Efficient Systematic Encoding of Non-binary VT Codes

Mahed Abroshan
University of Cambridge
ma675@cam.ac.uk

Ramji Venkataramanan
University of Cambridge
ramji.v@eng.cam.ac.uk

Albert Guillén i Fàbregas
ICREA & Universitat Pompeu Fabra
University of Cambridge
guillen@ieee.org

***Abstract*—This paper addresses the problem of efficient encoding of non-binary Varshamov-Tenengolts (VT) codes. We propose a linear-time encoding method to systematically map binary message sequences onto VT codewords. The method provides a new lower bound on the size of q-ary VT codes of length n.**

I. Introduction

Designing codes for correcting deletions or insertions is well known to be a challenging problem; see, e.g., [1]–[7]. For the special case of correcting one insertion or deletion, there exists an elegant class of codes called Varshamov-Tenengolts (VT) codes. Binary VT codes were first introduced in [8] for channels with asymmetric errors. Later, Levenshtein [9] showed that they can be used for correcting a single deletion or insertion with a simple decoding algorithm whose complexity is linear in the code length [1]. Tenengolts subsequently introduced a non-binary version of VT codes, defined over a q-ary alphabet for any $q > 2$ [10]. The q-ary VT codes retain many of the attractive properties of the binary codes. In particular, they can correct deletion or insertion of a single symbol from a q-ary VT codeword with a linear-time decoder.

Given the simplicity of VT decoding, a natural question is: can one construct a linear-time encoder to efficiently map binary message sequences onto VT codewords? For binary VT codes, such an encoder was proposed by Abdel-Ghaffar and Ferriera [11]. (A similar encoder was also described in [12].) However, the issue of efficient encoding for non-binary VT codes has not been addressed previously, to the best of our knowledge. In this paper, we propose an efficient systematic encoder for non-binary VT codes. The encoder has complexity that is linear in the code length, and is systematic in the sense that the message bits are assigned to pre-specified positions in the codeword. The encoder also yields a new lower bound on the size of q-ary VT codes, for $q > 2$.

In [10, Sec. 5], Tenengolts introduced a systematic non-binary code that can correct a single deletion or insertion. However, this code is not strictly a VT code as its codewords do not necessarily share the same VT parameters. (Formal definitions of VT codes and their parameters are given in the next two sections.) Here we propose an encoder for VT codes defined in the standard way, noting that standard VT codes are a key ingredient in certain code constructions, e.g., [13].

This work has been funded in part by the European Research Council under ERC grant agreements 259663 and 725411, and by the Spanish Ministry of Economy and Competitiveness under grant TEC2016-78434-C3-1-R.

Notation: Sequences are denoted by capital letters, and scalars by lower-case letters. Throughout, we use n for the code length and k for the number of message bits mapped to the codeword. The set $\mathbb{Z}_q = \{0, 1, \cdots, q-1\}$ is the finite integer ring of size q. We consider the natural order for the elements of $\mathbb{Z}_q$, i.e., $0 < 1 \ldots < (q-1)$. The term dyadic index will be used to refer to an index that is a power of 2.

The rest of the paper is organized as follows. In the next section, we formally define binary VT codes and briefly review the systematic encoder from [11]. In Section III, we define the q-ary VT codes, and describe the systematic encoding method and the resulting lower bound on the size of the codes.

II. Binary VT codes

The VT syndrome of a binary sequence $S = s_1 s_2 \cdots s_n \in \mathbb{Z}_2^n$ is defined as

$$\mathsf{syn}(S) \triangleq \sum_{i=1}^{n} i\, s_i \mod (n+1). \tag{1}$$

For positive integers n and $0 \le a \le n$, the VT code of length n and syndrome a, is defined as

$$\mathcal{VT}_a(n) = \big\{ S \in \mathbb{Z}_2^n : \mathsf{syn}(S) = a \big\}, \tag{2}$$

i.e., the set of binary sequences S of length n that satisfy $\mathsf{syn}(S) = a$. For example, the VT code of length 3 and syndrome 2 is

$$\mathcal{VT}_2(3) = \Big\{ s_1 s_2 s_3 \ \in \mathbb{Z}_2^3 : \sum_{j=1}^{3} j\, s_j = 2 \bmod 4 \Big\} \tag{3}$$

$$= \{010, 111\}. \tag{4}$$

Each of the sets $\mathcal{VT}_a(n)$, $0 \le a \le n$, is a code that can correct a single deletion or insertion with a decoder whose complexity is linear in n. The details of the decoding algorithm can be found in [1].

The $(n+1)$ sets $\mathcal{VT}_a(n)$, $0 \le a \le n$, partition the set of all binary sequences of length n, i.e., each sequence $S \in \mathbb{Z}_2^n$ belongs to exactly one of the sets. Therefore, the smallest of the codes $\mathcal{VT}_a(n)$ will have at most $\frac{2^n}{n+1}$ sequences. Hence

$$\min_{0 \le a \le n} \frac{1}{n} \log_2 |\mathcal{VT}_a(n)| \le 1 - \frac{1}{n} \log_2(n+1). \tag{5}$$

An exact formula for the size $|\mathcal{VT}_a(n)|$ was given by Ginzburg [14]. The formula [1, Theorem 2.2] does not give an analytical expression for general n, but it shows that if $(n+1)$ is a power of 2, then $|\mathcal{VT}_a(n)| = 2^n/(n+1)$, for $0 \le a \le n$. Moreover, for general n, the formula can be used to deduce that the sizes of the codes $\mathcal{VT}_a(n)$ are all approximately $2^n/(n+1)$. In particular, for $a \in \{0, \dots, n\}$

$$\frac{2^n}{(n+1)} - 2^{(n+1)/3} \le |\mathcal{VT}_a(n)| \le \frac{2^n}{(n+1)} + 2^{(n+1)/3}. \quad (6)$$

Abdel-Ghaffar and Ferriera [11] proposed a systematic encoder to map k-bit message sequences onto codewords in $|\mathcal{VT}_a(n)|$, where $k = n - \lceil \log_2(n+1) \rceil$. We briefly review the encoding procedure as it is an ingredient of our systematic q-ary VT encoder. Consider a k-bit message $M = m_1 \cdots m_k$ to be encoded into a codeword $C = c_1 \cdots c_n \in \mathcal{VT}_a(n)$, for some $a \in \{0, 1, \cdots, n\}$. The number of "parity" bits is denoted by $t = n - k = \lceil \log_2(n+1) \rceil$. The idea is to use the code bits in dyadic positions, i.e., c_{2^i}, for $0 \le i \le (t-1)$, to ensure that $\mathsf{syn}(C) = a$. The encoding steps are:

1) Denote the first k non-dyadic indices by $\{j_1, \cdots, j_k\}$, where the indices are in ascending order, i.e., $j_1 = 3, j_2 = 5, \dots$ We set c_{j_i} equal to the message bit m_i, for $1 \le i \le k$.
2) First set the bits in all the dyadic positions to be zero and denote the resulting sequence by $C' = c'_1 \cdots c'_n$ (so that we have $c'_{2^i} = 0$ for $0 \le i \le t-1$ and $c'_{j_l} = m_l$ for $1 \le l \le k = (n-t)$). Define the deficiency d as the difference between the desired syndrome a and the syndrome of C'. That is,

$$d = a - \mathsf{syn}(C') \mod (n+1). \quad (7)$$

3) Let the binary representation of d be $d_{t-1} \dots d_1 d_0$, i.e., $d = \sum_{i=0}^{t-1} 2^i d_i$. Set $c_{2^i} = d_i$, for $0 \le i \le (t-1)$, to obtain C.

The rate of this systematic encoder is $R = 1 - \frac{1}{n}\lceil \log_2(n+1) \rceil$, regardless of the syndrome $a \in \{0, \dots, n\}$. Comparing with (5), we observe that the rate loss for the smallest VT code of length n is less than $\frac{1}{n}$. On the other hand, if $(n+1)$ is not a power of two, the rate loss for the larger VT codes may be higher due to codewords that are unused by the encoder. However this rate loss is unavoidable with any systematic encoder [11].

We remark that the dyadic positions are not the only set of positions that can be used for syndrome bits. For instance, the following set of indices also produce all syndromes:

$\{c_{i_0}, \cdots, c_{i_{t-1}}\}$ where $i_j = -2^j \pmod{n+1}$ for $0 \le j \le t-1$.

This can be helpful in some applications (see e.g. [13]) where some code bits are already reserved for prefixes or suffixes, and thus cannot be used as syndrome bits. In general, a set of positions $\{p_1, p_2, \cdots, p_r\}$ can be used for syndrome bits if for each syndrome $a \in \{0, \cdots, n\}$, there exists a subset $\mathcal{P} \subseteq \{p_1, p_2, \cdots, p_r\}$ such that

$$\sum_{j \in \mathcal{P}} p_j = a \pmod{n+1}. \quad (8)$$

In other words, for each $a \in \{0, \cdots, n\}$, there should exist binary coefficients $b_1, \cdots, b_r$ such that

$$\sum_{j=1}^{r} b_j p_j = a \pmod{n+1}. \quad (9)$$

III. Efficient Encoding for Non-binary VT codes

For any code length n, the VT codes over $\mathbb{Z}_q$, for $q > 2$ are defined as follows [10]. For each q-ary sequence $S = s_0 s_1 \cdots s_{n-1} \in \mathbb{Z}_q^n$, define a corresponding length $(n-1)$ auxiliary binary sequence $A_S = \alpha_1 \alpha_2 \dots \alpha_{n-1}$ as follows[1]. For $1 \le i \le n-1$,

$$\alpha_i = \begin{cases} 1 & \text{if } s_i \ge s_{i-1} \\ 0 & \text{if } s_i < s_{i-1}. \end{cases} \quad (10)$$

We also define the modular sum of S as

$$\mathsf{sum}(S) = \sum_{i=0}^{n-1} s_i \pmod{q}. \quad (11)$$

For $0 \le a \le n-1$ and $b \in \mathbb{Z}_q$, the q-ary VT code with length n and parameters (a, b) is defined as

$$\mathcal{VT}_{a,b}(n) = \{S \in \mathbb{Z}_q^n : \mathsf{syn}(A_S) = a, \, \mathsf{sum}(S) = b\}. \quad (12)$$

Each of the sets $\mathcal{VT}_{a,b}(n)$ is a code that can correct deletion or insertion of a single symbol with a decoder whose complexity is linear in the code length n. The details of the decoding algorithm can be found in [10, Sec. II].

Similarly to the binary case, the codes $\mathcal{VT}_{a,b}(n)$, for $0 \le a \le n-1$ and $b \in \mathbb{Z}_q$, partition the space $\mathbb{Z}_q^n$ of all q-ary sequences of length n. For a given n, there are nq of these codes, and hence the smallest of them will have no more than $\frac{q^n}{nq}$ sequences. Let R_{min} be the rate of the smallest of these codes, i.e.,

$$R_{\text{min}} \triangleq \min_{a,b} \frac{\log_2 |\mathcal{VT}_{a,b}(n)|}{n}, \quad (13)$$

where the minimum is over $0 \le a \le n-1$ and $b \in \mathbb{Z}_q$. We then have the bound

$$R_{\text{min}} \le \log_2 q - \frac{1}{n}\log_2 n - \frac{1}{n}\log_2 q \ \text{ bits/symbol.} \quad (14)$$

The encoding procedure described below yields a lower bound on the size of $|\mathcal{VT}_{a,b}(n)|$ (see Proposition 1), which shows that for $q \ge 4$,

$$\begin{aligned} R_{\text{min}} \ge & \log_2 q - \frac{1}{n}\lceil \log_2 n \rceil (3\log_2 q - 2\log_2(q-1)) \\ & - \frac{1}{n}(5\log_2(q-1) - 3\log_2 q) \ \text{ bits/symbol.} \end{aligned} \quad (15)$$

Kulkarni and Kiyavash [15] have shown that the size of any single deletion correcting q-ary code of length n is bounded by $\frac{q^n - q}{(q-1)(n-1)}$. This yields a rate upper bound R_{max} for any single deletion correcting code, where

$$R_{\text{max}} \le \log_2 q - \frac{\log_2(n-1)}{n} - \frac{\log_2(q-1)}{n}. \quad (16)$$

[1]For non-binary sequences, we start the indexing from 0 as this makes it convenient to describe the encoding procedure in Section III.

We now describe the encoder to map a sequence of message bits to a codeword of the q-ary VT code $\mathcal{VT}_{a,b}(n)$. For simplicity, we first assume that q is a power of two, and address the case of general q at the end of this section. We will map a k-bit message $M = m_1 \cdots m_k$ to a codeword in $\mathcal{VT}_{a,b}(n)$, where

$$k = (n-3t+3)\log_2 q + (t-3)(2\log_2 q - 1) + (\log_2 q - 1) \quad (17)$$

$$= n\log_2 q - t(\log_2 q + 1) - 2(\log_2 q - 1), \quad (18)$$

with $t = \lceil \log_2 n \rceil$. Hence the rate of our encoding scheme is

$$R = \log_2 q - \frac{\lceil \log_2 n \rceil (\log_2 q + 1)}{n} - \frac{2\log_2 q - 2}{n} \quad \frac{\text{bits}}{\text{symbol}}. \quad (19)$$

Our encoding method gives a lower bound on the size of any non-binary VT code of length n. An immediate lower bound on the size is 2^k, with k given by (18). The proposition below gives a slightly better bound, which is obtained by modifying the encoding method to map q-ary message sequences to q-ary VT codewords, rather than a binary message sequence to a q-ary VT codeword.

Proposition 1. *For $n \geq 6$, $q \geq 4$, and any $0 \leq a \leq n$, and $b \in \mathbb{Z}_q$, we have*

$$|\mathcal{VT}_{a,b}(n)| \geq (q-1)^{2t-5} q^{n-3t+3},$$
$$= q^{n\left(1 - \frac{t}{n}[3\log_2 q - 2\log_2(q-1)] - \frac{1}{n}[5\log_2(q-1) - 3\log_2 q]\right)}$$

where $t = \lceil \log_2 n \rceil$.

The proof of the proposition is given at the end of this section, after describing the encoding procedure. We emphasize that we use $t = \lceil \log_2 n \rceil$ throughout this section (as opposed to $\lceil \log_2(n+1) \rceil$ used for binary VT encoding) because the binary auxiliary sequence has length $(n-1)$.

A. Encoding procedure

The high level idea for mapping a k-bit message to a codeword $C \in \mathcal{VT}_{a,b}(n)$ is as follows. As in the binary case, we reserve the t dyadic positions in the binary auxiliary sequence A_C to ensure that $\mathsf{syn}(A_C) = a$. Recall from (10) that each bit of A_C is determined by comparing two adjacent symbols of the q-ary sequence C. To ensure that $\mathsf{syn}(A_C) = a$, in addition to reserving the symbols in the dyadic positions of C, we also place some restrictions on the symbols adjacent to the dyadic positions. Finally, we use the first three symbols of C to ensure that $\mathsf{sum}(C) = b$. We explain the method in six steps with the help of the following running example.

Example 1. *Let $q = 8, n = 16$, and suppose that we wish to encode a binary message M to a codeword C in $\mathcal{VT}_{0,1}(16)$. Then $t = \lceil \log_2 n \rceil = 4$ and $\log_2 q = 3$. From (17) the length of M is $k = (n-3t+3)3 + (t-3)5 + 2 = 28$ bits. Let*

$$M = 110\,001\,000\,111\,010\,101\,000 \quad 11100 \quad 11, \quad (20)$$

where the spacing indicates the bits corresponding to the three terms in (17).

Step 1. Let $\mathcal{S}$ be the set of pairs of symbols adjacent to a dyadic symbol, i.e.,

$$\mathcal{S} = \{(c_{2^j-1}, c_{2^j+1}), \quad \text{for } 2 \leq j \leq (t-1)\}. \quad (21)$$

There are $|\mathcal{S}| = (t-2)$ pairs of symbols in $\mathcal{S}$. Excluding c_0, the number of symbols in C that are neither in dyadic positions nor in $\mathcal{S}$ is

$$(n-1) - 2|\mathcal{S}| - t = (n-3t+3). \quad (22)$$

Assign the first $(n-3t+3)\log_2 q$ bits of the message M to these symbols, by converting each set of $\log_2 q$ bits to a q-ary symbol. This corresponds to the first term in (17).

In Example 1, $(n-3t+3)\log_2 q = 21$, and the representation of first 21 bits of M in $\mathbb{Z}_8$ is 6 1 0 7 2 5 0. Therefore the sequence C is

$$C = c_0\ c_1\ c_2\ c_3\ c_4\ c_5\ 6\ c_7\ c_8\ c_9\ 1\ 0\ 7\ 2\ 5\ 0. \quad (23)$$

Step 2. In this step, we assign the remaining bits of the message to the symbols in $\mathcal{S}$. For a given dyadic position c_{2^j}, $j = 2, 3, \cdots, (t-1)$, we constrain the pair of adjacent symbols (c_{2^j-1}, c_{2^j+1}) to belong to the following set

$$\mathcal{T} = \{(r, l) \in \mathbb{Z}_q \times \mathbb{Z}_q : r \neq 0,\ l \neq (r-1)\}. \quad (24)$$

Via (24), we enforce $c_{2^j-1} \neq 0$ because if c_{2^j-1} were 0, then we necessarily have $c_{2^j} \geq c_{2^j-1}$ which constrains the value of α_{2^j} to 1 (recall from (10) that $\alpha_1 \ldots \alpha_{n-1}$ is the auxiliary sequence). However, α_{2^j} needs to be unconstrained in order to guarantee that any desired syndrome can be generated. Furthermore, we will see in Step 5 that if $c_{2^j+1} = c_{2^j-1} - 1$, then we may be unable to find a suitable symbol c_{2^j} compliant with the restrictions induced by the auxiliary sequence. We therefore enforce the constraint $c_{2^j+1} \neq c_{2^j-1} - 1$ using (24). It is easy to see that $|\mathcal{T}| = (q-1)^2$.

Excluding the pair (c_3, c_5), there are $(t-3)$ pairs in $\mathcal{S}$. If we were encoding q-ary message symbols, each of these $(t-3)$ pairs could take any pair of symbols in $\mathcal{T}$. Since we are encoding message bits, we use a look up table to map $\lfloor \log_2 |\mathcal{T}| \rfloor = 2\log_2 q - 1$ bits to each of the pairs in $\mathcal{S}$ excluding (c_3, c_5). We thus map $(t-3)(2\log_2 q - 1)$ bits to the pairs in $\mathcal{S}$ excluding (c_3, c_5). This corresponds to the second term in (17).

Next, set $c_3 = q-1$. This choice is important as it will facilitate step 6. Since $(c_3, c_5) \in \mathcal{T}$, when $c_3 = q-1$, then c_5 has to be such that $c_5 \neq q-2$. Hence, there are $q-1$ possible values for c_5. As we are encoding a binary message, we map $\lfloor \log_2(q-1) \rfloor = \log_2 q - 1$ bits to c_5 using a look-up table. This corresponds to the third term in (17). We note that the two look-up tables used in this step have sizes at most $(q-1)^2$ and q, respectively. Thus, in steps one and two in total we have mapped the claimed k message bits to the symbols of C.

In Example 1, as seen from (23), (c_7, c_9) is the only pair in $\mathcal{S}$ other than (c_3, c_5). We can assign $2\log_2 q - 1 = 5$ bits to (c_7, c_9). After the first 18 message bits mapped in Step 1, the next five bits in M are 11100. Suppose that in our look-up table these bits correspond to the pair $(3, 5)$. We then have

$(c_7, c_9) = (3, 5)$. Also, we fix $c_3 = q - 1 = 7$, and the last two message bits determine c_5. The last two message bits are 11. Suppose that 3 is the corresponding symbol in the look-up table. We therefore set $c_5 = 3$. Therefore, we have

$$C = c_0\ c_1\ c_2\ 7\ c_4\ 3\ 6\ 3\ c_8\ 5\ 1\ 0\ 7\ 2\ 5\ 0. \tag{25}$$

Up to this point, we have mapped our k message bits to a partially filled q-ary sequence. In the following steps we ensure that the resulting sequence lies in the correct VT code by carefully choosing remaining $(t+1)$ symbols to obtain the auxiliary sequence syndrome a and the modular sum b.

Step 3. In this step, we specify the bits in the non-dyadic locations of the auxiliary sequence A_C. Notice that according to (10), in order to define α_{2^j+1}, the value of c_{2^j} should be known. This is not the case here as the dyadic positions in C have been reserved to generate the required syndrome. To circumvent this issue, we determine α_{2^j+1} (for $1 < j < t$) by comparing c_{2^j+1} with c_{2^j-1} as follows:

$$\alpha_{2^j+1} = \begin{cases} 1 & \text{if } c_{2^j+1} \geq c_{2^j-1}, \\ 0 & \text{if } c_{2^j+1} < c_{2^j-1}. \end{cases} \tag{26}$$

As we shall show in step 5, we will be able to make these choices for the auxiliary sequence compatible with the definition of a valid auxiliary sequence in (10).

Next, since we have chosen $c_3 = q - 1$, from the rule in (10) we have $\alpha_3 = 1$, regardless of what c_2 is. The other non-dyadic positions of the auxiliary sequence A_C can be filled in using (10), i.e., $\alpha_i = 1$ if $c_i \geq c_{i-1}$, and 0 otherwise.

For our example with C shown in (25), at the end of this step we have

$$A_C = \alpha_1\ \alpha_2\ 1\ \alpha_4\ 0\ 1\ 0\ \alpha_8\ 1\ 0\ 0\ 1\ 0\ 1\ 0. \tag{27}$$

Step 4. In this step, we use the binary encoding method described in Section II to find the bits in the dyadic positions $\alpha_{2^0}, \cdots, \alpha_{2^{t-1}}$ such that $\mathsf{syn}(A_C) = a$. With this, the auxiliary sequence A_C is fully determined.

In the example, we need to find $\alpha_1, \alpha_2, \alpha_4$ and α_8 such that $\mathsf{syn}(A_C) = 0$. First, we set the syndrome bits $\alpha_1 = \alpha_2 = \alpha_4 = \alpha_8 = 0$, and denote the resulting sequence by

$$A'_C = 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0. \tag{28}$$

Now, $\mathsf{syn}(A'_C) = 12$, and the deficiency $d = 0 - 12 \,(\text{mod } 16) = 4$. The binary representation of d is $d_3 d_2 d_1 d_0 = 0100$. Hence, $\alpha_1 = \alpha_2 = \alpha_8 = 0$ and $\alpha_4 = 1$ will produce the desired syndrome. Summarizing, we have the following auxiliary sequence

$$A_C = 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0. \tag{29}$$

Step 5. In this step, we specify the symbols of C in the dyadic positions (except c_1 and c_2). This will be done by ensuring that A_C is a valid auxiliary sequence consistent with the definition in (10). In particular, the choice of c_{2^j} for $j = 2, \cdots, t-1$, should be consistent with α_{2^j+1} and α_{2^j}. We ensure this by choosing c_{2^j} (for $1 < j < t$) as follows:

$$c_{2^j} = \begin{cases} c_{2^j-1} - 1 & \text{if } \alpha_{2^j} = 0, \\ c_{2^j-1} & \text{if } \alpha_{2^j} = 1. \end{cases} \tag{30}$$

From the definition in (10), this choice is consistent with α_{2^j}. Now we show that it is also consistent with α_{2^j+1}. If $\alpha_{2^j+1} = 1$, then according to (26), $c_{2^j+1} \geq c_{2^j-1}$; then the choice of c_{2^j} in (30) always guarantees that $c_{2^j-1} \geq c_{2^j}$, and thus $c_{2^j+1} \geq c_{2^j}$. Next suppose that $\alpha_{2^j+1} = 0$. Then according to (26), $c_{2^j+1} < c_{2^j-1}$. We need to verify that $c_{2^j+1} < c_{2^j}$ Now, if $\alpha_{2^j} = 1$, then $c_{2^j} = c_{2^j-1}$ and $c_{2^j+1} < c_{2^j}$. Also, if $\alpha_{2^j} = 0$, from (30) we have $c_{2^j} = c_{2^j-1} - 1$. Since symbols adjacent to dyadic positions (c_{2^j-1}, c_{2^j+1}) are chosen from $\mathcal{T}$ (see step 2), then $c_{2^j+1} \neq c_{2^j-1} - 1$. Thus, we have that $c_{2^j+1} < c_{2^j-1} - 1 = c_{2^j}$. Therefore, in either case the choice is consistent with (10).

For the example, using (30) and (29) we obtain

$$C = c_0\ c_1\ c_2\ 7\ 7\ 3\ 6\ 3\ 2\ 5\ 1\ 0\ 7\ 2\ 5\ 0. \tag{31}$$

Step 6. Finally, we need to find c_0, c_1 and c_2 that are compatible with $\alpha_1, \alpha_2, \alpha_3$ (the first three bits of the auxiliary sequence), and such that $\mathsf{sum}(C) = b$. Let

$$w \triangleq b - \sum_{i=3}^{n} c_i \pmod q. \tag{32}$$

Hence we need $c_0 + c_1 + c_2 = w \pmod q$. We will show that when $q \geq 4$, we can find three distinct integers (x, y, z) such that $0 \leq x < y < z < q$ and $x + y + z = w \pmod q$. We will assign these numbers to c_0, c_1 and c_2. Also recall that we set $c_3 = q - 1$; hence we always have $x, y, z \leq c_3$, which is consistent with $\alpha_3 = 1$.

The triplet with smallest numbers that we can choose is $x = 0, y = 1, z = 2$. For this choice, $w = 0 + 1 + 2 = 3 \pmod q$. By increasing z from 2 to $q - 1$ with $x = 0$ and $y = 1$, we can produce any value of w from 3 to $q - 1$ as well as $w = 0$. Finally, the only remaining values are $w = 1, 2$. To obtain these values, we choose x, y, z as follows.

1) $\underline{w = 1}$: Choose $x = 0, y = 2, z = q - 1$.
2) $\underline{w = 2}$: Choose $x = 1, y = 2, z = q - 1$.

Hence, we have shown that for $q \geq 4$, any $w \in \mathbb{Z}_q$ can be expressed as the (mod q) sum of three distinct elements of $\mathbb{Z}_q$. Assigning these elements to c_0, c_1, c_2 in the order required by the auxiliary sequence completes the encoding procedure. We now have $\mathsf{sum}(C) = b$ and $\mathsf{syn}(A_C) = a$, and thus $C \in \mathcal{VT}_{a,b}(n)$ as required.

In our example, from (31) we have

$$\sum_{i=3}^{15} c_i = 48 = 0 \pmod 8, \tag{33}$$

and $b = 1$. Therefore we need $c_0 + c_1 + c_2 = 1 \pmod 8$. We have $\alpha_1 = \alpha_2 = 0$ so $c_0 > c_1 > c_2$ is the correct order. We therefore assign $c_0 = 7$, $c_1 = 2$, and $c_2 = 0$ to obtain the codeword.

$$C = 7\ 2\ 0\ 7\ 7\ 3\ 6\ 3\ 2\ 5\ 1\ 0\ 7\ 2\ 5\ 0 \in \mathcal{VT}_{0,1}(16). \tag{34}$$

It can be verified that $\mathsf{sum}(C) = 1$, and the auxiliary sequence syndrome $\mathsf{syn}(A_C) = 0$.

B. The case where q is not a power of two

When $\log_2 q$ is not an integer, the main difference is that we map longer sequences of bits to sequences of q-ary symbols. Recall that in step 1, we determine $(n-3t+3)$ symbols of the q-ary codeword. One can map $\lfloor (n-3t+3)\log_2 q \rfloor$ bits to these $(n-3t+3)$ symbols using standard methods to convert an integer expressed in base 2 into base q. In the second step, as described earlier we can map $\lfloor \log_2 (q-1)^2 \rfloor$ bits to $(t-3)$ pairs in $\mathcal{S}$ (excluding (c_3, c_5)). Moreover $\lfloor \log_2 (q-1) \rfloor$ bits can be mapped to c_5. Therefore, in total we can map k bits to a q-ary VT codeword of length n, where

$$k = \lfloor (n-3t+3)\log_2 q \rfloor \tag{35}$$

$$+ (t-3)\lfloor \log_2 (q-1)^2 \rfloor + \lfloor \log_2 (q-1) \rfloor \tag{36}$$

$$\geq n \log_2 q - t(\log_2 q + 2) - (2\log_2 q - 4). \tag{37}$$

For $q \geq 4$, the remaining steps are identical to the case where q is a power of two. The case of $q = 3$ is slightly different and it is discussed in Appendix A.

C. Proof of Proposition 1

The result can be directly derived from steps one and two of our encoding method by mapping sequences of q-ary message symbols (rather than sequences of message bits) to distinct codewords in $|\mathcal{VT}_{a,b}(n)|$. In step 1, we can assign $(n-3t+3)$ arbitrary symbols to positions that are neither dyadic nor in $\mathcal{S}$. There are q^{n-3t+3} ways to choose these symbols. Then in step two, we can choose $(q-1)^2$ pairs for each of the $(t-3)$ specified pairs of positions; furthermore, there are $(q-1)$ choices for c_5. According to steps 3 to 6, we can always choose the remaining symbols such that resulting codeword lies in $\mathcal{VT}_{a,b}(n)$. Therefore, we can map $q^{n-3t+3}(q-1)^{2t-5}$ different sequences of message symbols to distinct codewords in $\mathcal{VT}_{a,b}(n)$. This yields the lower bound on $|\mathcal{VT}_{a,b}(n)|$.

APPENDIX

A. Encoding for $q = 3$

For $q = 3$, we need to slightly modify the proposed algorithm. The first step is as described in Section III-B. The difference in the second step is that we do not embed data in c_5 and simply choose $c_5 = c_3 = 2$. Steps three to five remain the same. In the sixth step, we compute w as in (32), and choose c_0, c_1, c_2 as follows depending on the values of α_0 and α_1:

1) $\underline{\alpha_1 = \alpha_2 = 1}$: Choose $c_2 = c_1 = 2$ and $c_0 = w - 4 \pmod 3$.
2) $\underline{\alpha_1 = 1, \alpha_2 = 0}$: Choose $c_2 = 1$ and $c_1 = 2$ and $c_0 = w - 4 \pmod 3$.
3) $\underline{\alpha_1 = 0, \alpha_2 = 1}$: Choose $c_2 = 2$. If $w = 1$, then $c_1 = 0, c_0 = 2$. If $w = 0$, then $c_1 = 0, c_0 = 1$. If $w = 2$, then $c_1 = 1, c_0 = 2$.

The only remaining case is when $\alpha_1 = \alpha_2 = 0$. For this case, we need to change c_3 and c_4, and also the first three bits of A_C. Since c_3 has been set to 2, the first three bits of A_C in this case are 001. If we change these three bits to 110, $\mathsf{syn}(A_C)$ will remain unchanged. We therefore set $\alpha_1 = \alpha_2 = 1$ and $\alpha_3 = 0$. Now we update c_0, c_1, c_2, c_3 to be compatible with the new auxiliary sequence. Set $c_3 = 1$, recall that $c_5 = 2$ so we still have $c_5 \geq c_3$ and hence this change will not affect α_5. Update c_4 according to (30). Set $c_2 = c_1 = 2$, and $c_0 = w - 4 \pmod 3$. Now we have $c_3 < c_2$ which is consistent with $\alpha_3 = 0$. Also $c_2 \geq c_1 \geq c_0$ is consistent with $\alpha_1 = \alpha_2 = 1$.

Hence, for $q = 3$, we have mapped $k = \lfloor \log_2 3(n-3t+3) \rfloor + 2(t-3)$ bits to a q-ary codeword C. This induces following rate:

$$R = \frac{\lfloor \log_2 3 \ (n - 3\lceil \log_2 n \rceil + 3) \rfloor}{n} + \frac{2(\lceil \log_2 n \rceil - 3)}{n} \tag{38}$$

$$\geq \log_2 3 - \frac{2.76 \ \lceil \log_2 n \rceil}{n} - \frac{2.25}{n}. \tag{39}$$

Similarly to Proposition 2, we can show that for $q = 3$ there are at least $2^{2(t-3)}3^{n-3t+3}$ codewords in each of the VT codes.

ACKNOWLEDGEMENT

The authors thank Andreas Lenz for pointing them to the references [11] and [12].

REFERENCES

[1] N. J. A. Sloane, "On single-deletion-correcting codes," in *Codes and Designs, Ohio State University (Ray-Chaudhuri Festschrift)*, pp. 273–291, 2000. Online: https://arxiv.org/abs/math/0207197.

[2] E. A. Ratzer, "Marker codes for channels with insertions and deletions," *Ann. Telecommn.*, vol. 60, no. 1, pp. 29–44, 2005.

[3] K. A. Abdel-Ghaffar, F. Paluncic, H. C. Ferreira, and W. A. Clarke, "On Helberg's generalization of the Levenshtein code for multiple deletion/insertion error correction," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1804–1808, 2012.

[4] T. A. Le and H. D. Nguyen, "New multiple insertion/deletion correcting codes for non-binary alphabets.," *IEEE Trans. Inf. Theory*, vol. 62, no. 5, pp. 2682–2693, 2016.

[5] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," in *Proc. ACM-SIAM Symp. on Discrete Algorithms*, pp. 1884–1892, 2016.

[6] S. K. Hanna and S. E. Rouayheb, "Guess & check codes for deletions and synchronization," in *Proc. IEEE Int. Symp. Inf. Theory*, 2017.

[7] M. Abroshan, R. Venkataramanan, and A. Guillén i Fàbregas, "Multilayer codes for synchronization from deletions," in *Proc. IEEE Inf. Theory Workshop*, 2017.

[8] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors," *Automatica i Telemekhanica*, vol. 26, no. 2, pp. 288–292, 1965.

[9] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Doklady Akademii Nauk SSSR*, vol. 163, no. 4, pp. 845–848, 1965.

[10] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Inf. Theory*, vol. 30, no. 5, pp. 766–769, 1984.

[11] K. A. S. Abdel-Ghaffar and H. C. Ferreira, "Systematic encoding of the Varshamov-Tenengolts codes and the Constantin-Rao codes," *IEEE Trans Inf. Theory*, vol. 44, pp. 340–345, Jan 1998.

[12] K. Saowapa, H. Kaneko, and E. Fujiwara, "Systematic deletion/insertion error correcting codes with random error correction capability," in *Int. Symp. Defect and Fault Tolerance in VLSI Systems*, pp. 284–292, Nov. 1999.

[13] M. Abroshan, R. Venkataramanan, and A. Guillén i Fàbregas, "Coding for segmented edit channels," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3086–3098, 2017.

[14] B. D. Ginzburg, "A number-theoretic function with an application in the theory of coding," *Problemy Kibernet.*, pp. 249–252, 1967.

[15] A. A. Kulkarni and N. Kiyavash, "Nonasymptotic upper bounds for deletion correcting codes," *IEEE Trans Inf. Theory*, vol. 59, pp. 5115–5130, Aug. 2013.