist \( i \in [s - \alpha + 1, t - \beta] \) such that \( u_i \neq \gamma \).

Proof: From Lemma 2, we have \( w = u_{\cdot} = v_{\cdot} \), or \( w = v_{\cdot} = v_{\cdot} \), and \( w = u_{\cdot} = v_{\cdot} \). Hence, \( w = u_{\cdot} = v_{\cdot} \) holds for a pair of integers \( (\alpha, \beta) \in \{(0, 0), (0, 1), (1, 1)\} \). We have

\[
\sum_{i=s}^{n} x_i - \sum_{i=1}^{n} y_i \pmod{q} = x_s - y_t,
\]

where the first equivalence follows from \( x, y \in \{ z \in [q]^n \mid \sum_{i=1}^{n} z_i \equiv f \pmod{q} \} \) and the second equality follows from \( x_{\cdot} = y_{\cdot} \). Since \( x_s, y_t \in [q] \), we get

\[
x_s = y_t.
\]

From \( x_{\cdot} = y_{\cdot} \) and \( u_{\cdot} = u_{\cdot} = v_{\cdot} \), we have

\[
x_i = \begin{cases} y_i, & (i \in [s - \alpha + 1, t + 1, n]), \\ y_{i-1}, & (i \in [s + 1, t]), \end{cases}
\]

\[
u_i = \begin{cases} v_i, & (i \in [s - \alpha + 1, t + 1, n - 1]), \\ v_{i-1}, & (i \in [s - \alpha + 1, t - \beta]). \end{cases}
\]

Firstly, we prove the case 1), i.e., the case of \( (\alpha, \beta) = (1, 0) \).

Let us hypothesize \( u_s = u_{s+1} = \cdots = u_t = 0 \). From (6), we get \( u_{s-1} = v_s = v_{s+1} = \cdots = v_{t-1} = 0 \). Hence, we have

\[
x_s \geq x_{s+1} \geq \cdots \geq x_{t+1}, \quad y_s \geq y_{s+1} \geq \cdots \geq y_{t+1}.
\]

Note that \( x_t = y_{t-1} \) and \( x_{t+1} = y_s \). Follow from (5), from (4), (5) and (7), we have

\[
x_s \geq x_{s+1} \geq \cdots \geq x_t \geq y_t \geq y_{t-1} \geq \cdots \geq y_s = x_s,
\]

\[
x_s \geq x_{s+1} \geq y_s \geq y_{s+1} \geq \cdots \geq y_t = x_s.
\]

Note that both ends of these equations are \( x_s \). Hence, these give

\[
x_s = x_{s+1} = \cdots = x_t = y_s = y_{s+1} = \cdots = y_t.
\]
From this equation and (5), we get $x = y$. This contradicts $x \neq y$. Next, let us hypothesize $u_s = u_{s+1} = \cdots = u_t = 1$. Similarly, we get

$$x_s < x_{s+1} < \cdots < x_{t+1}, \quad y_{s-1} < y_s < \cdots < y_t.$$  

Note that $x_{s+1} = y_s$ follows from (5). Combining those and (4), we have the following contradiction

$$x_s < x_{s+1} < \cdots < x_{t} = y_{t-1} < y_t = x_s.$$  

Thus, we obtain the case 1).

Secondly, we prove the case 2), i.e. the case of $(\alpha, \beta) \in \{(0, 0), (0, 1), (1, 1)\}$. From the assumption, we have $u_{s-\alpha} = v_{t-\beta} = \gamma$. Now, let us hypothesize $u_i = \gamma$ for all $i \in [s - \alpha + 1, t - \beta]$. Suppose $\gamma = 0$. Then, $u_i = v_i = 0$ for all $i \in [s - \alpha, t - \beta]$. Hence, we have

$$x_s - \alpha \geq x_{s-\alpha+1} \geq \cdots \geq x_{t-\beta+1},$$

$$y_s - \alpha \geq y_{s-\alpha+1} \geq \cdots \geq y_{t-\beta+1}.$$  

Combining (4), (5), (8), and (9), we get

$$x_s = x_{s+1} = \cdots = x_t = y_s = y_{s+1} = \cdots = y_t,$$

for all pair of $(\alpha, \beta) \in \{(0, 0), (0, 1), (1, 1)\}$. Combining this and (5), we get $x = y$. This contradicts $x \neq y$. Next, suppose $\gamma = 1$. Then, $u_i = v_i = 1$ for all $i \in [s - \alpha, t - \beta]$. Similarly, we get

$$x_s = x_{s+1} = \cdots = x_t = y_s = y_{s+1} = \cdots = y_t,$$

This leads the contradiction $x_s < x_s$. Thus, we obtain the case 2).

Now we will prove the two theorems.

**Proof of Theorem 2:** Let us hypothesize that there exists a pair of codewords $x, y \in qSVT_{d,e,f}(n, r, q)$ such that $x \neq y$ and $x_{s-\alpha} = y_{t-\beta}$ for two integers $s < t$ and $t - s < r$. Here, without loss of generality, we assume $s < t$. Denote $u = \sigma(x)$ and $v = \sigma(y)$. From Lemma 2, $\sigma(x_{s-\alpha}) = u_{s-(s-\alpha)}$ and $\sigma(y_{t-\beta}) = v_{s-(t-\beta)}$ holds for a pair of integers $(\alpha, \beta) \in \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. We have

$$0 \equiv \sum_{i=1}^{n-1} u_i - \sum_{i=1}^{n-1} v_i \pmod 2 = u_{s-\alpha} - v_{t-\beta},$$

where the first equivalence follows from $x, y \in qSVT_{d,e,f}(n, r, q)$, i.e., $u, v \in SVT_{d,e}(n - 1, r)$, and the second equation follows from $u_{s-(s-1)} = \sigma(x_{s-\alpha}) = \sigma(y_{t-\beta}) = v_{t-\beta}$. Hence, we get

$$u_{s-\alpha} = v_{t-\beta}.$$  

(10)

Since $u_{s-(s-\alpha)} = v_{s-(t-\beta)}$, we get (6). From (6) and (10), we have

$$\sum_{i=s}^{t} u_i - \sum_{i=s}^{t} v_i = \sum_{i=s}^{t} u_i = (t - s + \alpha - \beta) u_{s-\alpha}.$$  

(11)

Note that $x, y \in qSVT_{d,e,f}(n, r, q) \subset \{z \in [q]^n \mid \sum_{i=1}^{n} z_i \equiv f \pmod {q} \}$. Hence, the pair of $x$ and $y$ satisfies the conditions of Lemma 4. Firstly, we assume $(\alpha, \beta) = (1, 0)$. Then, case 1) of Lemma 4 derives

$$0 < \sum_{i=s}^{t} u_i \leq t - s.$$  

Recall that $t - s < r$. Combining the above with (11), we obtain for $u_{s-\alpha} = 0$

$$0 < \sum_{i=s}^{t} u_i - \sum_{i=s}^{t} v_i \leq t - s < r,$$

and for $u_{s-\alpha} = 1$

$$r - s < t - s - 1 < \sum_{i=s}^{t} u_i - \sum_{i=s}^{t} v_i \leq t - s - 1.$$  

However, these contradict $\sum_{i=s}^{t} u_i - \sum_{i=s}^{t} v_i \equiv 0 \pmod r$ which follows from $x, y \in qSVT_{d,e,f}(n, r, q)$, i.e., $u, v \in SVT_{d,e}(n - 1, r)$. Secondly, we assume $(\alpha, \beta) \in \{(0, 0), (0, 1), (1, 1)\}$. Then, case 2) of Lemma 4 derives

$$0 < \sum_{i=s}^{t} u_i \leq t - s + \alpha - \beta, \quad (\text{if } u_{s-\alpha} = 0),$$

$$0 < \sum_{i=s}^{t} u_i \leq t - s + \alpha - \beta, \quad (\text{if } u_{s-\alpha} = 1).$$

Since $t - s < r$ and $\alpha - \beta \leq 0$, we have $t - s + \alpha - \beta < r$

Combining the above and (11), we obtain

$$0 < \sum_{i=s}^{t} u_i - \sum_{i=s}^{t} v_i \leq r, \quad (\text{if } u_{s-\alpha} = 0),$$

$$0 < \sum_{i=s}^{t} u_i - \sum_{i=s}^{t} v_i < 0, \quad (\text{if } u_{s-\alpha} = 1).$$

Similarly, these contradict $\sum_{i=s}^{t} u_i - \sum_{i=s}^{t} v_i \equiv 0 \pmod r$.

Hence, we obtain the theorem.

- Theorem 3 is proved in a similar way to [8, Theorem 5].

**C. Decoding Algorithm**

Due to space limitations, we only describe the insertion/deletion correcting algorithm for the non-binary SVT code. In other words, we omit the decoding algorithm for $C_{q,b}$.

We denote the remainder when $i$ is divided by $q$, by $(i)_{q}$. Denote the transmitted sequence, by $x$. Algorithms 1 and 2 describe the deletion and insertion correcting algorithm for the SVT code, respectively. The set of inputs of those algorithms is the received sequence $y$, code parameters $(d, e, f, n, r)$, and
Ensure:

estimated sequence

received sequence

Algorithm 1

| Require: Received sequence \( y \), code parameters \((d, e, f, n, r, \sigma)\), first possible deletion position \( k \)
| Ensure: Estimated sequence \( x^\prime \)
| \begin{align*}
1: & \quad \hat{x} \leftarrow (f - \sum_{i=1}^{n-1} y_i) / q \\
2: & \quad \text{if } y_{-(k, \hat{x})} \in qSVT_{d,e,f}(n, r, q) \text{ then}
3: & \quad x^\prime \leftarrow y_{-(k, \hat{x})}
4: & \quad \text{else}
5: & \quad (u, [s, t]) \leftarrow \text{SVT-DC}(\sigma(y), d, e, n-1, r, k) \\
6: & \quad s' \leftarrow \max\{s, k+1\}, t' \leftarrow \min\{t, k + r - 2\} \\
7: & \quad j \leftarrow t' + 1 \\
8: & \quad \text{if } u_{s'} = 0 \text{ (i.e., } u_{s'} = u_{s'+1} = \cdots = u_n = 0) \text{ then}
9: & \quad \text{for } i = s', s' + 1, \ldots, t' \text{ do}
10: & \quad \text{if } \hat{x} \geq y_i \text{ then}
11: & \quad \hat{x} \leftarrow i \text{ and go to Step 21}
12: & \quad \text{end if}
13: & \quad \text{end for}
14: & \quad \text{else}
15: & \quad \text{for } i = s', s' + 1, \ldots, t' \text{ do}
16: & \quad \text{if } \hat{x} < y_i \text{ then}
17: & \quad \hat{x} \leftarrow i \text{ and go to Step 21}
18: & \quad \text{end if}
19: & \quad \text{end for}
20: & \quad \text{end if}
21: & \quad x^\prime \leftarrow y_{r(j, \hat{x})}
22: & \quad \text{end if}
| |

first possible deletion/insertion position \( k \). The output of those algorithms is the estimated sequence.

In Algorithm 1, \( \hat{x} \) stands the deleted symbol and \( j \) represents the position of the deleted symbol. Step 1 calculates the deleted symbol since \( \sum_{i=1}^{n-1} y_i + \hat{x} = \sum_{i=1}^{n} x_i \equiv f \pmod{q} \). Step 2 checks whether the \( k \)-th symbol is deleted. If the condition of Step 2 does not satisfy, then the deletion position is in \([k + 1, k + r - 1]\). In such a case, from Lemma 2, \( \sigma(y) \) equals \( \sigma(x) \) with an integer \( i \in [k, k + r - 1] \). Hence, we obtain \( u = \sigma(x) \) as in Step 5. The algorithm searches the position of the deleted symbol in Steps 7-20.

In Algorithm 2, \( \hat{x} \) stands the inserted symbol and \( j \) represents the position of the inserted symbol. Step 1 calculates the inserted symbol since \( \sum_{i=1}^{n+1} y_i - \hat{x} = \sum_{i=1}^{n} x_i \equiv f \pmod{q} \). Step 2 checks whether the \( k \)-th symbol is inserted. If the condition of Step 2 does not satisfy, then the inserted position is in \([k + 1, k + r]\). In such case, from Lemma 3, \( \sigma(y) \) equals \( \sigma(x) - (i, \delta) \) with an integer \( i \in [k, k + r] \) and \( \delta \in \{0, 1\} \). Hence, we obtain \( u = \sigma(x) \) as in Step 5. The algorithm searches the position of the inserted symbol in Steps 7-11.

IV. THE NUMBER OF CODEWORDS

This section evaluates the gap between the lower bound of the cardinality of the constructed code and the upper bound of the cardinality of arbitrary non-binary \( b \)-burst insertion/deletion correcting codes. Moreover, we evaluate the number of codewords of the SVT codes by a numerical example for an evidence that the code in (3) has a larger cardinality.

A. LOWER BOUND OF CARDINALITY OF CONSTRUCTED CODE

In a similar way to [8, Lemma 2], we have the following lemma.

Lemma 5: The following holds

\[
|S_{n,q}(r)| \geq (q^r - n)q^{n-r}.
\]

By the pigeonhole principle and this lemma, we get the following two lemmas.

Lemma 6: The cardinality of non-binary run-length-limited VT code is lower bounds as:

\[
\max_{a \in [n], c \in [q]} |\text{RLL-}q\text{VT}_{a,c}(n, r, q)| \geq \frac{(q^r - n)q^{n-r} - 1}{n}.
\]

Lemma 7: The cardinality of non-binary SVT code is lower bounds as:

\[
\max_{d \in [r], e \in [2], f \in [q]} |q\text{SVT}_{d,e,f}(n, r, q)| \geq \frac{q^{n-1}}{2^r}.
\]

From those lemmas, we obtain a lower bound of cardinality of the constructed code.

Theorem 4: For all \( r \), the cardinality of \( C_{q,b} \) satisfies

\[
\max |C_{q,b}| \geq \frac{q^{n-b} - (b - nq^r)}{2^{b-1}(r+1)^{b-1}} .
\]

Substituting \( r = \log_q n \) in (12), we have

\[
\max |C_{q,b}| \geq \frac{2q^{n-b} - b - 1}{2^{b\log_q n + 1} b^{b-1}} .
\]

We define redundancy of a \( q \)-ary code \( C \) by \( n - \log_q |C| \). From (13), an upper bound of redundancy of \( C_{q,b} \) with the best parameter is

\[
b + \log_q n - \log_q (b - 1) + (b - 1) \log_q 2 \\
+ (b - 1) \log_q (\log_q n + 1).
\]
B. Upper Bound of Cardinality of Burst Insertion/Deletion Correcting Code

Let $C$ be the set of non-binary $b$-burst insertion/deletion correcting codes of length $n$. Define $M_b(n) := \arg\max_{C \in C} |C|$. In words, $M_b(n)$ is the non-binary $b$-burst insertion/deletion correcting code of length $n$ with maximum cardinality, i.e., $M_b(n)$ is the best code. The following theorem gives an upper bound of the cardinality of $M_b(n)$.

**Theorem 5:** For enough large $n$, the following holds:

$$|M_b(n)| \leq \frac{q^{n-b+1}}{(q-1)^n}.$$  

This theorem is proved in a similar way to [4, Lemma 1].

**Proof:** Define $m := n/b - 1$. Denote the number of runs in $x$, by $|x|$. For a positive integer $r$, define

$$M_b(r) := \{ x \in M_b(n) \mid \forall i \in [1, b] \ | A_b(x)_i| \geq r + 2 \},$$

$$M_2(r) := \{ x \in M_b(n) \mid \exists i \in [1, b] \ s.t. \ | A_b(x)_i| \leq r + 1 \}.$$  

Note that $M_b(n) = M_b(r) \cup M_2(r)$ and $M_1(r) \cap M_2(r) = \emptyset$. This leads

$$|M_b(n)| = |M_b(r)| + |M_2(r)| \quad \text{for all } r. \quad (15)$$

Now, we will derive upper bounds of $|M_1(r)|$ and $|M_2(r)|$.

Firstly, we consider $|M_1(r)|$. Denote $D(x) := \{ x_{i|i+i+b-1} \mid i \in [1, n-b+1] \}$. In words, $D(x)$ is a $b$-burst deletion ball for $x$, i.e., the set of sequences after $b$-burst deletion to $x$. The volume of $b$-burst deletion ball for $x$ is derived in [4] as

$$|D(x)| = 1 + \sum_{i=1}^{b} (|A_b(x)_i| - 1).$$

Since $|A_b(x)_i| \geq r + 2$ for all $i$, $|D(x)|$ is bounded by

$$|D(x)| \geq b(r+1) + 1.$$  

Since $M_b(n)$ is a $b$-burst deletion correcting code, $M_1(r)$ is also a $b$-burst deletion correcting code. Hence, $\bigcup_{x \in M_1(r)} D(x) \subseteq \left[ q \right]^{n-b}$ and $D(x) \cap D(y)$ for all $x, y \in M_1(r)$ hold. This leads $q^{n-b} \geq \sum_{x \in M_1(r)} |D(x)|$. Combining the above yields

$$q^{n-b} \geq |M_1(r)| (b(r+1) + 1).$$

As the result, we have an upper bound for $|M_1(r)|$ as follows:

$$|M_1(r)| \leq \frac{q^{n-b}}{b(r+1) + 1} < \frac{q^{n-b}}{b(r+1)} =: f(r). \quad (16)$$

Secondly, we derive an upper bound for $|M_2(r)|$. Define

$$B_{\leq r+1} := \{ x \in \left[ q \right]^{n} \mid \exists i.s.t. |A_b(x)_i| \leq r + 1 \},$$

$$B_{\leq r+1, i} := \{ x \in \left[ q \right]^{n} \mid |A_b(x)_i| \leq r + 1 \},$$

$$B_{j, i} := \{ x \in \left[ q \right]^{n} \mid |A_b(x)_i| = j \}.$$  

Then, the following holds

$$M_2(r) \subseteq B_{\leq r+1} = \bigcup_{i=1}^{b} B_{\leq r+1, i} = \bigcup_{i=1}^{b} \bigcup_{j=1}^{r+1} B_{j, i}.$$  

Now, the cardinality of $B_{j, i}$ is

$$|B_{j, i}| = \binom{m}{j-1} q^{n-m} (q-1)^{j-1}.$$  

These derives

$$|M_2(r)| \leq bq^{n-m} \sum_{j=0}^{r} \binom{m}{j} (q-1)^j.$$  

For $r < (1 - q^{-1})m$, the summation is bounded by (e.g. see [13, Exercise 5.8])

$$\sum_{j=0}^{r} \binom{m}{j} (q-1)^j \leq (q-1)^r \exp [mh_2(r/m)], \quad (17)$$

where $h_2(x) := -x \ln x - (1-x) \ln(1-x)$. For $r \geq \frac{q-1}{q} m$,

$$\sum_{j=0}^{r} \binom{m}{j} (q-1)^j \leq q^{m} - (q-1)^{r+1} \exp [mh_2((r+1)/m)] \leq \sqrt{2m}.$$  

where the last inequality follows from $\sum_{j=0}^{m} \binom{m}{j} (q-1)^j \geq (q+1)(q-1)^{r+1}$ and $(\frac{m}{r+1}) \geq \exp [mh_2((r+1)/m)]$. Thus, $|M_2(r)|$ is bounded by

$$|M_2(r)| \leq g(r) := \begin{cases} g_1(r), & \text{if } r < \frac{q-1}{q} m, \quad \text{(18)} \\
g_2(r), & \text{if } r \geq \frac{q-1}{q} m, \end{cases}$$

$$g_1(r) := bq^{n-m} (q-1)^r \exp [mh_2(r/m)],$$

$$g_2(r) := bq^{n-m} \left[ q^{m} - (q-1)^{r+1} \exp [mh_2((r+1)/m)/\sqrt{2m}] \right].$$

Combining (15), (16) and (18) yields

$$|M_b(n)| \leq \min_{r} (f(r) + g(r)). \quad (19)$$

Note that $f(r)$ is monotonically decreasing function and $g_1(r), g_2(r)$ are monotonically increasing functions. Firstly, we consider the case of $r < \frac{q-1}{2} m$. Let $\alpha$ be a positive real number. Define $\epsilon := \sqrt{\frac{\alpha \ln m}{m}}$. Substituting $r = (1 - q^{-1} - \epsilon)m$ yields

$$f \left( (1 - q^{-1} - \epsilon)m \right) \leq \frac{q^{n-b}}{b} \frac{1}{q^{1} (m-1) - \epsilon m} = \frac{q^{n-b+1}}{n(q-1)} (1 + O(\epsilon)),$$  

where the last equation follows from $(1 - \epsilon)^{-1} = 1 + O(\epsilon)$. Note that $\ln(1+x) = x - \frac{1}{2} x^2 + O(x^3)$. This leads

$$h_2(1 - q^{-1} - \epsilon) = - (1 - q^{-1} - \epsilon) \ln(q-1) + \ln q + \frac{1}{2} q^{-1} + O(\epsilon^3).$$

This yields

$$g_1 \left( (1 - q^{-1} - \epsilon)m \right) = bq^{n-m} \frac{q^2}{q^{1} - \epsilon m} \exp \left[ O(\epsilon^3 m) \right].$$

Hence, if $\alpha > 2q^{-1}/q$, $g_1(r) = o(f(r))$. Otherwise, $f(r) = O(g_1(r))$. Thus, for $r \leq \frac{q-1}{2} m$, (19) is evaluated as

$$\min_{r \leq \frac{q-1}{2} m} (f(r) + g(r)) \leq f \left( (1 - q^{-1})m - \sqrt{m \ln m} \right) = \frac{q^{n-b+1}}{n(q-1)} (1 + O \left( \sqrt{\frac{\ln m}{m}} \right)). \quad (20)$$
Comparing (20) and (21) leads the theorem.

Secondly, let us consider the case of $r \geq \frac{1}{q} m$. Recall that $f(r)$ and $g_2(r)$ are monotonically decreasing and increasing function, respectively. Since $f((1-q^{-1})m) < g_2((1-q^{-1})m)$, $f(r) < g_2(r)$ holds for all $r \geq \frac{2-1}{q} m$. Thus, for $r \geq \frac{2-1}{q} m$, (19) is evaluated as

$$
\min_{r \geq \frac{2-1}{q} m} (f(r) + g_2(r)) = bq^n + o(q^n).
$$

Comparing (20) and (21) leads the theorem.

From Theorem 5, the redundancy of $M_b(n)$ is lower bounded by

$$
b - \log_q 2 + \log_q n.
$$

By comparing (14), the gap of redundancy between the constructed code and the best code is upper bounded by

$$
- \log_q (b - 1) + b \log_q 2 + (b - 1) \log_q (\log_q n + 1).
$$

C. Numerical Example

Table I shows the number of codewords of the non-binary SVT code with best parameters for $n = 10, q = 4$, i.e., shows $\max_{d,e,f} |q\text{SVT}_{d,e,f}(10, r, 4)|$ for $r = 2, 3, \ldots, 10$. From Table I, we see that the number of codewords decreases for $r \leq 8$ as $r$ increases. In other $n, q$, we also observe the number of codewords decreases except that $r$ is nearly equals to $n$. Hence, for small $r$ (e.g., $r = \log_q n$, employed in (13)), we conclude that $C_{q,b}$ has a larger cardinality than $C_{q,b}^*$.

| $r$ | 2    | 3    | 4    | 5    | 6    |
|-----|------|------|------|------|------|
| $\text{Cardinality}$ | 66240 | 44028 | 33136 | 26475 | 22108 |
| $r$ | 7    | 8    | 9    | 10   |
| $\text{Cardinality}$ | 19000 | 17874 | 17918 | 18156 |

V. CONCLUSION AND FUTURE WORKS

In this paper, we have constructed a non-binary $b$-burst insertion/deletion correcting code with a larger cardinality and presented a decoding algorithm for the code. We also have derived a lower bound on the cardinality of the proposed code and an asymptotic upper bound on the cardinality of non-binary $b$ burst deletion correcting codes. Our future works are (1) construction of non-binary codes which correct a deletion burst of at most $b$ consecutive symbols and (2) deriving non-asymptotic upper bound on the maximum cardinality of any non-binary $b$ burst deletion correcting code.

ACKNOWLEDGMENT

The authors wish to thank to Dr. C. Schoeny for telling us [11]. This work was supported by JSPS KAKENHI Grant Number 16K16007.

REFERENCES

[1] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in Soviet physics doklady, vol. 10, no. 8, 1966, pp. 707–710.
[2] R. Varshamov and G. Tenenholtz, “Codes which correct single asymmetric errors,” Avtomatika i Telemekhanika, vol. 26, no. 2, pp. 288–292, 1965.
[3] G. Tenengolts, “Nonbinary codes, correcting single deletion or insertion (corresp.),” IEEE Transactions on Information Theory, vol. 30, no. 5, pp. 766–769, 1984.
[4] V. Levenshtein, “Asymptotically optimum binary code with correction for losses of one or two adjacent bits,” Problemy Kibernetiki, vol. 19, pp. 293–298, 1967.
[5] A. S. J. Helberg and H. C. Ferreira, “On multiple insertion/deletion correcting codes,” IEEE Transactions on Information Theory, vol. 48, no. 1, pp. 305–308, Jan. 2002.
[6] F. Paluni, T. G. Swart, J. H. Weber, H. C. Ferreira, and W. A. Clarke, “A note on non-binary multiple insertion/deletion correcting codes,” in 2011 IEEE Information Theory Workshop, Oct 2011, pp. 683–687.
[7] L. Cheng, T. G. Swart, H. C. Ferreira, and K. A. S. Abdel-Ghaffar, “Codes for correcting three or more adjacent deletions or insertions,” in 2014 IEEE International Symposium on Information Theory, June 2014, pp. 1246–1250.
[8] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, “Codes correcting a burst of deletions or insertions,” IEEE Transactions on Information Theory, vol. 63, no. 4, pp. 1971–1985, 2017.
[9] Y. M. Chee, V. K. Vu, and X. Zhang, “Permutation codes correcting a single burst deletion i: Unstable deletions,” in 2015 IEEE International Symposium on Information Theory (ISIT), June 2015, pp. 1741–1745.
[10] Y. M. Chee, S. Ling, T. T. Nguyen, V. K. Vu, and H. Wei, “Permutation codes correcting a single burst deletion ii: Stable deletions,” in 2017 IEEE International Symposium on Information Theory (ISIT), June 2017, pp. 2688–2692.
[11] C. Schoeny, F. Sala, and L. Dolecek, “Novel combinatorial coding results for dna sequencing and data storage,” in 2017 51st Asilomar Conference on Signals, Systems, and Computers, Oct 2017, pp. 511–515.
[12] K. A. S. Abdel-Ghaffar, “Detecting substitutions and transpositions of characters,” The Computer Journal, vol. 41, no. 4, pp. 270–277, 1998.
[13] R. G. Gallager, Information theory and reliable communication. Springer, 1968, vol. 2.

TABLE I

The cardinality of non-binary SVT codes with best parameters for $n = 10, q = 4$.

| $r$ | 2    | 3    | 4    | 5    | 6    |
|-----|------|------|------|------|------|
| $\text{Cardinality}$ | 66240 | 44028 | 33136 | 26475 | 22108 |
| $r$ | 7    | 8    | 9    | 10   |
| $\text{Cardinality}$ | 19000 | 17874 | 17918 | 18156 |