Optimal Codes for the q-ary Deletion Channel

Jin Sima[1], Ryan Gabrys[2] and Jehoshua Bruck[1]

[1]Department of Electrical Engineering, California Institute of Technology
[2]Department of Electrical and Computer Engineering, University of California San Diego

***Abstract*—The problem of constructing optimal multiple deletion correcting codes has long been open until recent breakthrough for binary cases. Yet comparatively less progress was made in the non-binary counterpart, with the only rate one non-binary deletion codes being Tenengolts' construction that corrects single deletion. In this paper, we present several q-ary t-deletion correcting codes of length n that achieve optimal redundancy up to a factor of a constant, based on the value of the alphabet size q. For small q, our constructions have $O(n^{2t}q^t)$ encoding/decoding complexity. For large q, we take a different approach and the construction has polynomial time complexity.**

I. Introduction

In the last few years, considerable progress has been made on the problem of coding for the binary deletion channel. In [1], Brakensiek et al. constructed t-deletion correcting codes of length n that require $O(t^2 \log t \log n)$ bits of redundancy, which was a dramatic improvement over existing coding schemes [9]. Several works quickly followed [1] that further improved upon this result. For the case where $t = 2$, [6] and [16] constructed two deletion correcting codes that require $8 \log n$ and $7 \log n$ bits of redundancy, respectively. For general t, Haeupler [7] gives an explicit systematic construction which requires $\theta(t\frac{\log^2 \frac{n}{t}}{\log q} + t)$ bits of redundancy. In [3], another construction was derived by Cheng et al., which is not systematic, but is order optimal in the sense that it requires $O(t \log n)$ bits of redundancy. An improved result in terms of redundancy was presented in [14], in which a non-systematic code that requires $8t \log n$ bits of redundancy was constructed.

Despite this recent progress, the problem of constructing codes for the q-ary deletion channel has received significantly less attention. Tenengolts constructed a nearly optimal code for the case of a single deletion [17]. The main idea in [17] is to use a parity code to identify the symbol which was deleted and an associated Levenshtein code to determine the location of the deletion. For the case of multiple deletions, the Helberg codes [9], which were originally proposed for the binary deletion channel, were adapted and shown to produce non-binary deletion correcting codes [10]. The primary drawback to this class of codes is their low rate [10]. Even for the case of two deletions the codes have rates that do not approach 1 as n becomes large. It was shown in [12] that the optimal redundancy of a q-ary t deletion code asymptotically falls between $t \log n + t \log q + o(\log qn)$ and $2t \log n + t \log q + o(\log qn)$.

This work was supported in part by NSF grants CCF-1816965 and CCF-1717884.

In this work, we attempt to bring the existing results for non-binary codes closer to the results obtained in the binary domain. We highlight the main contributions of this work through the following two theorems.

Theorem 1. *Let t be a constant with respect to k and suppose that $q < k$. Then, there exists a non-systematic, efficiently encodable/decodable t-deletion code of message length k over an alphabet of size q that requires at most $2t(1+\epsilon)(2\log n + \log q) + o(\log n)$ bits of redundancy.*

Theorem 2. *Let t be a constant with respect to k and suppose that $q \geqslant k$. Then, there exists a non-systematic, efficiently encodable/decodable t-deletion code of message length k over an alphabet of size q that requires at most $(30t+1)\log q$ bits of redundancy.*

As will be explained in more detail in Section III, the result stated in the second theorem also extends to the case where t is a constant fraction of code length n, when q is large enough. A similar case when t is a fraction of n and q is a polynomial of n was solved in [8].

We make use of two different approaches to obtain the advertised results. For the results stated in the first theorem, we rely on the *syndrome compression* technique, which is introduced in our companion paper [15]. For the results in the second theorem, we make use of results from repeat-free sequences from [4]. To the best of the authors' knowledge, the best known constructions of non-binary codes for the deletion channel can be found in [10] and so our results represent a significant improvement over existing work.

This paper is organized as follows. In Section II, we present our constructions for the case where $q \leqslant k$, which make use of the syndrome compression technique. Section III uses repeat-free sequences to construct codes when $q \geqslant k$. Finally Section IV concludes the paper.

II. q-ary Codes Correcting t Deletions for Small q

In this section we present t-deletion correcting codes for q-ary alphabets where q is less than the message length k. In particular, we consider the following two cases: (1) $q \leqslant \log k$. (2) $\log k < q \leqslant k$. The redundancy of the resulting q-ary t-deletion codes is $2t(1+\epsilon)(2\log k + \log q) + o(\log k)$ bits. Before describing the code constructions, let us introduce a few notations for this section. For a sequence $\mathbf{u} \in [[q]]^m \triangleq \{0,\ldots,q-1\}^m$ of length m over the alphabet $\{0,\ldots,q-1\}$, let $\mathcal{B}_t^q(\mathbf{u})$ be its deletion ball with radius t, consisting of all length m sequences obtained by deleting t q-ary symbols and inserting t q-ary symbols in $\mathbf{u}$. The following result comes from a simple counting argument.

Claim 3. *For any t and $\mathbf{u} \in [[q]]^m$, we have $|\mathcal{B}_t^q(\mathbf{u})| \leqslant m^{2t} q^t$.*

Define a binary matrix representation U for $\mathbf{u} \in [[q]]^m$ as

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,m} \\ \vdots & \ddots & \vdots \\ u_{\lceil \log q \rceil,1} & u_{\lceil \log q \rceil,2} & \cdots & u_{\lceil \log q \rceil,m} \end{bmatrix} \in \{0,1\}^{\lceil \log q \rceil \times m}, \tag{1}$$

where the i-th symbol of $\mathbf{u}$ is given by the i-th column of U for $i \in [m] \triangleq \{1, \ldots, m\}$. Let U_i^r, $i \in [[\log q]]$ and U_j^c, $j \in [m]$ be the i-th row and j-th column of U respectively. Then the deletion of the j-th symbol of $\mathbf{u}$ corresponds to the deletion of the column U_j^c in the matrix U.

A. Case (1) : $q \leqslant \log k$

In the following, we describe t-deletion correcting codes for the case where $q \leqslant \log k$. The basic idea, which will be described in more details that follow, is to interpret our non-binary sequences as a set of $\lceil \log q \rceil$ sequences over the binary alphabet as illustrated in (1). We will then use a compound labeling which is defined using the labeling from binary deletion codes [1] on each of these binary sequences to form a code that can correct t deletions. To further reduce the size of the compound labeling, we adapt a technique called syndrome compression [15]. Define

$\mathcal{M}(m,t) = \{\mathbf{x} \in \{0,1\}^m$: For integers

$\ell = \lceil \log t + \log\log(t+1) + 5 \rceil$ and $d = O(t(\log t)^2 \log m)$

and for any string $\boldsymbol{p} \in \{0,1\}^\ell$, every substring of consecutive d bits in $\mathbf{x}$ contains $\boldsymbol{p}$ as a substring.$\}$

The following results will be used.

Lemma 4. *(c.f., [1]) Fix an integer $t \geqslant 2$. Then for all large m, there exists $R(m) = \mathcal{O}(t^2 \log t \log m)$ and a hash function $f_t : \mathcal{M}(m,t) \to \{0,1\}^{R(m)}$ so that for any distinct $\mathbf{x}, \mathbf{y} \in \mathcal{M}(m,t)$, we have $f_t(\mathbf{x}) \neq f_t(\mathbf{y})$, if $\mathbf{y} \in \mathcal{B}_t^2(\mathbf{x}) \backslash \{\mathbf{x}\}$.*

Lemma 5. *(c.f., [15]) Let $f : \{0,1\}^m \to [[2^{o((\log\log m \cdot \log m))}]]$ be a labeling function such that for any fixed $\mathbf{x} \in \{0,1\}^m$ and any $\mathbf{y} \in \mathcal{B}_t^q(\mathbf{x})$, we have that $f(\mathbf{x}) \neq f(\mathbf{y})$. Then there exists an integer $a \leqslant 2^{\log|\mathcal{B}(\mathbf{x})| + o(\log m)}$ such that for any $\mathbf{y} \in \mathcal{B}_t^q(\mathbf{x}) \backslash \{\mathbf{u}\}$, we have that $f(\mathbf{x}) \not\equiv f(\mathbf{y}) \bmod a$.*

We begin by describing the labeling. According to Lemma 4, there exists a labeling function $f_t(\mathbf{u}) : \{0,1\}^m \to \{0,1\}^{O(t^2 \log t \log m)}$ such that $f_t(\mathbf{u}) \neq f_t(\mathbf{y})$ for any $\mathbf{u} \in \mathcal{M}(m,t)$ and $\mathbf{y} \in (\mathcal{B}_t(\mathbf{u}) \backslash \{\mathbf{u}\}) \cap \mathcal{M}(m,t)$. Define the set

$$\mathcal{M}(m,t,q) := \left\{ \mathbf{u} \in \{0,1\}^m : U_i^r \in \mathcal{M}(m,t), \forall i \in [[\log q]] \right\}.$$

and the labeling function

$$f_t^q(\mathbf{u}) = \left(f_t(U_1^r), \ldots, f_t(U_{\lceil \log q \rceil}^r) \right) \in \{0,1\}^{R_q}.$$

Then, from definition of $f_t(\mathbf{u})$, we have that $f_t^q(\mathbf{u}) \neq f_t^q(\mathbf{y})$ for $\mathbf{u} \in \mathcal{M}(m,t,q)$ and $\mathbf{y} \in \mathcal{M}(m,t,q) \cap (\mathcal{B}_t^q(\mathbf{u}) \backslash \{\mathbf{u}\})$. The size of $f_t^q(\mathbf{u})$ is $R_q = O(t^2 \log t \log q \log m)$ bits, so we cannot immediately apply Lemma 5. To resolve this issue, we apply the syndrome compression technique in [15] on $f_t^q(\mathbf{u})$ and obtain a new labeling $f_t^{q_1}$ in the next lemma.

Lemma 6. *There exists a labeling $f_t^{q_1} : [[q]]^m \to [[2^{o((\log\log m \cdot \log m))}]]$ and $f_t^{q_1}$ is such that for any $\mathbf{u} \in [[q]]^m$ and $\mathbf{y} \in (\mathcal{B}_t^q(\mathbf{u}) \backslash \{\mathbf{u}\}) \cap \mathcal{M}(m,t,q)$, we have $f_t^{q_1}(\mathbf{u}) \neq f_t^{q_1}(\mathbf{y})$.*

Since the labeling $f_t^{q_1}$ works for sequences $\mathbf{u}$ in $\mathcal{M}(m,t,q)$, we need to encode the information $\mathbf{u} \in \mathbb{F}_q^m$ to a sequence in $\mathcal{M}(m,t,q)$. The next lemma, which can be proved following similar arguments in [3], shows that the sequences in $\mathcal{M}(m,t,q)$ can be generated using an $O(\log m)$ bit seed s.

Lemma 7. *For any $\mathbf{u} \in [[q]]^m$, there exists a seed $s \in \{0,1\}^{O(\log m)}$ and a function $T_1^q(\mathbf{u},s)$, computable in poly(m,t) time, such that $T_1^q(\mathbf{u},s) \in \mathcal{M}(m,t,q)$.*

We are now ready to present our code construction in terms of the encoding process. Let $\mathcal{E}_t$ be the encoder that takes a q-ary information sequence $\mathbf{c} \in [[q]]^m$ as input and outputs a q-ary codeword $\mathcal{E}_t(\mathbf{c})$ such that the i-th row of the matrix representation of $\mathcal{E}_t(\mathbf{c})$ is $f_t(C_i^r)$, where f_t is given in Lemma 4.

1) Let $\mathbf{u} \in \{0,1\}^k$ and suppose s is such that $\mathbf{u}_T = T_1^q(\mathbf{u},s) \in \mathcal{M}(k,t,q)$.
2) Suppose $a \in [[2^R]]$ is such that $f_t^{q_1}(\mathbf{u}_T) \not\equiv f_t^{q_1}(\mathbf{y}) \bmod a$ for any $\mathbf{y} \in \mathcal{B}_t^q(\mathbf{u}_T) \cap \mathcal{M}(k,t,q)$ where $R = 2t \log k + o(\log k)$.
3) Then,

$$\mathbf{x} = \left(\mathbf{u}_T, \mathcal{E}_t\left(s, a, f_t^{q_1}(\mathbf{u}_T) \bmod a\right) \right) \in [[q]]^n.$$

In the resulting codeword $\mathbf{x}$ above, we assume that s, a are represented using q-ary symbols and similarly for the vector $\mathcal{E}_t\left(s, a, f_t^{q_1}(\mathbf{u}_T) \bmod a\right)$.

Since $f_t^{q_1}(\mathbf{u}_T) \in [[2^{o((\log\log k) \cdot \log k)}]]$, it follows from Lemma 5 that $(a, f_t^{q_1}(\mathbf{u}_T) \bmod a)$ can be described using at most $4t \log k + O(\log k)$ bits. Since the size of s is $O(\log k)$ bits, the redundancy $\mathcal{E}_t\left(s, a, f_t^{q_1}(\mathbf{u}_T) \bmod a\right)$ can be represented by at most $4t \log k + \mathcal{O}(\log k) + \mathcal{O}(t^2 \log q \log t \log(4t \log k))$ bits, which is at most $4t(1+\epsilon) \log k$ bits.

Theorem 8. *Let $\mathbf{z}$ be the result of at most t deletions occurring to $\mathbf{x}$. Then, we can uniquely recover $\mathbf{x}$ from $\mathbf{z}$.*

Proof: Let $\mathbf{z}$ be a length $n-t$ subsequence of $\mathbf{x}$. Then $(z_{k+1}, \ldots, z_{n-t})$ is a length $n-t-k$ subsequence of $(x_{k+1}, \ldots, x_n)$. Since $(x_{k+1}, \ldots, x_n) = \mathcal{E}_t\left(s, a, f_t^{q_1}(\mathbf{u}_T) \bmod a\right)$ is a codeword from a t-deletion correcting code, we can recover $\left(s, a, f_t^{q_1}(\mathbf{u}_T) \bmod a\right)$ from $(z_{k+1}, \ldots, z_{n-t})$. Since $T_1^q(\mathbf{u},s) \in \mathcal{M}(k,t,q)$, we have that $f_t^{q_1}(\mathbf{y}) \not\equiv f_t^{q_1}(T_1^q(\mathbf{u},s)) \bmod a$ for $\mathbf{y} \in \mathcal{M}(k,t,q)$ and $\mathbf{y} \in \mathcal{B}_t^q(T_1^q(\mathbf{u},s))$. Therefore, the sequence $T_1^q(\mathbf{u},s)$ can be recovered. Finally, given $T_1^q(\mathbf{u},s)$ and s, we can recover $\mathbf{u}$, and from $\mathbf{u}$ we can recover $\mathbf{x}$. ■

B. Case $(2) : \log k < q \leqslant k$

We now present a t-deletion code for the case $\log k < q < k$, which is more involved than the case $q \leqslant \log k$. There are two key ideas in constructing the code. The first is to narrow down the ranges of deletion locations to blocks of length $O(poly(t)\log k)$, thus recovering most of the blocks in the sequence. To further recover the remaining blocks, the second idea decomposes a q-ary representation of a sequence down to its symbol histogram information, which counts the frequency of the symbols, and the its permutation information, which records the index of each symbol. The second idea has a similar flavor to the construction of q-ary single deletion correcting codes in [17], where the symbol histogram and ascending/descending order information are used.

To achieve the first part, we generate binary sequences that satisfy a period constraint, similar to [2]. A sequence $\mathbf{u} \in \{0,1\}^m$ has period p if $\mathbf{u}_i = \mathbf{u}_{i+p}$ for $i \in [m-p]$. Let $L(\mathbf{u}, p)$ be the length of the longest subsequence of consecutive bits in $\mathbf{u}$ that has period p. Denote

$$\mathcal{L}(m,t) = \left\{\mathbf{u} : L(\mathbf{u},p) \leqslant 2\log m + t + 1, \forall p \in [t]\right\}$$

to be the set of sequences whose subsequences of any period $p \in [t]$ has length at most $2\log m + t + 1$. The following lemma provides a randomized algorithm to generate sequences in $\mathcal{L}(m,t)$.

Lemma 9. *Let U_ℓ be a random string uniformly distributed over $\{0,1\}^\ell$ where $\ell = 2\log m + t + 1$. Let $g_1(U_\ell) \in \{0,1\}^m$ be the sequence obtained by repeating U_ℓ and taking the first m bits. Then for any sequence $\mathbf{u} \in \{0,1\}^m$, the bitwise XOR $g_1(U_\ell) + \mathbf{u} \in \mathcal{L}(m,t)$ with probability at least $1 - 1/m$.*

Lemma 7 and Lemma 9 imply the following lemma.

Lemma 10. *Let U_ℓ be a random string uniformly distributed over $\{0,1\}^\ell$, where $\ell = O(\log m)$. Then there exists a map $g_2 : \{0,1\}^\ell \rightarrow \{0,1\}^m$ such that for every string $\mathbf{u} \in \{0,1\}^m$, we have that $\mathbf{u} + g_2(U_\ell) \in \mathcal{M}(m,t) \cap \mathcal{L}(m,t)$ with probability at least $1 - 1/m - 1/poly(m)$.*

Lemma 10 implies that for any sequence $\mathbf{u} \in \{0,1\}^m$, it is possible to search in $poly(m)$ time for a seed s that can be described in $\mathcal{O}(\log m)$ bits such that $g_2(s) + \mathbf{u} \in \mathcal{M}(m,t) \cap \mathcal{L}(m,t)$. For a sequence $\mathbf{u} \in \{0,1\}^m$ and integers $\{\delta_1, \ldots, \delta_t\}$, where $1 \leqslant \delta_1 < \delta_2 < \ldots < \delta_t \leqslant m$, let $\mathbf{u}(\delta_1, \ldots, \delta_t)$ denote the length $m - t$ subsequence obtained by deleting bits u_{δ_i}, $i \in [t]$. The next lemma shows that given $\mathbf{u} \in \mathcal{L}(m,t)$ and $\mathbf{u}(\delta_1, \ldots, \delta_t)$, it is possible to narrow down the range of δ_i, $i \in [t]$.

Lemma 11. *If a sequence $\mathbf{u} \in \mathcal{L}(m,t)$, then given $\mathbf{u}$ and $\mathbf{u}(\delta_1, \ldots, \delta_t)$, we can find at most $t+2$ disjoint intervals $[a_i, b_i] \subset [m]$ for $i \in [t+2]$, with length $T \triangleq (2t+1)(2\log m + t + 2) + t$ each, such that $\{\delta_1, \ldots, \delta_t\} \subset \cup_{i \in [1,t+2]}[a_i, b_i]$. In addition, for any $j \in [m] \backslash (\cup_{i \in [[t+2]]}[a_i, b_i])$, the number of deletions $N_j = |[j-1] \cap \{\delta_1, \ldots, \delta_t\}|$ that occur in interval $[j-1]$ can be determined.*

Recall the matrix representation U of the q-ary codeword $\mathbf{u}$. In light of Lemma 11, we protect the first row U_1^r from t deletions and use it to determine the ranges where the deletions occur. Since the deletion ranges have short length, most of the symbols in sequence $\mathbf{u} \in [[q]]^m$ can be recovered. To this end, the first row U_1^r is generated such that $U_1^r \in \mathcal{L}(m,t) \cap \mathcal{M}(m,t)$. Define the set

$$\mathcal{M}^q(m,t) = \{\mathbf{u} : \mathbf{u} \in [[q]]^m, U_1^r \in \mathcal{M}(m,t) \cap \mathcal{L}(m,t)\}.$$

The following Lemma generates sequences in $\mathcal{M}^q(m,t)$ and can be proved using similar arguments that prove Lemma 7.

Lemma 12. *For any $\mathbf{u} \in [[q]]^m$, there exists a seed s of $\mathcal{O}(\log m)$ bits and a function $T_2^q(\mathbf{u}, s) : [[q]]^m \times \{0,1\}^{O(\log m)} \rightarrow [[q]]^m$, computable in $poly(m,t)$ time, such that $T_2^q(\mathbf{u}, s) \in \mathcal{M}^q(m,t)$.*

Let $\mathbf{u} \in \mathcal{M}^q(m,t)$ be a sequence and $\boldsymbol{z} = \mathbf{u}(\delta_1, \ldots, \delta_t)$ be the length $m-t$ subsequence of $\mathbf{u}$ after deleting the δ_i-th symbol, $i \in [t]$. Since $U_1^r \in \mathcal{M}(m,t)$, we can protect U_1^r against t deletions and recover it by using the code in Lemma 4. Then given U_1^r and its $m-t$ subsequence Z_1^r, it is possible from Lemma 11 to find $t+2$ intervals $[a_i, b_i]$, $i \in [t+2]$ each having length $T = (2t+1)(2\log m + t + 2) + t$, that contain all deletion locations. Split $\mathbf{u}$ into blocks $\mathbf{u}_i = (u_{(i-1)T+1}, \ldots, u_{iT})$, $i \in [[m/T]]$, of length T. Then the interval $[a_i, b_i]$, $i \in [t+2]$, covers at most two blocks in $\mathbf{u}$. Note that the symbol $\mathbf{u}_j$ for $j \in [m] \backslash (\cup_{i=1}^{t+2}[a_i, b_i])$ can be determined by

$$\mathbf{u}_j = \mathbf{z}_{j-N_j}, \tag{2}$$

where N_j is obtained from Lemma 11. Hence there are at most $2t+4$ block errors in $\mathbf{u}$ after recovering U_1^r.

Next, we show how to correct the block errors. The idea is to represent each block using its symbol frequency and the symbol location. Specifically, for a q-ary sequence $\mathbf{u} \in [[q]]^m$, define its histogram vector $H(\mathbf{u}) : [[q]]^m \rightarrow [[m+1]]^q$ by

$$H(\mathbf{u})_i = |\{j : \mathbf{u}_j = i, j \in [[m]]\}|, i \in [[q]], \tag{3}$$

where the i-th entry of $H(\mathbf{u})$ is the number of occurrence of $i \in [[q]]$ in $\mathbf{u}$. Its location vector $V(\mathbf{u}) : [[q]]^m \rightarrow [m]^m$ is defined by

$$V(\mathbf{u})_i = \text{the index of the } i\text{-th largest symbol in } \mathbf{u}, \tag{4}$$

where a symbol u_i is larger than u_j, if u_i is lexicographically larger than u_j or if $u_i = u_j$ and $i > j$. Note that by definition, we have that $u_{V(\mathbf{u})_1} > u_{V(\mathbf{u})_2} > \ldots > u_{V(\mathbf{u})_m}$. The following lemma shows that a sequence $\mathbf{u} \in [[q]]^m$ is uniquely determined by its histogram and the location vectors.

Lemma 13. *Let $\mathbf{u}, \mathbf{y} \in \mathbb{F}_q^m$ be two sequences. If $H(\mathbf{u}) = H(\mathbf{y})$ and $V(\mathbf{u}) = V(\mathbf{y})$, then $\mathbf{u} = \mathbf{y}$*

Next we show how to protect the histogram and location vectors of $\mathbf{u}$ from block errors. Let the block histogram vector $BH(\mathbf{u}) : [[q]]^m \rightarrow \left([[T+1]]^q\right)^{\lceil m/T \rceil}$ of the sequence $\mathbf{u} \in [[q]]^m$ be given by

$$BH(\mathbf{u})_i = H(\mathbf{u}_i), i \in [[m/T]], \tag{5}$$

where the i-th entry of $BH(\mathbf{u})$ is the histogram vector of the i-th block in $\mathbf{u}$, $i \in [[m/T]]$. Similarly, define the block location vector $BV(\mathbf{u}) : [[q]]^m \rightarrow \left([T]^T\right)^{\lceil m/T \rceil}$ by

$$BV(\mathbf{u})_i = V(\mathbf{u}_i), i \in [[m/T]], \tag{6}$$

where the i-th entry $BV(\mathbf{u})_i$ is the location vector of the i-th block in $\mathbf{u}$. According to Lemma 13, the sequence $\mathbf{u}$ can be uniquely determined by $BH(\mathbf{u})$ and $BV(\mathbf{u})$.

Define the function $f^{BH}(\mathbf{u}) = (RS_{2t+4}(RS_{2t}(BH(\mathbf{u})_1), \ldots, RS_{2t}(BH(\mathbf{u})_{\lceil \frac{m}{T} \rceil}))$, which is the redundancy of a systematic Reed-Solomon code correcting $2t+4$ erasure errors that has length $\lceil \frac{m}{T} \rceil$ sequence with entries $RS_{2t}(BH(\mathbf{u})_i)$, $i \in [[m/T]]$. Each entry $RS_{2t}(BH(\mathbf{u})_i)$ represents the redundancy of a systematic Reed Solomon code correcting t substitution errors in the length q sequence $H(\mathbf{u}_i)$ for $i \in [[m/T]]$. The next lemma shows that $f^{BH}(\mathbf{u})$ can be used to protect $BH(\mathbf{u})$. In the following, the function f_t is from Lemma 4.

Lemma 14. *Let $\mathbf{u}, \mathbf{y} \in \mathcal{M}^q(m,t)$ be two sequences such that $\mathbf{y} \in \mathcal{B}_t^q(\mathbf{u})$. If $f_t(U_1^r) = f_t(Y_1^r)$ and $f^{BH}(\mathbf{u}) = f^{BH}(\mathbf{y})$, then $BH(\mathbf{u}) = BH(\mathbf{y})$.*

We now protect the block location vector $BV(\mathbf{u})$. Let

$$f^{BV}(\mathbf{u}) = RS_{2t+4}(BV(\mathbf{u})) \tag{7}$$

be the redundancy of a systematic Reed-Solomon code correcting $2t+4$ erasure errors in the length $\lceil m/T \rceil$ sequence $BV(\mathbf{u})$ with entries $BV(\mathbf{u})_i = V(\mathbf{u}_i)$ for $i \in [[m/T]]$ (see Eq. (6)). The next lemma shows that $f^{BV}(\mathbf{u})$ can be used to recover $BV(\mathbf{u})$.

Lemma 15. *For sequences $\mathbf{u}, \mathbf{y} \in \mathcal{M}^q(n,t)$ such that $\mathbf{y} \in \mathcal{B}_t^q(\mathbf{u})$, if $f_t(U_1^r) = f_t(Y_1^r)$ and $f^{BV}(\mathbf{u}) = f^{BV}(\mathbf{y})$, then $BV(\mathbf{u}) = BV(\mathbf{y})$.*

Now we are ready to define the labeling function $f_t^{q_2}(\mathbf{u})$ for $\mathbf{u} \in \mathcal{M}^q(m,t)$. Let

$$f_t^{q_2}(\mathbf{u}) = (f_t(U_1^r), f^{BH}(\mathbf{u}), f^{BV}(\mathbf{u})).$$

Then from Lemma 4, Lemma 11, Lemma 13, Lemma 14, and Lemma 15, we have the following lemma.

Lemma 16. *For two sequences $\mathbf{u}, \mathbf{y} \in \mathcal{M}^q(m,t)$, if $\mathbf{y} \in \mathcal{B}_t^q(\mathbf{u}) \backslash \mathbf{u}$, then $f_t^{q_2}(\mathbf{u}) \neq f_t^{q_2}(\mathbf{y})$.*

The image of the labeling function $f_t^{q_2}(\mathbf{u})$ consists of R^q bits where $R^q = O((2t+4)\log\log m * \log m)$, Since the size of $f_t^{q_2}(\mathbf{u})$ is greater than $o((\log\log m) \cdot \log m)$, we apply the syndrome compression technique twice, as we did in Lemma 6 in Case (1). Then there exists a labeling function $f^{q_3} : [[q]]^m \rightarrow [[2^{o((\log\log m)\cdot\log m)}]]$ such that $f_t^{q_3}(\mathbf{u}) \neq f_t^{q_3}(\mathbf{y})$ for $\mathbf{u}, \mathbf{y} \in \mathcal{M}^q(m,t)$ and $\mathbf{y} \in \mathcal{B}_{t,q}(\mathbf{u}) \backslash \mathbf{u}$. Since $f_t^{q_3}(\mathbf{u}) \in [[2^{o((\log\log m)\cdot\log m)}]]$, we use Lemma 5 and find an integer $\alpha \leqslant 2^{|\mathcal{B}_t^q(\mathbf{u})|+o(\log m)} = 2^{2\log n+\log q+o(\log m)}$ such that $f_t^{q_3}(\mathbf{u}) \not\equiv f_t^{q_3}(\mathbf{y}) \bmod \alpha$ for $\mathbf{u}, \mathbf{y} \in \mathcal{M}^q(m,t)$ and $\mathbf{y} \in \mathcal{B}_t^q(\mathbf{u}) \backslash \mathbf{u}$.

Define the code $\mathcal{C}(n,t,q)$ of length n as follows:

$$\mathcal{C}(n,t,q) = \Big\{\mathbf{x} = \Big(T_2^q(\mathbf{u},s), \mathcal{E}_t\big(f_t^{q_3}(T_2^q(\mathbf{u},s)) \bmod \alpha, \alpha, s\big)\Big) : \mathbf{u} \in \{0,1\}^k\Big\}.$$

Since $f_t^{q_3}(T_2^q(\mathbf{u},s))$ satisfies the redundancy property, it follows from syndrome compression that the redundancy $(f_t^{q_3}(T_2^q(\mathbf{u},s)) \bmod \alpha, \alpha)$ can be described by $2\log|\mathcal{B}_t^q(\mathbf{u})| + o(\log k) = 4t\log k + 2t\log q + o(\log n)$ bits. The seed s has length $O(\log k)$. Therefore, the total redundancy is $4t(1+\epsilon)\log k + 2t\log q + o(\log k)$ bits. The correctness of the code can be proved by similar argument to the one in the proof of Theorem 8.

Theorem 17. *The code $\mathcal{C}(n,t,q)$ is a t-deletion code.*

III. q-ary Codes Correcting t Deletions for Large q

In this section, we consider the problem of coding for deletions over large non-binary alphabets. Specifically, it is assumed that $q > k$ in this section. We will show that in this regime we can construct efficiently encodable/decodable codes capable of correcting t deletions that requires roughly $30t\log q$ bits of redundancy. The approach taken in this section is fundamentally different than the syndrome compression technique that has been used up to this point. Note that, compared to the syndrome compression technique, the redundancy of our code is high. However, the advantage of the approach discussed here is that our methods are more applicable to a wider range of t. In particular, the technique described here, which is similar in spirit to the approach taken in [18] to correct errors in permutations, has decoding/encoding complexity which scales polynomially for any t and, in addition, leads to efficiently encodable/decodable codes for the regime where t is a small constant fraction of n.

We will construct codes by making use of the L-spectrum, which represents the set of all length L subsequences of consecutive symbols that appear in a vector. For a vector $\mathbf{u} \in [[q]]^m$ (where $q > m$), we denote the L-spectrum for $\mathbf{u}$, denoted $S_L(\mathbf{u})$ as follows:

$$S_L(\mathbf{u}) = \Big\{(u_i, u_{i+1}, \ldots, u_{i+L-1}) \in [[q]]^L : i \in [m-L+1]\Big\}.$$

Furthermore, we say that a sequence $\mathbf{u} \in [[q]]^m$ is L-substring unique if $|S_L(\mathbf{u})| = m - L + 1$. The approach taken to correcting deletions is motivated by the following two lemmas, the first of which also appears in [5].

Lemma 18. *(c.f., [5]) Suppose $\mathbf{u} \in [[q]]^m$ is $(L-1)$-substring unique. Then, $\mathbf{u}$ can be uniquely recovered from $S_L(\mathbf{u})$.*

For shorthand, for two sets A, B let $A \triangle B = (A \backslash B) \cup (B \backslash A)$ denote their symmetric difference.

Lemma 19. *Suppose $\mathbf{u} \in [[q]]^m$ is $(L-1)$-substring unique and $\mathbf{z} \in [[q]]^{m-t}$ is the result of t deletions occurring to $\mathbf{u}$. Then, $|S_L(\mathbf{u}) \triangle S_L(\mathbf{z})| \leqslant (2L-1)t$.*

In light of the previous two lemmas, our approach will consist of two basic steps:

1) **Transform step**: In this step, we convert our information vectors into vectors which are L-substring unique.
2) **Coding step**: We add additional redundancy symbols to our codewords to ensure that we can recover their L-spectrum provided deletion errors are allowed to occur.

We begin by describing some results that are related to the transform step. Define $\mathcal{U}_L^q(m)$ so that

$$\mathcal{U}_L^q(m) = \Big\{ \mathbf{u} \in [[q]]^m : \mathbf{u} \text{ is } L\text{-substring unique} \Big\}.$$

The following result provides an algorithm to generate binary sequences in $\mathcal{U}_L^q(m)$ for $L = 2\log m + 2$. This algorithm will be used in Lemma 21 to generate non-binary sequences that L-substring unique.

Lemma 20. *[4], There exists an invertible function $h_L : \{0,1\}^{m-1} \to \{0,1\}^m$, computable in $poly(m)$ time, that takes any binary sequence $\mathbf{u} \in \{0,1\}^{m-1}$ as input and outputs a sequence $h_L(\mathbf{u}) \in \mathcal{U}_L^2(m)$ for $L = 2\log m + 2$.*

Lemma 20 can be used to generate sequences in $\mathcal{U}_3^q(m)$.

Lemma 21. *There exists an invertible function $h_L : [[q]]^m \to [[q]]^{m+1}$, computable in $poly(m)$ time, such that $h_L(\mathbf{u}) \in \mathcal{U}_L^q(m+1)$ for any $\mathbf{u} \in [[q]]^m$ where $L = 3$.*

Now, we turn to describing the coding step (step 2)) of our construction. We will interpret the 4-spectrum of our codewords using indicator vectors. In particular, define the 4-profile indicator vector $\mathbf{1}_4^q(u) \in \{0,1\}^{q^4}$, which is indexed by the non-zero elements in $[[q]]^4$, by

$$\mathbf{1}_4^q(u)_{\mathbf{p}} = \begin{cases} 1 & \text{if } \mathbf{p} \in S_4(\mathbf{u}), \\ 0 & \text{else.} \end{cases}$$

for $\mathbf{p} \in [[q^4]]$. Note that the indices of the 1 entries in $\mathbf{1}_4^q(\mathbf{u})$ correspond to the 4-spectrum of $\mathbf{u}$.

An immediate consequence of Lemma 19 is the following.

Corollary 22. *Suppose $\mathbf{u} \in [[q]]^m$ is 3-substring unique and $\mathbf{z} \in [[q]]^{m-t}$ is the result of t deletions occurring to $\mathbf{u}$. Then, we have that $d_H(\mathbf{1}_4^q(\mathbf{u}), \mathbf{1}_4^q(\mathbf{z})) \leqslant 7t$, where d_H denotes the Hamming distance.*

For a vector $\mathbf{v} \in \{0,1\}^{q^4}$, let $\text{BCH}_{7t}(\mathbf{v})$ denote the $28t\log q$ redundant bits from a systematic BCH code of dimension q^4 that is capable of correcting $7t$ substitution errors. The idea now is to encode these $28t\log q$ bits of information (which will be used to protect the indicator vectors for our codewords), into the final $29t+1$ symbols of our codewords, while reserving a portion of each symbol to store location information. Let $\text{BCH}_{7t}(\mathbf{1}_4^q(\mathbf{u}))$

Let $\mathbf{r}_I = (r_1, r_2, \ldots, r_{29t}) \in [[\frac{q}{30t}]]^{29t}$ be the output of $\text{BCH}_{7t}(\mathbf{1}_4^q(\mathbf{u}))$ represented as $\frac{q}{30t}$-ary symbols. Clearly, we can represent $29t\log\frac{q}{30t}$ bits of information with $\mathbf{r}_I$. Note that this encoding is possible since $29t\log\frac{q}{30t} \geqslant 28t\log q$, which holds when $\log q \geqslant 29\log(30t)$.

Let $RS_{\lfloor\frac{t+1}{2}\rfloor}$ be a Reed-Solomon code over $[[\frac{q}{30t}]]$ that can correct either $\lfloor\frac{t+1}{2}\rfloor$ substitution errors or t erasures and the code has minimum distance at least $t+1$. We assume $RS_{\lfloor\frac{t+1}{2}\rfloor}$ has dimension $29t$ and that for a vector $\mathbf{v} \in [[\frac{q}{30t}]]$ $RS_{\lfloor\frac{t+1}{2}\rfloor}(\mathbf{v})$ outputs t redundant symbols. Let

$$\mathbf{r} = \left(\mathbf{r}_I, RS_{\lfloor\frac{t+1}{2}\rfloor}(\mathbf{r}_I)\right) \in [[\frac{q}{30t}]]^{30t}.$$

We are now ready to present the t-deletion code in terms of the encoding process. Suppose $\mathbf{u} \in [[q]]^k$ is an information vector of dimension k. The output of the encoding process will be a vector $\mathbf{x} \in [[q]]^n$ where $n = k + 1 + 30t$.

1) Suppose $\mathbf{u}_T = h_3(\mathbf{u}) \in [[q]]^{k+1}$ where h_3 is defined in Lemma 21.
2) Let $\mathbf{r}_I = (r_1, r_2, \ldots, r_{29t}) \in [[\frac{q}{30t}]]^{29t}$ denote the $\frac{q}{30t}$-ary representation of $\text{BCH}_{7t}(\mathbf{1}_4^q(\mathbf{u}_T)) \in \{0,1\}^{28t\log q}$.
3) Let $\mathbf{r} = \left(\mathbf{r}_I, RS_{\lfloor\frac{t+1}{2}\rfloor}(\mathbf{r}_I)\right) = (R_1, R_2, \ldots, R_{30t}) \in [[\frac{q}{30t}]]^{30t}$.
4) Define $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ so that

$$x_i = \begin{cases} (\mathbf{u}_T)_i & \text{if } i \in [k+1], \\ \Big(i-(k+1), (\mathbf{r})_{i-(k+1)}\Big) & \text{if } i \in [n]\backslash[k+1] \end{cases}$$

Theorem 23. *Suppose $\mathbf{x} \in [[q]]^n$ is transmitted and $\mathbf{z} \in [[q]]^{n-t}$ is received where $\mathbf{z}$ is the result of t deletions occurring to $\mathbf{x}$. Then given $\mathbf{z}$ it is possible to uniquely determine $\mathbf{x}$.*

IV. Conclusion

In this work, we constructed q-ary codes capable of correcting deletions that significantly improves upon the current state of the art. However, many important problems remain. In particular, when $q \leqslant n$, our technique is only applicable to the setup where t is a constant with respect to n since the encoding/decoding complexity becomes exponential otherwise. In addition, none of the constructions presented here are systematic.

References

[1] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1884-1892, SIAM, 2016.

[2] Y. Chee, H. Kiah, A.Vardy, V. Vu and E. Yaakobi "Coding for racetrack memories," *IEEE Trans. Inform. Theory*, vol. 64, no. 11, pp. 7094-7112, 2018.

[3] K. Cheng, Z. Jin, X. Li and K. Wu, "Deterministic document exchange protocols, and almost optimal binary codes for edit errors," *IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 200–211, 2018.

[4] O. Elishco, R. Gabrys, E. Yaakobi, and M. Medard, "Repeat-free codes," available at *arXiv:1909.05694*, 2019.

[5] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded strings from multiset substring spectra," *IEEE Transactions on Information Theory*, vol. 65, no. 12, pp. 7682-7696, 2019.

[6] R. Gabrys and F. Sala, "Codes correcting two deletions," *IEEE Transactions on Information Theory*, vol. 65, no. 2, 2019.

[7] B. Haeupler, "Optimal document exchange and new codes for small number of insertions and deletions," *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 334–347, 2019.

[8] B. Haeupler and A. Shahrasbi,"Synchronization strings: codes for insertions and deletions approaching the singleton bound." *ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 33–46. 2017.

[9] 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, 2002.

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

[11] V.I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet Physics-Doklady*, vol. 10, no. 8, pp. 707-710, 1966.

[12] V.I. Levenshtein, “Bounds for deletion/insertion correcting codes,” in *Proc. IEEE Int. Symp. Inf. Theory*, Lausanne, Switzerland, 2002.

[13] 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.

[14] J. Sima and J. Bruck, “Optimal k-deletion correcting codes,” in *Proc. IEEE Int. Symp. Inf. Theory*, Paris, France, 2019.

[15] J. Sima, R. Gabrys, and J. Bruck, “Syndrome Compression for Optimal Redundancy Codes,” *Proc. ISIT*, 2020.

[16] J. Sima, N. Raviv, and J. Bruck, “Two deletion correcting codes from indicator vectors,” in *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2375-2391, 2020.

[17] G. Tenengolts, “Nonbinary codes, correcting single deletion or insertion,” *IEEE Trans. Inform. Theory*, vol. 30, no. 5, pp. 766-769, 1984.

[18] S. Yang, C. Schoeny, and L. Dolecek, “Theoretical bounds and constructions of codes in the generalized cayley metric,” in *IEEE Transactions on Information Theory*, vol. 65, no. 8, 2019.