Mutually Uncorrelated Codes for DNA Storage

Maya Levy and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*—*Mutually uncorrelated* (*MU*) codes are a class of codes in which no proper prefix of one codeword is a suffix of another codeword. These codes were originally studied for synchronization purposes and recently, Yazdi *et al.* showed their applicability to enable random access in DNA storage. In this paper,we follow the research of Yazdi *et al.* and study MU codes along with their extensions to correct errors and balanced codes. We first review a well-known construction of MU codes and study the asymptotic behavior of its cardinality. This task is accomplished by studying a special class of run-length limited codes that impose the longest run of zeros to be at most some function of the codewords length. We also present an efficient algorithm for this class of constrained codes and show how to use this analysis for MU codes. Next, we extend the results on the run-length limited codes in order to study (d_h, d_m)-MU codes that impose a minimum Hamming distance of d_h between different codewords and d_m between prefixes and suffixes. In particular, we show an efficient construction of these codes with nearly optimal redundancy. We also provide similar results for the edit distance and balanced MU codes. Last, we draw connections to the problems of comma-free and prefix synchronized codes.**

***Index Terms*—DNA storage, mutually uncorrelated codes, constrained codes, non-overlapping codes, cross-bifix-free codes, comma-free codes.**

I. Introduction

Mutually Uncorrelated (*MU*) codes satisfy the constraint in which the prefixes set and suffixes set of all codewords are disjoint. This class of codes was first studied by Levenshtein [20] for the purpose of synchronization, and has received attention recently due to its relevance and applicability for DNA storage [35]. Namely, these codes offer random access of DNA blocks in synthetic DNA storage.

The potential of DNA molecules as a volume for storing data was recognized due to its unique qualities of density and durability. The first large scale DNA storage system was designed by Church *et al.* [11] in 2012. Since then a few similar systems were implemented for archival applications as they did not support random access to the memory. Recently, in [8], [35] two random access DNA storage systems were proposed. To enable random access, the authors suggested to equip the DNA information blocks with unique addresses, also known as *primers*. Under this setup, the reading process starts with a phase aimed to identify the requested DNA block. During the identification process the complementary sequence of the unique primer is sent to the DNA pool and stitches to the primer part of the requested DNA block. By that, the selection of the DNA block is done and only the selected block is read.

Obtaining a *good* set of primers is therefore a key property in achieving the desired random access feature as it guarantees the success of the chemical processes involved in the identification phase. The constraints that a good primers set should satisfy are listed by Yazdi *et al.* in [35]. In this paper we focus on three of the four listed constraints, described as follows: 1) the MU constraint is imposed as it avoids overlaps in two primers, which are likely to cause erroneous identification of DNA blocks, 2) both the writing and reading channels of DNA introduce substitution errors, therefore we are interested in large mutual Hamming distance, and 3) we require the primers to be balanced since balanced DNA sequences increase the chances of successful reads. In addition, Yazdi *et al.* [33] mentioned that deletion errors are introduced during synthesis. We therefore also extend our study to MU codes with edit distance.

MU codes were rigorously studied in the literature under different names such as *codes without overlaps* [23], [24], *non-overlapping codes* [7] and *cross-bifix-free codes* [2], [6], [10]. However, the basic problem of finding the largest MU code is still not fully solved. Let us define by $A_{MU}(n, q)$ the size of the largest MU code over a field of size q. The best upper bound, found by Levenshtein [23], states that $A_{MU}(n, q) < q^n/(e(n-1))$, while the best known constructive lower bound, for the case of a fixed q, was given independently in [13], [20], and it states that $A_{MU}(n, q) \gtrsim \frac{q-1}{qe} \cdot \frac{q^n}{n}$. Extensions were studied in [7] to provide better constructions when q is large. The construction of MU codes in [13], [20] is explicit. Given some $k < n$, one fixes the first k symbols to be zero, followed by a single non-zero symbol, and the last symbol is non-zero as well. The sequence of the remaining $n-k-2$ symbols needs to satisfy the constraint that it does not have a zero run of length k. Previous results claimed that $\frac{q-1}{qe} \cdot \frac{q^n}{n}$ is a lower bound on the construction's code size, when $n = (q^i - 1)/(q - 1)$, however it was not known whether it is possible to achieve codes with larger cardinality. We give an explicit expression of the asymptotic cardinality of this construction for any value of n and show that the lower bound $\frac{q-1}{qe} \cdot \frac{q^n}{n}$ is indeed tight. This result is accomplished by studying a special family of run-length limited constrained codes, called *k-run length limited* (*RLL*) *codes*, which impose the longest run of zeros to be of size at most $k-1$, where k is a function of the word's length. We also present an efficient

Manuscript received March 7, 2018; revised August 27, 2018; accepted September 9, 2018. Date of publication October 1, 2018; date of current version May 20, 2019. This work was supported in part by the Israel Science Foundation under Grant 1624/14. This paper was presented in part at the IEEE International Symposium on Information Theory [25].

The authors are with the Department of Computer Science, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: mayalevy@cs.technion.ac.il; yaakobi@cs.technion.ac.il).

Communicated by C. Xing, Associate Editor for Coding Theory.
Digital Object Identifier 10.1109/TIT.2018.2873138

encoding and decoding algorithm for k-RLL codes with linear complexity which results in efficient binary MU codes with $\lceil\log(n)\rceil + 4$ redundancy bits.

Next, we extend the study of k-RLL codes to the *window weight limited constraint* that imposes the Hamming weight of every length-k subsequence to be at least some prescribed number d. Accordingly, we study MU codes with error-correction capabilities. A (d_h, d_m)-*MU code* is an MU code with minimum Hamming distance d_h, with the additional property that every prefix of length i differs by at least $\min\{i, d_m\}$ symbols from all proper length suffixes. We show an upper bound on the size of (d_h, d_m)-MU codes and give a construction of such codes with encoder and decoder of linear complexity. The redundancy of the construction is $\left\lfloor \frac{d_h+1}{2} \right\rfloor \log(n) + (d_m - 1)\log\log n + \mathcal{O}(d_m \log d_m)$, which is nearly optimal with respect to our upper bound on the code size. A similar constraint is imposed when studying MU codes with edit distance. We give a general result of such codes and study MU codes that can correct a single deletion or insertion. For the latter case we use a systematic encoder for the Varshamov Tenengolts codes [31]. Lastly, we study balanced MU codes, that is, codes which are both MU and balanced. We show that the achievable minimum redundancy of these codes is approximately $1.5\log(n)$ and for an efficient construction we use Knuth's algorithm while the redundancy is roughly $2\log(n)$ bits.

The rest of this paper is organized as follows. In Section II, we formally define the codes studied in the paper and review related work. In Section III, we analyze the redundancy of k-RLL codes, when k is a function of the word's length, and propose efficient encoding and decoding algorithms for these codes. We use this analysis in Section IV in order to study the asymptotic cardinality of the MU codes. We extend the results on k-RLL codes in Section V to study the window weight limited constraint and accordingly in Section VI, we extend the class of MU codes to (d_h, d_m)-MU codes. We continue in Section VII to study MU codes that can correct deletions and insertions, and in Section VIII we study balanced MU codes. Furthermore, in Section IX we draw connections to the problems of comma-free and prefix synchronized codes. Lastly, Section X concludes and summarizes the results in the paper.

II. Definitions, Preliminaries, and Related Work

For every two integers $i \le k$ we denote by $[i, k]$ the set of integers $\{j \mid i \le j \le k\}$ and use $[k]$ as a shortening to $[1, k]$. We use the notation $\Sigma = \{0, 1\}$ as the binary alphabet and $\Sigma_q = \{0, 1, \ldots, q-1\}$ as the notation for larger alphabets. For two integers n, k, the notation $\langle n\rangle_k$ stands for n modulo k. For a vector $\mathbf{a} = (a_1, \ldots, a_n)$ and $i, j \in [n], i \le j$, we denote by $\mathbf{a}_i^j$ the subvector $(a_i, \ldots, a_j)$ of $\mathbf{a}$. For $j < i$, $\mathbf{a}_i^j$ is the empty word. The Hamming weight of a vector $\mathbf{a}$ is denoted by $w_H(\mathbf{a})$ and $d_H(\mathbf{a}, \mathbf{b})$ is the Hamming distance between $\mathbf{a}$ and $\mathbf{b}$. A *zero run* of length r of a vector $\mathbf{a}$ is a subsequence $\mathbf{a}_i^{i+r-1}$, $i \in [n-r+1]$ such that $a_i = \cdots = a_{i+r-1} = 0$. The notation $\mathbf{a}^i$ denotes the concatenation of the vector $\mathbf{a}$ a total of i times and $\mathbf{ab}$ is the concatenation of the two vectors $\mathbf{a}$ and $\mathbf{b}$. Let A, B be two sets of vectors over Σ_q. We write $AB = \{ab | a \in A, b \in B\}$ and $A^i = AA\cdots A$ to be i concatenations of the set A. For two functions $f(n), g(n)$ we say that $f(n) \lesssim g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} \le 1$, and $f(n) \approx g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$. The *redundancy* of a set $A \subseteq \Sigma_q^n$ is defined as $\mathrm{red}(A) = n - \log_q |A|$. If the base of the logarithm is omitted then it is assumed to be 2. Throughout the paper our asymptotic approach when we use notations such as $\Theta, \mathcal{O}, \omega$, assumes that the codewords length, denoted by n, tends to infinity and parameters like alphabet size, q and code's distances d, d_m, d_h, d_e are all fixed.

*Definition 1: Two not necessarily distinct words $\boldsymbol{a}, \boldsymbol{b} \in \Sigma_q^n$ are **mutually uncorrelated** if any non-trivial prefix of $\boldsymbol{a}$ does not match any non-trivial suffix of $\boldsymbol{b}$. A code $\mathcal{C} \subseteq \Sigma_q^n$ is a **mutually uncorrelated (MU) code** if any two not necessarily distinct codewords of $\mathcal{C}$ are mutually uncorrelated.*

For example, the words 00010 and 11001 are mutually uncorrelated because no prefix of one is a suffix of the other. However, the code $\{00010, 11001\}$ is not a mutually uncorrelated code because the word 00010 has a prefix 0 and a suffix 0, hence it does not satisfy the MU property with itself. On the other hand, the code $\{001111, 001101, 001011\}$ is an MU code because all prefixes of length one are 0 while suffixes of length one are 1. Moreover, non trivial prefixes of length greater than 1 include a zero run of length two while non trivial suffixes of these lengths do not.

Let us denote by $A_{MU}(n, q)$ the largest cardinality of an MU code of length n over Σ_q. Levenshtein showed in [23] that for all n and q

$$A_{MU}(n, q) \le \left(\frac{n-1}{n}\right)^{n-1} \frac{q^n}{n} < \frac{q^n}{e(n-1)}, \tag{1}$$

or in other words, the redundancy is lower bounded by $\log_q e + \log_q(n-1)$.

We now recall a well studied family of MU codes. It was suggested independently first by Gilbert in [13], later by Levenshtein in [20].

Construction I: Let n, k be two integers such that $1 \le k < n$ and let $\mathcal{C}_1(n, q, k) \subseteq \Sigma_q^n$ be the following code:

$$\mathcal{C}_1(n, q, k) = \{\boldsymbol{a} \in \Sigma_q^n \mid \forall i \in [k], a_i = 0, a_{k+1} \ne 0, a_n \ne 0, \boldsymbol{a}_{k+2}^{n-1} \text{ has no zero run of length } k\}.$$

We denote

$$\mathcal{C}_1(n, q) = \max_{1\le k<n} \{|\mathcal{C}_1(n, q, k)|\}.$$

It was proved in [13], [20] that there exists an appropriate choice of k for which the following lower bounds hold

$$\mathcal{C}_1(n, q) \gtrsim q^{-\frac{q}{q-1}} \ln q \frac{q^n}{n}, \tag{2}$$

as $n \to \infty$, and more specifically

$$\mathcal{C}_1(n, q) \gtrsim \frac{q-1}{qe} \cdot \frac{q^n}{n}, \tag{3}$$

where $n \to \infty$ over the subsequence $\frac{q^i-1}{q-1}, i \in \mathbb{N}$. However, it was not established in these works what the asymptotic cardinality of $\mathcal{C}_1(n, q)$ is and for which k it is achieved.

We answer these questions in Section IV and show that the asymptotic inequalities in (2), (3) are asymptotically tight.

Additional interesting approach for constructing binary MU codes was proposed in [6] and was later generalized in [2] to alphabets of size $q > 2$. In this case, it was shown that for some small values of n the approach in [2] achieves larger cardinality than Construction I. Both works [2], [6] do not offer asymptotic analysis of the constructions' redundancy, however, for the binary case it is commonly believed that Construction I provides the best known asymptotic code size. Furthermore, Blackburn [7] extended Construction I for $q > 2$ and showed that if q is a multiple of n then this extension results with strictly optimal codes. Lastly, we note that in [4], [5] Gray codes were presented for listing the vectors of the code $\mathcal{C}_1(n, q, k)$.

Due to the structure of Construction I, the tools that we use for analyzing its cardinality are taken from studies in the field of constraint coding. We recall the well known Run Length Limited (RLL) constraint in which the lengths of the zero runs are limited to a fixed range of values.

*Definition 2: Let k and n be two integers. We say that a vector $\boldsymbol{a} \in \Sigma_q^n$ satisfies the k-**run length limited (RLL)** constraint, and is called a k-**RLL vector**, if $n < k$ or $\forall i \in [n-k+1] : w_H(\boldsymbol{a}_i^{i+k-1}) \geq 1$.*

In other words, a vector is called a k-RLL vector if it does not contain a zero run of length k. We denote by $A_q(n, k)$ the set of all k-RLL length-n vectors over the alphabet Σ_q, and $a_q(n, k) = |A_q(n, k)|$.

The definition of the k-RLL constraint is similar to the (d, k)-RLL constraint from [26] that states that any zero run is of length at least d and at most k. In other words, the k-RLL constraint is equivalent to the well studied $(0, k-1)$-RLL constraint. We chose the notation of k-RLL for simplicity of the following analysis to come.

The *capacity* of the k-RLL constraint, is defined for any fixed values of k and q as

$$E_{k,q} = \lim_{n\to\infty} \frac{\log\left(a_q(n, k)\right)}{n}. \tag{4}$$

Kato and Zeger [16] provided the following result for the binary case and later Jain *et al.* [15] generalized it for $q \geq 2$

$$E_{k,q} = \log q - \frac{(q-1)\log e}{q^{k+2}}(1 + o(1)).$$

III. Run Length Limited Constraint

The k-RLL constraint has been studied extensively in the literature; see [26] and references therein. However, this study was focused solely for the case in which k is fixed, meaning k is independent of n. As will be explained later, the MU codes problem requires analysis of values of k which depend on n. In Subsection III-A, we resort to previous results on the k-RLL constraint, when k is fixed to provide a better understanding of the asymptotic behavior of $a_q(n, k)$ when k depends on n. Later in Subsection III-B we present efficient encoding and decoding algorithms to avoid zero runs of length $\lceil \log n \rceil + 1$, with linear time and space complexity, and with only a single bit of redundancy.

A. Cardinality Analysis

We start by giving general bounds on $a_q(n, k)$ for most values of n, k in Lemma 3 and Lemma 5. The intuition of the proofs is as follows. In Lemma 3, we consider the set of all length-n vectors which are a concatenation of multiple k-RLL vectors each of length $2k$, i.e. each length-$2k$ vector satisfies the k-RLL constraint. The set $A_q(n, k)$ is a subset of this set and hence $a_q(n, k)$ is upper bounded by its size. For the lower bound we consider the set of all vectors of length n which are again, a concatenation of multiple shorter k-RLL vectors, however, now we remove vectors that start or end with $\lceil k/2 \rceil$ zeros. The new set is a subset of $A_q(n, k)$ and a lower bound is derived accordingly.

Lemma 3: Let n, k be positive integers such that $5 \leq k \leq n$, then

$$q^{n - p_q(n,k)} \leq a_q(n, k) \leq q^{n - c_3 \frac{n-2k}{q^k}},$$

where

$$p_q(n, k) = \log_q\left(\frac{q}{q-1}\right) c_2 \frac{n}{q^{k-1}} + \frac{\log_q(e) c_2}{c_1} \frac{n}{q^{1.5k-1}},$$

$$c_1 = \frac{(q-1)q^{\lceil k/2 \rceil - k/2}}{2q}, \quad c_2 = \frac{\left\lfloor \frac{n}{q^{k-1}} \right\rfloor + 1}{\frac{n}{q^{k-1}}},$$

$$c_3 = \frac{\log_q e(q-1)^2}{2q^2}.$$

Specifically, assuming q is fixed and $n \to \infty$, if $k \leq \log_q n + \mathcal{O}(1)$ then c_1, c_2 are all bounded by positive constants and $p_q(n, k) = \Theta(\frac{n}{q^k})$.

Proof:

Part 1 (Upper Bound):

We consider the set $A_q(2k, k)^{\lfloor \frac{n}{2k} \rfloor}$, that is, the set of vectors which are a concatenation of $\lfloor \frac{n}{2k} \rfloor$ vectors from $A_q(2k, k)$. We then append it with the set of all length-$\langle n \rangle_{2k}$ q-ary vectors. The resulting set of length-n vectors is denoted by

$$B_q(n, k) = A_q(2k, k)^{\lfloor \frac{n}{2k} \rfloor} \Sigma_q^{\langle n \rangle_{2k}}.$$

Note that $A_q(n, k) \subseteq B_q(n, k)$ and $|B_q(n, k)| = a_q(2k, k)^{\lfloor \frac{n}{2k} \rfloor} q^{\langle n \rangle_{2k}}$, hence

$$a_q(n, k) \leq a_q(2k, k)^{\lfloor \frac{n}{2k} \rfloor} q^{\langle n \rangle_{2k}}. \tag{5}$$

Let $b(k)$ be the number of vectors of length $2k$ with a exactly one zero run of length exactly k and with no zero run of length greater than k. There are $q^{2k-(k+1)}(q-1)$ vectors of length $2k$ that start with a zero run of length exactly k and $q^{2k-(k+1)}(q-1)$ different vectors that end with such a run. There are $2k - (k-1) - 2 = k - 1$ other positions in which a zero run of length k can start within the $2k$ vector. For each such a position there are $q^{2k-(k+2)}(q-1)^2$ different vectors with a zero run of length exactly k. In total we have

$$\begin{aligned} b(k) &= 2\, q^{2k-(k+1)}(q-1) + (k-1)q^{2k-(k+2)}(q-1)^2 \\ &= 2\, q^{k-1}(q-1) + (k-1)q^{k-2}(q-1)^2 \\ &\geq (k+1)q^{k-2}(q-1)^2. \end{aligned}$$

Note that all vectors in the count contain a zero run of length k with non-zero symbols on both its sides. There are only $k-1$

or $k-2$ remaining symbols in the vector, hence it does not contain another zero run of length k and therefore, no vector was counted twice. All length-$2k$ vectors with a zero run of length exactly k are not included in $A_q(2k,k)$. Therefore,

$$a_q(2k,k) \le q^{2k} - b(k) \le q^{2k} - (k+1)q^{k-2}(q-1)^2. \quad (6)$$

By combining inequalities (5) and (6), we get

$$\begin{aligned} a_q(n,k) &\le (q^{2k} - (k+1)q^{k-2}(q-1)^2)^{\lfloor \frac{n}{2k} \rfloor} q^{\langle n \rangle_{2k}} \\ &= \left(q^{2k}\left(1 - \frac{(k+1)(q-1)^2}{q^{k+2}}\right)\right)^{\lfloor \frac{n}{2k} \rfloor} q^{\langle n \rangle_{2k}} \\ &= q^n \left(1 - \frac{(k+1)(q-1)^2}{q^{k+2}}\right)^{\lfloor \frac{n}{2k} \rfloor} \\ &\overset{(a)}{\le} q^n \left(e^{-\frac{(k+1)(q-1)^2}{q^{k+2}}}\right)^{\lfloor \frac{n}{2k} \rfloor} \\ &\le q^{n - \log_q e \frac{(k+1)(q-1)^2}{q^{k+2}}(\frac{n}{2k} - 1)} \\ &\le q^{n - \log_q e \frac{(q-1)^2(n-2k)}{2q^{k+2}}}, \end{aligned}$$

where (a) results from the inequality $1 - x \le e^{-x}$ for all x.

Part 2 (Lower Bound):

We start by giving an upper bound on the number of length-n vectors with a zero run of length at least k. There are $n-k+1$ positions in which a zero run can start and for each position we have at most q^{n-k} different vectors. From the union bound we have that the number of length-n vectors with a zero run of length at least k is upper bounded by $q^{n-k}(n-k+1)$, and by excluding those vectors from the set of all length-n vectors we get

$$a_q(n,k) \ge q^n - q^{n-k}(n-k+1).$$

This bound is irrelevant for values of k smaller than $\log_q n$ as it is less than zero. However, if we choose $k \ge \log_q n + 1$ it becomes useful and we get

$$\begin{aligned} a_q(n,k) &\ge q^n - q^{n - \log_q n - 1}(n - \log_q n) \\ &= q^n \left(1 - \frac{n - \log_q n}{qn}\right) \\ &\ge q^n \left(1 - \frac{n}{qn}\right) \\ &= q^{n-1}(q-1). \end{aligned}$$

Since we are also interested in the case of $k < \log_q n + 1$, or equivalently $n > q^{k-1}$, we continue our analysis by breaking down the length-n vector to blocks of length q^{k-1} and applying the bound on them in the following manner.

For $\ell \ge k$ we denote the set

$$\begin{aligned} C_q(\ell,k) = \{\mathbf{a} \mid \ \mathbf{a} \in A_q(\ell,k),\ \mathbf{a} &\text{ starts, or ends, or both} \\ &\text{starts and ends with at least } \lceil k/2 \rceil \text{ zeros}\} \end{aligned}$$

and $c_q(\ell,k) = |C_q(\ell,k)|$. Note that

$$c_q(\ell,k) \le 2 \cdot q^{\ell - \lceil k/2 \rceil}.$$

We also denote $D_q(\ell,k) = A_q(\ell,k) \setminus C_q(\ell,k)$ and

$$\begin{aligned} d_q(\ell,k) = |D_q(\ell,k)| &= a_q(\ell,k) - c_q(\ell,k) \\ &\ge a_q(\ell,k) - 2q^{\ell - \lceil k/2 \rceil}. \end{aligned}$$

For $k \ge \log_q \ell + 1$, or equivalently $\ell \le q^{k-1}$ we have that

$$\begin{aligned} d_q(\ell,k) &\ge q^{\ell-1}(q-1) - 2q^{\ell - \lceil k/2 \rceil} \\ &= q^{\ell-1}(q-1)\left(1 - \frac{2q}{(q-1)q^{\lceil k/2 \rceil}}\right). \quad (7) \end{aligned}$$

Consider the set of vectors which are a concatenation of $\lfloor \frac{n}{q^{k-1}} \rfloor$ vectors from $D_q(q^{k-1},k)$ appended by a vector from $D_q(\langle n \rangle_{q^{k-1}},k)$. We denote this set by

$$E_q(n,k) = D_q(q^{k-1},k)^{\lfloor \frac{n}{q^{k-1}} \rfloor} D_q(\langle n \rangle_{q^{k-1}},k).$$

Note that $E_q(n,k) \subseteq A_q(n,k)$ and

$$|E_q(n,k)| = d_q(q^{k-1},k)^{\lfloor \frac{n}{q^{k-1}} \rfloor} d_q(\langle n \rangle_{2^{k-1}},k).$$

Hence,

$$\begin{aligned} a_q(n,k) &\ge d_q(q^{k-1},k)^{\lfloor \frac{n}{q^{k-1}} \rfloor} d_q(\langle n \rangle_{q^{k-1}},k) \\ &\overset{Eq.(7)}{\ge} \left(q^{q^{k-1}-1}(q-1)\left(1 - \frac{2q}{(q-1)q^{\lceil k/2 \rceil}}\right)\right)^{\lfloor \frac{n}{q^{k-1}} \rfloor} \\ &\quad \cdot q^{\langle n \rangle_{q^{k-1}}-1}(q-1)\left(1 - \frac{2q}{(q-1)q^{\lceil k/2 \rceil}}\right) \\ &= q^n \left(\frac{q-1}{q}\right)^{\lfloor \frac{n}{q^{k-1}} \rfloor + 1} \left(1 - \frac{2q}{(q-1)q^{\lceil k/2 \rceil}}\right)^{\lfloor \frac{n}{q^{k-1}} \rfloor + 1}. \end{aligned}$$

For all $x < -1$, it is known that $(1 + \frac{1}{x})^{x+1} < e$. We denote $x = -\frac{(q-1)q^{\lceil k/2 \rceil}}{2q} = -c_1 q^{\frac{k}{2}}$, for some $c_1 > 0$. For $k \ge 5$, $q \ge 2$ we have that $x < -1$ and we deduce that

$$\begin{aligned} \left(1 - \frac{2q}{(q-1)\,q^{\lceil k/2 \rceil}}\right)^{\lfloor \frac{n}{q^{k-1}} \rfloor + 1} &= \left(1 + \frac{1}{x}\right)^{(x+1)\left(\lfloor \frac{n}{q^{k-1}} \rfloor + 1\right)/(x+1)} \\ &> e^{\left(\lfloor \frac{n}{q^{k-1}} \rfloor + 1\right)/(x+1)} \\ &\overset{(a)}{=} e^{\left(c_2 \frac{n}{q^{k-1}}\right)/\left(-c_1 q^{\frac{k}{2}}\right)} \\ &= e^{-\frac{c_2}{c_1} \frac{n}{q^{1.5k-1}}}, \end{aligned}$$

where (a) follows from a choice of $c_2 > 0$ which satisfies $c_2 \frac{n}{q^{k-1}} = \lfloor \frac{n}{q^{k-1}} \rfloor + 1$. Finally, we conclude that

$$a_q(n,k) \ge q^n \left(\frac{q-1}{q}\right)^{c_2 \frac{n}{q^{k-1}}} e^{-\frac{c_2}{c_1} \frac{n}{q^{1.5k-1}}}$$

and the bound follows by rearranging. ■

From Lemma 3 we have that

$$\begin{aligned} c_3 \frac{n-2k}{q^k} &\le \operatorname{red}(A_q(n,k)) \\ &\le \log_q \left(\frac{q}{q-1}\right) c_2 \frac{n}{q^{k-1}} + \frac{\log_q(e) c_2}{c_1} \frac{n}{q^{1.5k-1}}. \end{aligned}$$

Recall that we refer to q as a fixed parameter and $n \to \infty$. When $k \le \log_q n + \mathcal{O}(1)$, we get that c_1, c_2, c_3 are all bounded by positive constants and the lower and upper bounds on the redundancy are of the same order. Namely, there exist constants $C_1, C_2 > 0$ such that for large enough n

$$C_1\frac{n}{q^k} \le \text{red}(A_q(n,k)) \le C_2\frac{n}{q^k},$$

and the next corollary follows.

Corollary 4: Let $f(n)$ be a function such that $\log_q n - f(n) \le \log_q n + \mathcal{O}(1)$ and $\log_q n - f(n)$ is a positive integer. Then, the redundancy of $A_q(n, \log_q n - f(n))$ is $\Theta(q^{f(n)})$.

Note that $f(n)$ can be negative. The result from Corollary 4 provides us with a general understanding on how the redundancy of the set $A_q(n,k)$ behaves for different values of k. We can conclude for example that for $k = 0.5\log_q n$ the redundancy is $\Theta(\sqrt{n})$. Motivated by the MU problem, we are interested in further exploring the case in which the redundancy is $\mathcal{O}(1)$. According to the result from Lemma 3 and Corollary 4, this holds when $f(n)$ is a $\mathcal{O}(1)$ function. Hence, we are interested in the asymptotic behavior of $a_q(n, \lceil\log_q n\rceil + z)$, $z \in \mathbb{Z}$ up to a better precision than the one suggested in Lemma 3. For this purpose, we provide in the next Lemma additional results on the asymptotic behavior of $a_q(n,k)$. The definition of $E_{k,q}$ is stated in Eq. (4).

Lemma 5: Let n, k be integers such that $0 < k \le n$. Then,

$$2^{nE_{k,q}} \le a_q(n,k) \le 2^{nE_{k,q}} + 2q^{n-\lceil k/2\rceil}.$$

Proof:

Part 1 (Upper Bound):

We use the notations $D_q(n,k), d_q(n,k)$ from the proof of Lemma 3 and consider the set $G(m) = D_q(n,k)^m$. We also denote $g(m) = |G(m)| = d_q(n,k)^m$. Note that the set $G(m)$ satisfies the k-RLL constraint, and hence

$$\lim_{m\to\infty}\frac{\log(g(m))}{nm} \le E_{k,q}.$$

We therefore get

$$\lim_{m\to\infty}\frac{\log\left(d_q(n,k)^m\right)}{nm} = \frac{\log(d_q(n,k))}{n} \le E_{k,q}.$$

By using Equation (7) we have

$$\frac{\log\left(a_q(n,k) - 2q^{n-\lceil k/2\rceil}\right)}{n} \le E_{k,q},$$

and

$$a_q(n,k) \le 2^{nE_{k,q}} + 2q^{n-\lceil k/2\rceil}.$$

Part 2 (Lower Bound):

For all positive integer m we have $A_q(mn,k) \subseteq A_q(n,k)^m$ and therefore we deduce that

$$\begin{aligned} E_{k,q} &= \lim_{m\to\infty}\frac{\log(a_q(mn,k))}{mn} \\ &\le \lim_{m\to\infty}\frac{\log\left(a_q(n,k)^m\right)}{mn} \\ &= \frac{\log(a_q(n,k))}{n}, \end{aligned}$$

and the lower bound follows directly. ■

For some values of k the result from Lemma 5 gives weaker bounds than the one presented in Lemma 3. However, for the case of $k = \lceil\log_q n\rceil + z$, $z \in \mathbb{Z}$, we next show that the lower and upper bound of Lemma 5 are asymptotically tight.

Recall that Jain *et al.* [15] showed that

$$E_{k,q} = \log q - \frac{(q-1)\log e}{q^{k+2}}(1 + o(1)) \tag{8}$$

or in other words,

$$\lim_{k\to\infty}\frac{(q-1)\log e q^{-k-2}}{\log q - E_{k,q}} = 1. \tag{9}$$

According to this result and the properties proved in Proposition 6 and Lemma 7 we conclude in Lemma 8 what the asymptotic behavior of $2^{nE_{k,q}}$ is. Then, in Theorem 9 we show that $2^{n-\lceil k/2\rceil+1}$ is negligible relatively to $2^{nE_{k,q}}$, therefore the bounds in Lemma 5 meet and the asymptotic behavior of $a_q(n,k)$ is established.

The following proposition will be used in the proof of Lemma 8 and its proof is given in Appendix A.

Proposition 6: Let $f(n)$, $g(n)$ be functions such that $\lim_{n\to\infty} g(n) = 1$ and $1 \le f(n) \le C$ for a constant C. Then,

$$f(n)^{g(n)} \approx f(n).$$

Note that the requirement that $f(n)$ is bounded is essential, otherwise the proposition does not hold. For example if $f(n) = 2^n$ and $g(n) = 1 + \frac{1}{n}$ we have that $\lim_{n\to\infty}\frac{f(n)^{g(n)}}{f(n)} = 2$.

Lemma 7: There exists an integer N such that for all $n \ge N$ and $k = \lceil\log_q n\rceil + z$, $z \in \mathbb{Z}$

$$\log q - \frac{C}{n} \le E_{k,q} \le \log q,$$

for some constant $C > 0$ which is independent of n.

Proof: First, $E_{k,q} \le \log q$ from the definition of $E_{k,q}$. From Corollary 4 there exists a constant C' such that for large enough n

$$\text{red}(A_q(n, \lceil\log_q n\rceil + z)) \le C',$$

and equivalently $a_q(n, \lceil\log_q n\rceil + z) \ge q^{n-C'}$. From Lemma 5

$$\begin{aligned} E_{k,q} &\ge \frac{\log\left(a_q(n,k) - 2q^{n-\lceil k/2\rceil}\right)}{n} \\ &\ge \frac{\log\left(q^{n-C'} - 2q^{n-\lceil k/2\rceil}\right)}{n} \\ &= \frac{n\log q + \log\left(q^{-C'} - 2q^{-\lceil k/2\rceil}\right)}{n}. \end{aligned}$$

There exists an integer N such that for all $n \ge N$, $2q^{-\lceil k/2\rceil} \le 0.5q^{-C'}$. Thus,

$$\begin{aligned} E_{k,q} &\ge \frac{n\log q + \log\left(q^{-C'} - 0.5q^{-C'}\right)}{n} \\ &= \frac{n\log q + \log\left(0.5q^{-C'}\right)}{n} \\ &= \log q - \frac{1 + C'\log q}{n} \end{aligned}$$

and by choosing $C = 1 + C'\log q$ the result follows. ■

We are now ready to use the result of Jain *et al.* [15] mentioned in Equation (II), in order to establish the following result.

Lemma 8: For $k = \lceil \log_q n \rceil + z, z \in \mathbb{Z}$,

$$2^{nE_{k,q}} \approx \frac{q^n}{e^{(q-1)q^{\Delta_n - z - 1}}},$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$.

Proof: We use Proposition 6 with $f(n) = 2^{n(\log q - E_{k,q})}$ and $g(n) = \frac{(q-1)\log e q^{-k-2}}{\log q - E_{k,q}}$. From Lemma 7, when $k = \lceil \log_q n \rceil + z$,

$$1 \le 2^{n(\log q - E_{k,q})} = f(n) \le 2^C.$$

From Eq. (9) $\lim_{n\to\infty} g(n) = 1$ thus the requirements of the proposition are satisfied and we get

$$2^{n(\log q - E_{k,q})} \approx 2^{n(q-1)\log e q^{-(k-1)-2}}.$$

We conclude that

$$\begin{aligned} 2^{nE_{k,q}} &= 2^{n\log q + n(E_{k,q} - \log q)} \\ &\approx 2^{n\log q - n(q-1)\log e q^{-k-1}} \\ &\approx 2^{n\log q - n(q-1)\log e q^{-\lceil \log_q n \rceil - z - 1}} \\ &\approx \frac{q^n}{e^{(q-1)q^{\Delta_n - z - 1}}}, \end{aligned}$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$. ■

Finally, we can now apply the result of Lemma 8 in Lemma 5 and obtain the asymptotic behavior of $a_q(n, \lceil \log_q n \rceil + z)$ in the following theorem.

Theorem 9: For $k = \lceil \log_q n \rceil + z$, $z \in \mathbb{Z}$,

$$a_q(n,k) \approx \frac{q^n}{e^{(q-1)q^{\Delta_n - z - 1}}},$$

where $\Delta_n = \log n - \lceil \log n \rceil$.

Proof: Lemma 5 gives us

$$1 \le \frac{a_q(n,k)}{2^{nE_{k,q}}} \le 1 + \frac{2q^{n-\lceil k/2 \rceil}}{2^{nE_{k,q}}}.$$

By using Lemma 8 we get that for $k = \lceil \log_q n \rceil + z$

$$\lim_{n\to\infty} \frac{2q^{n-\lceil k/2 \rceil}}{2^{nE_{k,q}}} = \lim_{n\to\infty} \frac{2q^{n-\lceil k/2 \rceil}}{q^n e^{-(q-1)q^{\Delta_n - z - 1}}} = 0,$$

and conclude that

$$1 \le \lim_{n\to\infty} \frac{a_q(n,k)}{2^{nE_{k,q}}} \le \lim_{n\to\infty} 1 + \frac{2q^{n-\lceil k/2 \rceil}}{2^{nE_{k,q}}} = 1.$$

Therefore,

$$a_q(n,k) \approx 2^{nE_{k,q}},$$

and together with Lemma 8 the result follows directly. ■

Remark 10: We would like to note that the results in this section match the results by Schilling [28] on the distribution of the longest runs in arbitrary vectors. In particular, it is stated in [28] that the typical length of the longest run in n flips of a fair coin converges to $\log n - 1$. However, we find the proof in this section to be more accurate for the purpose of exactly calculating the number of redundancy bits of k-RLL codes.

B. Efficient Encoding and Decoding Algorithm

In this subsection we provide an algorithm to efficiently encode and decode vectors which avoid zero run of length $\lceil \log_q n \rceil + 1$. We show the algorithm for the binary case, however, it is straightforward to extend it for $q > 2$. According to Theorem 9 the redundancy of $A_2(n, \log n + 1)$ is approximately $\frac{\log e}{4} \approx 0.36$ when $n = 2^i, i \in \mathbb{N}$. The algorithm described next uses one redundancy bit, however, it has linear encoding and decoding complexities.

Algorithm 1 Zero Run-Length Encoding

Input: Sequence $\mathbf{x} \in \Sigma^{n'}$, $n' \le n$
Output: $\mathbf{y} \in \Sigma^{n'+1}$ with zero runzero run length $\le \lceil \log n \rceil$
Define $\mathbf{y} = \mathbf{x}1 \in \Sigma^{n'+1}$
Set $i = 1$ and $i_{end} = n'$
while $i \le i_{end} - \lceil \log n \rceil$ **do**
  if $\omega_H(\mathbf{y}_i^{i+\lceil \log n \rceil}) = 0$ **then**
    remove the zero run $\mathbf{y}_i^{i+\lceil \log n \rceil}$ from $\mathbf{y}$
    p(i): binary representation of i with $\lceil \log n \rceil$ bits
    append **p(i)**0 to the right of $\mathbf{y}$
    set $i_{end} = i_{end} - \lceil \log n \rceil - 1$
  else
    set $i = i + 1$
  end if
end while

The following lemma proves the correctness of Algorithm 1.

Lemma 11: For all $n' \le n$, given any vector $\mathbf{x} \in \Sigma^{n'}$, Algorithm 1 outputs a sequence $\mathbf{y} \in \Sigma^{n'+1}$, where any zero run has length at most $\lceil \log n \rceil$ and such that $\mathbf{x}$ can be uniquely reconstructed given $\mathbf{y}$. Furthermore, the time and space complexity of the algorithm and its inverse is $\Theta(n)$.

Proof: The algorithm starts by initializing $\mathbf{y} = \mathbf{x}1$. We then iterate over the indices of $\mathbf{y}$ that correspond to the indices of the input word $\mathbf{x}$. If we encounter an index in which a $\lceil \log n \rceil + 1$ zero run starts, we remove the run and append **p(i)**0, which we call a pointer, to the right of $\mathbf{y}$, where **p(i)** is the binary representation of the index i.

First, notice that each appended pointer **p(i)**0 has the same length as the corresponding removed run. Therefore throughout the algorithm the length of $\mathbf{y}$ does not change. There exists an index $1 \le t \le n'$ such that the output $\mathbf{y}$ is of the form $\mathbf{y}_1^t 1 \mathbf{y}_{t+2}^{n'+1}$ where $\mathbf{y}_1^t$ is the remainder of $\mathbf{x}$ after removing the zero runs and $\mathbf{y}_{t+2}^{n'+1}$ is the list of the pointers $(\mathbf{p(i)}, 0)$ representing the indices of the removed zero runs.

To reconstruct $\mathbf{x}$ given $\mathbf{y}$ we start by locating the separating bit 1 on position t. We start from the right and check whether the rightmost bit is 1 or 0. In case it is 0, the $\lceil \log n \rceil$ bits to the left correspond to a pointer, we skip them and repeat the process until we encounter the separating 1. We then construct the original $\mathbf{x}$ by inserting zero runs of length $\lceil \log n \rceil + 1$ to the remainder part $\mathbf{y}_1^t$ according to the pointers part $\mathbf{y}_{t+2}^{n'+1}$.

Next, we show that $\mathbf{y}$ does not contain a zero run of length greater than $\lceil \log n \rceil$. It is clear that $\mathbf{y}_1^t$ does not contain such a run. The separating 1 ensures that there is no zero run which starts in $\mathbf{y}_1^t$ and ends in $\mathbf{y}_{t+2}^{n'+1}$. The structure of $\mathbf{y}_{t+2}^{n'+1}$ is a sequence of concatenated pointers of the form $\mathbf{p(i)}0$. It suffices to show that any sub-vector of $\mathbf{y}_{t+2}^{n'+1}$ of the form $\mathbf{p(j)}0\mathbf{p(k)}$ does not consist a zero run of length greater than $\lceil \log n \rceil$. Here, j and k represent the indices of two consecutive locations where a $\lceil \log n \rceil + 1$ zero run was found in the while loop in the algorithm and note that $0 < j \le k$.

We consider the leftmost one in the binary representations $\mathbf{p(j)}$ and $\mathbf{p(k)}$. Since $j \le k$, the position of the leftmost one within $\mathbf{p(j)}$ is smaller or equal to the position of the leftmost one within $\mathbf{p(k)}$, thus any window of length $\lceil \log n \rceil + 1$ must contain at least one of the leftmost ones from $\mathbf{p(j)}$ and $\mathbf{p(k)}$.

Note that since the indices of the input vector $\mathbf{x}$ are indexed starting from 1, we do not write 0 in any of the pointers. We do not write n' either since the largest index we consider for zero runs is $n' - \lceil \log n \rceil$. Lastly, the complexity of the algorithm is $\mathcal{O}(n)$ since the complexity of each pointer update when encountering a zero run is $\Theta(\log n)$ and the number of these operations is at most $n'/\log n$. ■

The following example demonstrates how the encoding in Algorithm 1 works.

Example 12: Let $n' = n = 13$ and therefore $\lceil \log n \rceil = 4$ and $\lceil \log n \rceil + 1 = 5$. Consider the following sequence:

$$\boldsymbol{x} = 1000000000001,$$

Let us go through the steps of Algorithm 1.

1) $\boldsymbol{y} = \boldsymbol{x}1 = 10000000000011$
2) $i = 1$ *and* $i_{end} = 13$
3) $i = 1$: $w_H(\mathbf{y}_1^5) \neq 0,\ i = i + 1$
4) $i = 2$: $w_H(\mathbf{y}_2^6) = 0,$
   a) *Remove* $\mathbf{y}_2^6$ *from* $\mathbf{y}$: $\mathbf{y} = 100000011$
   b) *Define* $\boldsymbol{p(2)} = 0010$
   c) *Append* $\boldsymbol{p(2)}0 = 00100$: $\mathbf{y} = 10000001100100$
   d) *Set* $i_{end} = 13 - 5 = 8$
5) $i = 2$: $w_H(\mathbf{y}_2^6) = 0,$
   a) *Remove* $\mathbf{y}_2^6$ *from* $\mathbf{y}$: $\mathbf{y} = 101100100$
   b) *Define* $\boldsymbol{p(2)} = 0010$
   c) *Append* $\boldsymbol{p(2)}0 = 00100$: $\mathbf{y} = 10110010000100$
   d) *Set* $i_{end} = 8 - 5 = 3$
6) $i = 2$: $w_H(\mathbf{y}_2^6) \neq 0,\ i = i + 1.$

The decoding works as described in the proof of Lemma 11.

A similar algorithm to the problem solved by Algorithm 1 was recently proposed in [29] to efficiently encode sequences that do not contain runs of zeros and ones of length k with a single redundancy bit. Specifically, Schoeny *et al.* [29] showed how to accomplish this task with $k = \lceil \log n \rceil + 4$. Algorithm 1 can be slightly adjusted in order to solve the problem in [29] with $k = \lceil \log n \rceil + 2$. Lastly, we note that Kauts presented in [17] an algorithm which encodes all words avoiding zero runs of any specific length with optimal redundancy. However, the space complexity of this algorithm is $\Theta(n^2)$.

Algorithm 1 solves the problem of avoiding zero runs of length $\lceil \log n \rceil + 1$. However, it can be extended to avoid zero runs of any length in the following manner. Assume for example that we want to avoid zero runs of length $0.5 \log n$. We start with a length-n vector, divide it into blocks of length $\frac{\sqrt{n}}{2}$ and apply Algorithm 1 on each block since $0.5 \log n = \log(\frac{\sqrt{n}}{2}) + 1$. We append 1 to each of the output vectors and the final resulting vector is the concatenation of all of them. This approach can be applied for any value of k and it achieves optimal order of redundancy, with linear encoding and decoding complexities.

IV. Mutually Uncorrelated Codes

In this section we expand the study of MU codes. Specifically, we show that the lower bounds in (2), (3) are asymptotically tight.

We are interested in maximizing the value of $|\mathcal{C}_1(n, q, k)|$ over all values of k. Notice that

$$|\mathcal{C}_1(n, q, k)| = (q-1)^2 a_q(n-k-2, k),$$

and therefore the proof of the main theorem in this section highly relies on the analysis in Section III. However, in Section III we analyzed the asymptotic size $a_q(n, k)$ while we are actually interested in $a_q(n-k-2, k)$. We chose to present the analysis of $a_q(n, k)$ in Section III since it is similar to the analysis of $a_q(n-k-2, k)$ and we believe it will be of use in other problems as well. In this section we complete some missing parts to obtain the asymptotic behavior of the maximal value of $|\mathcal{C}_1(n, q, k)| = (q-1)^2 a_q(n-k-2, k)$. We mainly focus on the trade-off formed by the choice of k: on one hand, reducing the value of k results with a larger value for $n-k-2$ and thus smaller redundancy due to the fixed k-length prefix in the construction. On the other hand, smaller k implies a stronger k-RLL constraint which requires larger redundancy in the remaining part.

We start by showing that it is sufficient to look only into values of k which are of the form $\lceil \log_q n \rceil + z, z \in \mathbb{Z}$. Recall that our asymptotic notations assume q is fixed and $n \to \infty$.

Lemma 13: The value of k that minimizes the redundancy of $\mathcal{C}_1(n, q, k)$ is $\lceil \log_q n \rceil + \mathcal{O}(1)$.

Proof: From Corollary 4, in the case of $k = \lceil \log_q n \rceil + \mathcal{O}(1)$ the redundancy of $\mathcal{C}_1(n, q, k)$ is $k + \mathcal{O}(1) = \lceil \log_q n \rceil + \mathcal{O}(1)$. Since the redundancy of $\mathcal{C}_1(n, q, k)$ is at least k, greater values of k i.e., $k = \lceil \log_q n \rceil + \omega(1)$, lead to higher redundancy, and so we disregard them. For values k of the form $\log_q n - f(n)$, where $f(n) = \omega(1)$ we again turn to Corollary 4 to claim that the redundancy is $\Theta(k + 2^{f(n)}) = \Theta(\log_q n - f(n) + 2^{f(n)})$ which is also asymptotically greater than $\lceil \log_q n \rceil + \mathcal{O}(1)$. We therefore summarize that the minimal redundancy, or the maximal cardinality of $\mathcal{C}_1(n, q, k)$ is achieved when k is of the form $k = \lceil \log_q n \rceil + \mathcal{O}(1)$. ■

We next look further into the specific choice of k that maximizes the cardinality and obtain the asymptotic behavior of the maximal cardinality.

For the rest of this section we denote $n' = n - k - 2$. From Lemma 5 we have that

$$2^{n' E_{k,q}} \le a_q(n', k) \le 2^{n' E_{k,q}} + 2q^{n' - \lceil k/2 \rceil}. \tag{10}$$

In the following lemma we establish how $2^{n' E_{k,q}}$ behaves asymptotically for the values of k of interest to us. The proofs of Lemma 14 and Lemma 15 are attached in Appendix B as they share similar ideas to the proofs in Section III.

Lemma 14: For $k = \lceil \log_q n \rceil + z, z \in \mathbb{Z}$,

$$2^{n' E_{k,q}} \approx \frac{q^n}{n} \cdot \frac{q^{\Delta_n - z - 2}}{e^{(q-1)q^{\Delta_n - z - 1}}},$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$.

We apply the result of Lemma 14 in the inequality (10) to obtain the asymptotic behavior of $a_q(n', \lceil \log_q n \rceil + z)$.

Lemma 15: For $z \in \mathbb{Z}$,

$$|\mathcal{C}_1(n, q, \lceil \log_q n \rceil + z)| \approx \frac{q^n}{n} \left(\frac{q-1}{q} \right)^2 q^{\Delta_n - z - \log_q e(q-1)q^{\Delta_n - z - 1}}$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$.

Next we optimize this term over all values of $z \in \mathbb{Z}$ in order to establish the maximal cardinality of $\mathcal{C}_1(n, q, k)$.

Theorem 16:

$$\mathcal{C}_1(n, q) \approx \frac{q^n}{n} \cdot \left(\frac{q-1}{q}\right)^2 q^{F(\Delta_n)} \le \frac{q^n}{n} \cdot \frac{q-1}{eq},$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$ and

$$F(\Delta_n) = \max_{z \in \{-2,-1,0\}} \left\{ \Delta_n - z - \log_q(e)(q-1)q^{\Delta_n - z - 1} \right\}.$$

The inequality is tight when $n \to \infty$ over any subsequence of n that satisfies $\Delta_n = -\log_q(q-1)$.

Proof: From Lemma 15, when $k = \lceil \log_q n \rceil + z$ we get

$$|\mathcal{C}_1(n, q, k)| \approx \frac{q^n}{n} \left(\frac{q-1}{q} \right)^2 q^{\Delta_n - z - \log_q e(q-1)q^{\Delta_n - z - 1}}.$$

Let us denote

$$f(\Delta_n, z) = \Delta_n - z - \log_q e(q-1)q^{\Delta_n - z - 1}.$$

We are interested in the value of $z \in \mathbb{Z}$ that maximizes the size of $\mathcal{C}_1(n, q, k)$, that is, the value of $z \in \mathbb{Z}$ that maximizes $f(\Delta_n, z)$ for each Δ_n.

$$\frac{\partial f}{\partial z} = -1 + (q-1)q^{\Delta_n - z - 1},$$

so the only maximum of the function $f(\Delta_n, z)$ is achieved for $z_0 = \log_q(q-1) - 1 + \Delta_n$. However, z_0 is not necessarily an integer while we are interested in integers only. Since $-1 < \Delta_n \le 0$ we have $-2 < z_0 < 0$ thus the maximum over $z \in \mathbb{Z}$ is achieved by one of the options $z \in \{-2, -1, 0\}$.

We therefore obtain the following:

$$\mathcal{C}_1(n, q) \approx \frac{q^n}{n} \left(\frac{q-1}{q} \right)^2 q^{F(\Delta_n)},$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$ and

$$F(\Delta_n) = \max_{z \in \{-2,-1,0\}} \left\{ \Delta_n - z - \log_q(e)(q-1)q^{\Delta_n - z - 1} \right\}.$$

In Appendix B we analyze how $F(\Delta_n)$ behaves for each value of Δ_n and obtain that when $q = 2$

$$F(\Delta_n) = \begin{cases} f(\Delta_n, -2), & for\ -1 < \Delta_n \le \log(\ln 2) \\ f(\Delta_n, -1), & otherwise \end{cases}$$

and when $q > 2$

$$F(\Delta_n) = \begin{cases} f(\Delta_n, -1), & for\ -1 < \Delta_n \le \delta_0 \\ f(\Delta_n, 0), & otherwise \end{cases}$$

for $\delta_0 = -\log_q \frac{(q-1)^2}{q \ln q}$. We also discuss in Appendix B the maximal value of $F(\Delta_n)$ which leads to the maximal cardinality $\frac{q^n}{n} \cdot \frac{q-1}{eq}$ ■

The result of Theorem 16 aligns with the results from [13] and [20] which we recalled in Equations (2) and (3). The lower bound from (2) states that

$$\mathcal{C}_1(n, q) \gtrsim q^{-\frac{q}{q-1}} \ln q \frac{q^n}{n} \tag{2}$$

and it holds when $n \to \infty$ over any series of n. The lower bound (3) claims that

$$\mathcal{C}_1(n, q) \gtrsim \frac{q-1}{qe} \cdot \frac{q^n}{n} \tag{3}$$

and it holds when $n \to \infty$ over the subseries $\frac{q^i - 1}{q-1}, i \in \mathbb{N}$. Observe that when $n = \frac{q^i - 1}{q-1}$, $\lim_{n \to \infty} \Delta_n = -\log_q(q-1)$ thus the result of Theorem 16 and the bound (3) agree.

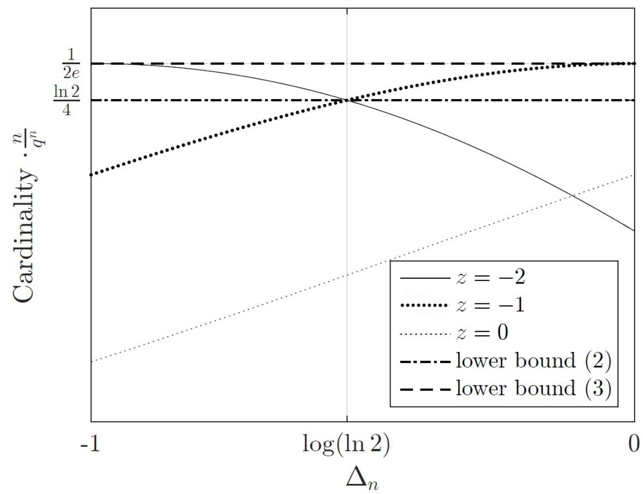
The bounds in (2) and (3) provide lower asymptotic bounds on the cardinality $\mathcal{C}_1(n, q)$, while in Theorem 16 we obtain the explicit expression of the asymptotic behavior for any value of Δ_n and show that the lower bounds are asymptotically tight. According to Lemma 15 the term

$$|\mathcal{C}_1(n, q, \lceil \log_q n \rceil + z)| \cdot \frac{n}{q^n}$$

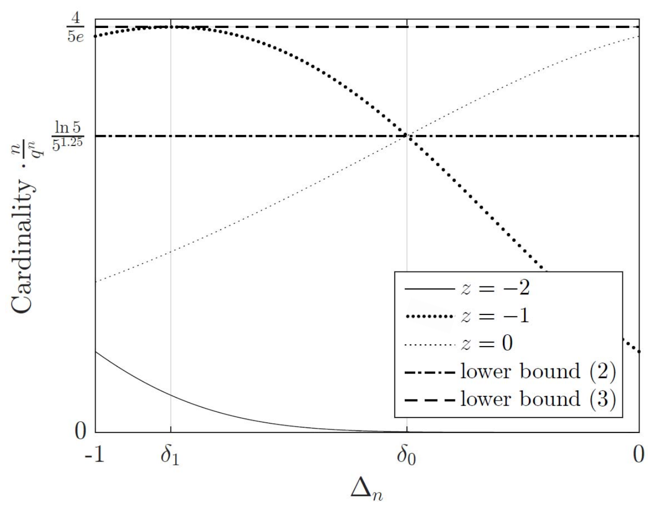
depends only on z, q and $\Delta_n = \log_q n - \lceil \log_q n \rceil$. Specifically it does not grow with n. In Fig. 1 we plot this term for the three interesting values of $z \in \{-2, -1, 0\}$ as a function of Δ_n, alongside the lower bounds from (2) and (3). The first, second graph in Fig. 1 corresponds to $q = 2$, $q = 5$, respectively. When $q = 5$ for example, the maximal cardinality is achieved when $z = -1$ or when $z = 0$, depending on the value of Δ_n. Among the range of values of Δ_n, $\delta_0 = -\log_5(\frac{16}{5 \ln 5}) \approx -0.43$ yields the minimal cardinality which aligns with the bound (2). The maximal cardinality is achieved for $\delta_1 = -\log_q(q-1) = -\log_5 4 \approx -0.86$ where it meets the bound (3), and the maximal cardinality is $\frac{q-1}{qe} \cdot \frac{q^n}{n}$. Recall that we consider q as a fixed number. This guarantees that the maximal cardinality $\frac{q-1}{qe} \cdot \frac{q^n}{n}$ is achieved as n tends to infinity over the subsequence $\frac{q^i - 1}{q-1}$, $i \in \mathbb{N}$. Taking this into account, an interesting fact to notice is that the ratio between the maximal cardinality $\frac{q-1}{qe} \cdot \frac{q^n}{n}$ and the upper bound from [23] $\frac{q^n}{e(n-1)}$ reduces as q grows. This is illustrated in Fig. 2. Having said that, note that in the binary case there is a gap of factor 2 between the maximal cardinality and the upper bound and closing this gap remains an open problem.

V. The Window Weight Limited Constraint

In this section we introduce a natural extension to the RLL constraint which we call the *window weight limited* constraint. We study this constraint for the purpose of constructing a new family of codes, called (d_h, d_m)-*Mutually Uncorrelated*



(a)



(b)

Fig. 1. Comparison between the construction's cardinality according to Lemma 15, multiplied by $\frac{n}{q^n}$ for different values of z, and the bounds (2) and (3) from [13], [20]. (a) Cardinalities for $q = 2$. (b) Cardinalities for $q = 5$.

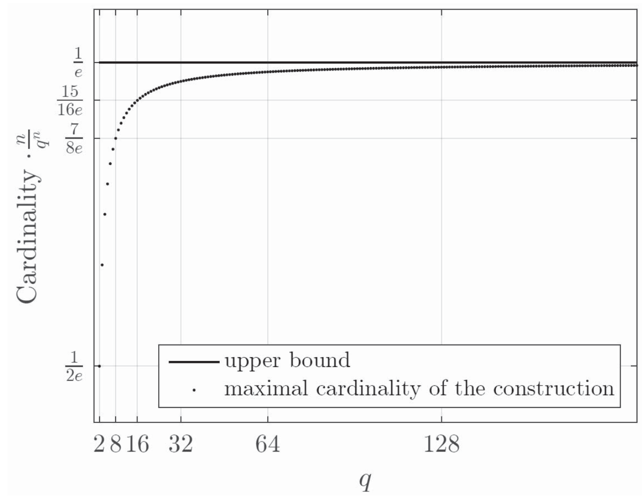


Fig. 2. A comparison between the construction's maximal cardinality according to Theorem 16 and the upper bound of the cardinality of MU codes from [13] and [20], mentioned in (1).

Codes, which will be presented later in Section VI. Our main results in this section are an upper bound on the size of the set of vectors satisfying the window weight limited constraint and a construction with efficient encoding and decoding, and almost optimal cardinality. We start with the definition of this constraint.

*Definition 17: Let d and k be positive integers. We say that a vector $\boldsymbol{a} \in \Sigma_q^n$ satisfies the (d,k)-**window weight limited (WWL)** constraint, and is called a (d,k)-**WWL** vector, if $n < k$ or $\forall i \in [n-k+1] : w_H(\boldsymbol{a}_i^{i+k-1}) \geq d$.*

We call a set of (d,k)-WWL vectors a (d,k)-*WWL code* and we denote by $A_q(n,k,d)$ the set of all (d,k)-WWL vectors over Σ_q^n. Lastly, $a_q(n,k,d) = |A_q(n,k,d)|$.

This constraint states that a vector $\mathbf{a} \in \Sigma_q^n$ is a (d,k)-WWL vector if the Hamming weight of every consecutive subsequence of length k in $\mathbf{a}$ is at least d. A similar constraint was studied in [27] when studying time space codes for phase change memories. However, Qin *et al.* [27] studied the opposite constraint in which the weight of every window of length k is *at most* d. Furthermore, as will be explained later, we are interested in the case where k depends on the word length n as opposed to the analysis in [27] where k is fixed. Notice that the $(1,k)$-WWL constraint is equivalent to the k-RLL constraint. In the next lemma we provide an upper bound on the size of $A_q(n,k,d)$. The proof is deferred to Appendix C since it shares similar ideas with the proof of the upper bound in Lemma 3.

Lemma 18: Let n,k,d be positive integers such that $d \leq k \leq n$. Then, there exists a constant $C > 0$ such that for n large enough

$$a_q(n,k,d) \leq q^{n - C\frac{(n-2k)k^{d-1}}{q^k}}.$$

For the rest of the paper we let $\mathcal{F}(n,d)$ be

$$\mathcal{F}(n,d) = \lceil \log_q n \rceil + (d-1)(\lceil \log_q \lceil \log_q n \rceil \rceil + C) + 2, \quad (11)$$

where C is a constant equal to the minimum integer such that $\lceil \log_q \lceil \log_q n \rceil \rceil + C \geq \lceil \log_q(\mathcal{F}(n,d)+2) \rceil + 1$. From Lemma 18 we also have that

$$C_1 \frac{(n-2k)k^{d-1}}{q^k} \leq \operatorname{red}(A_q(n,k,d)),$$

where $C_1 > 0$ is a constant. By setting k to be $\mathcal{F}(n,d) - f(n)$, $f(n) > 0$ we get that the redundancy of $A_q(n, \mathcal{F}(n,d) - f(n), d)$ is $\Omega(q^{f(n)})$.

Next we present an explicit algorithm for encoding and decoding $(d, \mathcal{F}(n,d))$-WWL vectors with $n' \leq n$ information bits and d redundancy symbols.

The next lemma proves the correctness of Algorithm 2. Its proof is deferred to Appendix C.

Lemma 19: For all $n' \leq n$, given any vector $\boldsymbol{x} \in \Sigma_q^{n'}$ Algorithm 2 outputs a $(d, \mathcal{F}(n,d))$-WWL vector $\boldsymbol{y} \in \Sigma_q^{n'+d}$ such that $\boldsymbol{x}$ can be uniquely reconstructed given $\boldsymbol{y}$. The time and space complexity of the algorithm and its inverse is $\Theta(n)$.

TABLE I

REDUNDANCY SUMMARY OF k-RLL AND (d,k)-WWL CONSTRAINTS

Constraint	k-RLL	k-RLL	(d,k)-WWL
Redundancy	$\Theta(\frac{n}{q^k})$	$\log_q e(q-1)q^{\Delta_n - z - 1}$	$\Theta(\frac{nk^{d-1}}{q^k})$
Comments	$k \le \log_q n + \mathcal{O}(1)$	$k = \lceil \log_q n \rceil + z, z \in \mathbb{Z}, \Delta_n = \log_q n - \lceil \log_q n \rceil$	$k = \log_q n + (d-1)\log_q \log_q n - \Omega(1)$
Stated in	Corollary 4	Theorem 9	Corollary 20

Algorithm 2 Window Weight Limited Encoding

Input: $\mathbf{x} \in \Sigma_q^{n'}$ and an integer $d > 1$

Output: $(d, \mathcal{F}(n,d))$-WWL vector $\mathbf{y} \in \Sigma_q^{n'+d}$

Define $\mathbf{y} = \mathbf{x}1^d \in \Sigma_q^{n'+d}$
Set $i = 1$ and $i_{end} = n'$
while $i \le i_{end} - \mathcal{F}(n,d) + 1$ **do**
  if $w_H(\mathbf{y}_i^{i+\mathcal{F}(n,d)-1}) < d$ **then**
    remove $\mathbf{y}_i^{i+\mathcal{F}(n,d)-1}$ from $\mathbf{y}$
    $p(i)$: q-ary representation of i with $\lceil \log_q n \rceil$ symbols
    for $j = 1, \ldots, d-1$ **do**
      if $w_H(\mathbf{y}_i^{i+\mathcal{F}(n,d)-1}) \ge j$ **then**
        $t(j)$:q-ary index of the j-th non-zero symbol in $\mathbf{y}_i^{i+\mathcal{F}(n,d)-1}$, appended by the j-th non-zero symbol
      else
        $t(j) = 1^{\lceil \log_q \lceil \log_q n \rceil \rceil + C}$
      end if
    end for
    append $p(i)t(1)\cdots t(d-1)01$ to the right of $\mathbf{y}$
    set $i_{end} = i_{end} - \mathcal{F}(n,d)$
    set $i = i - \mathcal{F}(n,d) + 1$
  else
    set $i = i + 1$
  end if
end while

Algorithm 2 solves the problem of WWL encoding by replacing each subvector of small weight with the index of the subvector (denoted by $p(i)$), appended by the indices of the non-zeros within it (denoted by $t(j)$s). The algorithm works when the window's length k is $\mathcal{F}(n,d) = \log_q n + (d-1)(\log_q(\log_q n) + \Theta(1))$, with a constant number of redundancy symbols. From Lemma 18, the redundancy in this case is at least $\Theta(1)$, hence the redundancy of the algorithm is optimal up to a constant number of bits. Lastly, we can also extend this algorithm to encoding WWL vectors with smaller values of k, that is $k = \mathcal{F}(n,d) - f(n)$, $f(n) > 0$, by breaking down the length-n vector to $q^{f(n)}$ blocks of length $n/q^{f(n)}$, applying Algorithm 2 on each block, and stitching the output vectors with d ones between each two subvectors. This approach yields redundancy of approximately $2dq^{f(n)}$, which is also optimal up to a constant factor, according to Lemma 18. The following corollary is established

Corollary 20: Let $f(n)$ be a positive function such that $\mathcal{F}(n,d) - f(n)$ is a positive integer. The redundancy of $A_q(n, \mathcal{F}(n,d) - f(n), d)$ is $\Theta(q^{f(n)})$.

We hereby summarize our study on the k-RLL and the (d,k)-WWL constraints. Our main results are presented in Table I. Note that the first column of the k-RLL constraint lists our result for all values of k such that $k \le \log_q n + \mathcal{O}(1)$ and provides the redundancy order in this case. On the other hand, the second column is targeted towards specific values of k and gives an exact value of the redundancy.

VI. (d_h, d_m)-MUTUALLY UNCORRELATED CODES

Since substitution errors may also happen both in the writing and reading processes of DNA molecules, we impose in this section a stronger and more general version of the mutual uncorrelatedness constraint, by requiring the prefixes and suffixes to not only differ but also have a large Hamming distance.

*Definition 21: A code $\mathcal{C} \subseteq \Sigma_q^n$ is called a (d_h, d_m)-**MU code** if*

1) *the minimum Hamming distance of the code is d_h,*
2) *for every two not necessarily distinct words $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$, and $i \in [n-1]$: $d_H(\boldsymbol{a}_1^i, \boldsymbol{b}_{n-i+1}^n) \ge \min\{i, d_m\}$.*

We set $A_{MU}(n, q, d_h, d_m)$ to be the largest cardinality of a (d_h, d_m)-MU code over Σ_q^n, and $M(n,q,d)$ as the largest cardinality of a length-n code over Σ_q^n with minimum Hamming distance d. According to the sphere packing bound we have that $M(n,q,d) \le q^n/(Cn^{\lfloor \frac{d-1}{2} \rfloor})$ for some C that depends on q and d. Since we refer to q and d as fixed parameters, we will treat C as fixed as well.

Motivated by Levenshtein's upper bound [22], we first provide an upper bound on the value $A_{MU}(n, q, d_h, d_m)$.

Theorem 22: For all positive integers n, q, d_h, d_m and $d = \min\{d_h, 2d_m\}$,

$$A_{MU}(n, q, d_h, d_m) \le \frac{M(n,q,d)}{\lfloor n/d_m \rfloor}.$$

Hence, the minimum redundancy of a (d_h, d_m)-MU code is

$$\left\lfloor \frac{d+1}{2} \right\rfloor \log_q n - \log_q d_m + \mathcal{O}(1).$$

Proof: Let $\mathcal{C} \subseteq \Sigma_q^n$ be a (d_h, d_m)-MU code. We let

$$\widehat{\mathcal{C}} = \{(\mathbf{aa})_{i+1}^{n+i} | \mathbf{a} \in \mathcal{C}, i = \alpha \cdot d_m, \alpha \in [0, \lfloor n/d_m \rfloor - 1]\}.$$

That is, the code $\widehat{\mathcal{C}}$ consists all cyclic shifts of words from $\mathcal{C}$ by αd_m bits. For $\mathbf{a}, \mathbf{b} \in \mathcal{C}$ let $\widehat{\mathbf{a}} = (\mathbf{aa})_{i+1}^{n+i}, \widehat{\mathbf{b}} = (\mathbf{bb})_{j+1}^{n+j}$ with $i = \alpha_i d_m$ and $j = \alpha_j d_m$, $\alpha_i, \alpha_j \in [0, \lfloor n/d_m \rfloor - 1]$. Note that $\widehat{\mathbf{a}}, \widehat{\mathbf{b}} \in \widehat{\mathcal{C}}$. We prove that if $i \ne j$ or $\mathbf{a} \ne \mathbf{b}$, then $d_H(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \ge \min\{d_h, 2d_m\}$. First, if $i = j$ and $\mathbf{a} \ne \mathbf{b}$ we have that $d_H(\mathbf{a}, \mathbf{b}) = d_H(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \ge d_h$. Otherwise, $i \ne j$ and we assume without loss of generality that $i > j$. Notice that $i > j$ implies $n \ge 2d_m$, otherwise if $n < 2d_m$ then $\mathcal{C} = \widehat{\mathcal{C}}$, meaning $i = j$. In this case $\widehat{\mathbf{a}}_{n-i+1}^{n-j}$ is a prefix of $\mathbf{a}$ and $\widehat{\mathbf{b}}_{n-i+1}^{n-j}$ is a suffix of $\mathbf{b}$. Moreover, the length of $\widehat{\mathbf{a}}_{n-i+1}^{n-j}$ is at least d_m, therefore, from the definition of (d_h, d_m)-MU code, $d_H(\widehat{\mathbf{a}}_{n-i+1}^{n-j}, \widehat{\mathbf{b}}_{n-i+1}^{n-j}) \ge d_m$. Similarly, $\widehat{\mathbf{b}}_{n-j+1}^{n}\widehat{\mathbf{b}}_1^{n-i}$

is a prefix of $\mathbf{b}$ and $\widehat{\mathbf{a}}_{n-j+1}^{n}\widehat{\mathbf{a}}_1^{n-i}$ is a suffix of $\mathbf{a}$, thus $d_H(\widehat{\mathbf{a}}_{n-j+1}^{n}\widehat{\mathbf{a}}_1^{n-i}, \widehat{\mathbf{b}}_{n-j+1}^{n}\widehat{\mathbf{b}}_1^{n-i}) \geq d_m$. Therefore,

$$d_H(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) = d(\widehat{\mathbf{a}}_1^{n-i}\widehat{\mathbf{a}}_{n-i+1}^{n-j}\widehat{\mathbf{a}}_{n-j+1}^{n}, \widehat{\mathbf{b}}_1^{i}\widehat{\mathbf{b}}_{n-i+1}^{n-j}\widehat{\mathbf{b}}_{n-j+1}^{n}) \geq 2d_m.$$

Since we showed that $d_H(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \geq \min\{d_h, 2d_m\}$ we conclude that all cyclic shifts by $\lfloor n/d_m \rfloor$ symbols of codewords from $\mathcal{C}$ are distinct, and the size of $\widehat{\mathcal{C}}$ is $\lfloor n/d_m \rfloor \cdot |\mathcal{C}|$. Moreover, $\widehat{\mathcal{C}}$ is a code with minimum Hamming distance $d = \min\{d_h, 2d_m\}$ which implies that

$$|\widehat{\mathcal{C}}| = \lfloor n/d_m \rfloor \cdot |\mathcal{C}| \leq M(n, q, d),$$

and the theorem follows directly. ■

We are now ready to show a construction of (d_h, d_m)-MU codes. For the sake of simplicity, all the constructions presented in the rest of the paper are for the binary case.

We say that a vector $\mathbf{u} \in \Sigma^{\ell}$ is a *d-auto-cyclic vector* if for every $1 \leq i \leq d$, $d_H(\mathbf{u}, 0^i\mathbf{u}_1^{\ell-i}) \geq d$. In the next construction we use the following d-auto-cyclic vector $\mathbf{u}$ of length $\ell(d) = d\lceil \log d \rceil + d$, which is given by

$$\mathbf{u} = 1^d\mathbf{u}_0 \cdots \mathbf{u}_{\lceil \log d \rceil - 1} \quad (12)$$

such that $\mathbf{u}_i = ((1^{2^i}0^{2^i})^d)_1^d$. For example, if $d = 5$ we have $\lceil \log d \rceil = 3$ and $\mathbf{u} = 11111\ 10101\ 11001\ 11110$.

Construction II: Let n, k be two integers such that $k \geq \ell(d_m)$ and $n \geq k + \ell(d_m) + 2d_m$. Denote $n' = n - k - \ell(d_m) - 2d_m$ and let $\mathcal{C}_H$ be a length-n' (d_m, k)-WWL code with minimum Hamming distance d_h. The code $\mathcal{C}_2(n, k) \subseteq \{0, 1\}^n$ is defined as follows,

$$\mathcal{C}_2(n, k, d_h, d_m) = \{0^k\boldsymbol{u}1^{d_m}\boldsymbol{c}1^{d_m} \mid \boldsymbol{c} \in \mathcal{C}_H\}.$$

where $\boldsymbol{u}$ is defined in Eq.(12).

The correctness of Construction II is proved in the next theorem.

Theorem 23: The code $\mathcal{C}_2(n, k, d_h, d_m)$ is a (d_h, d_m)-MU code.

Proof:

For simplicity of notation let $\mathcal{C} = \mathcal{C}_2(n, k, d_h, d_m)$ and $\mathbf{a}, \mathbf{b} \in \mathcal{C}$. The code $\mathcal{C}$ has minimum distance d_h since $\mathbf{a}_{k+\ell(d_m)+d_m+1}^{n-d_m}, \mathbf{b}_{k+\ell(d_m)+d_m+1}^{n-d_m} \in \mathcal{C}_H$ and $\mathcal{C}_H$ has minimum distance d_h. For the second part of the proof we use the following claim. The proof is left to the reader.

Claim 24: Let $\boldsymbol{x}, \boldsymbol{y}$ be two (d, k)-WWL vectors, then the vector $\boldsymbol{x}1^d\boldsymbol{y}$ is also a (d, k)-WWL vector.

We show that for any $\mathbf{a}, \mathbf{b}$ which are not necessarily distinct, for all $i \in [n-1]$: $d_H(\mathbf{a}_1^i, \mathbf{b}_{n-i+1}^n) \geq \min\{i, d_m\}$. We consider the following cases:

(1) For $i \in [1, d_m]$, $\mathbf{a}_1^i = 0^i, \mathbf{b}_{n-i+1}^n = 1^i$, and thus $d_H(\mathbf{a}_1^i, \mathbf{b}_{n-i+1}^n) = i = \min\{i, d_m\}$.
(2) For $i \in [d_m + 1, k]$, $\mathbf{a}_1^i = 0^i, \mathbf{b}_{n-i+1}^n = \mathbf{b}_{n-i+1}^{n-d_m}1^{d_m}$, and hence $d_H(\mathbf{a}_1^i, \mathbf{b}_{n-i+1}^n) \geq d_m$.
(3) For $i \in [k + 1, n - d_m]$, notice that $\mathbf{b}_{d_m+1}^n = 0^{k-d_m}1^{d_m}\mathbf{u}_{d_m+1}^{\ell(d_m)}1^{d_m}\mathbf{c}1^{d_m}$, where $\mathbf{c} \in \mathcal{C}_H$, 0^{k-d_m}, and $\mathbf{u}_{d_m+1}^{\ell(d_m)}$ are all (d_m, k)-WWL vectors. From Claim 24 $\mathbf{b}_{d_m+1}^n$ is also a (d_m, k)-WWL vector. Since $n - i + 1 > d_m$, $\mathbf{b}_{n-i+1}^n$ is a subvector $\mathbf{b}_{d_m+1}^n$ and as such, its first k positions consist at least d_m ones. Hence, $d_H(\mathbf{a}_1^i = 0^k\mathbf{a}_{k+1}^i, \mathbf{b}_{n-i+1}^n) \geq d_m$.
(4) For $i \in [n-d_m+1, n-1]$, let $j = n-i$, and $\widehat{\mathbf{a}} = \mathbf{a}_1^i$, $\widehat{\mathbf{b}} = \mathbf{b}_{n-i+1}^n = \mathbf{b}_{j+1}^n$. Notice that $\widehat{\mathbf{a}}_{k-j}^{k-j+\ell(d_m)-1} = 0^j\mathbf{u}_1^{\ell(d_m)-j}$ and $\widehat{\mathbf{b}}_{k-j}^{k-j+\ell(d_m)-1} = \mathbf{u}$. $\mathbf{u}$ is a d_m-auto-cyclic vector, hence $d_H(0^j\mathbf{u}_1^{\ell(d_m)-j}, \mathbf{u}) \geq d_m$ and we get $d_H(\mathbf{a}_1^i, \mathbf{b}_{n-i+1}^n) = d_H(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \geq d_H(\widehat{\mathbf{a}}_{k-j}^{k-j+\ell(d_m)}, \widehat{\mathbf{b}}_{k-j}^{k-j+\ell(d_m)}) \geq d_m$. ■

Next, we are interested in determining the maximum value of the codes' cardinality from Construction II, when optimizing over all possible values of k. We start with the case $d_h = 1$, that is, the case in which we don't require the codewords to differ by more than one symbol, but we require the MU property with distance d_m.

Lemma 25: The redundancy of the code $\mathcal{C}_2(n, k, 1, d_m)$ is minimized when k is of the form $k = \log n + (d_m - 1)\log\log n + \Theta(1)$. The minimal redundancy is $k + \Theta(1) = \log n + (d_m - 1)\log\log n + \Theta(1)$.

Proof: The construction's redundancy includes the $k + 2$ fixed bits and the redundancy bits of $\mathcal{C}_H$. Therefore, values of k of the form $k = \mathcal{F}(n, d_m) + \omega(1)$ result in redundancy greater than $\log n + (d_m - 1)\log\log n + \Theta(1)$. When $k = \mathcal{F}(n, d_m) + \Theta(1)$ we can use Algorithm 2 to construct $\mathcal{C}_H$ with $\Theta(1)$ redundancy bits and get total redundancy of $\log n + (d - 1)\log\log n + \Theta(1)$ bits. Lastly, from Corollary 20, when $k = \mathcal{F}(n, d_m) - f(n)$, $f(n) \geq 0$ the redundancy of $\mathcal{C}_2(n, k, 1, d_m)$ is $k + \Theta(2^{f(n)}) = \log n + (d - 1)\log\log n - f(n) + \Theta(2^{f(n)})$. This term is minimized when $f(n) = \Theta(1)$ and again, it yields total redundancy of $\log n + (d - 1)\log\log n + \Theta(1)$. ■

According to Lemma 25, Construction II provides existence of $(d_h = 1, d_m)$-MU codes which are $\Theta(\log\log n)$ away from the lower bound on redundancy in Theorem 22. Note that the results so far did not include an explicit description of an encoder and decoder for general values of d_h, d_m. In order to provide such an efficient construction we present in our next result a (d_h, d_m)-MU code with linear encoding and decoding complexities, and $\lfloor \frac{d_h+1}{2} \rfloor \log n + (d_m - 1) \cdot \log\log n + \mathcal{O}(d_m \log d_m)$ redundancy bits. In this case, it is also $\Theta(\log\log n)$ away from the bound on redundancy in Theorem 22.

For the suggested construction, we choose $k = \mathcal{F}(n, d_m) + 1$ in Construction II, and construct the code $\mathcal{C}_H$ with Algorithm 2 to provide the WWL property. We also allow greater values of d_h by incorporating a systematic code of minimum hamming distance d_h. This result is stated in the following corollary.

Corollary 26: There exists a binary (d_h, d_m)-MU code with redundancy $\left\lfloor \frac{d_h+1}{2} \right\rfloor \log n + (d_m - 1)\log\log n + \mathcal{O}(d_m \log d_m)$ and linear time and space encoding and decoding complexities.

Proof Sketch: We use Algorithm 2 to generate a $(d_m, \mathcal{F}(n, d_m))$-WWL code of length $\tilde{n} < n$, where the choice of $\tilde{n}$ will be explained later. We then guarantee minimum distance d_h by applying a systematic BCH encoder on the output from Algorithm 2. The approximately $\lfloor \frac{d_h-1}{2} \rfloor \log(\tilde{n})$ redundancy bits are spread within the $\tilde{n}$ bits such that the difference in the positions of each two redundancy bits is greater than $\mathcal{F}(n, d_m) + 1$ (we assume that $\tilde{n}$ is large enough to allow this).

That way the resulting vector is a $(d_m, \mathcal{F}(n, d_m) + 1)$-WWL vector. We denote this resulting set of vectors by $\mathcal{C}'$, so the code $\mathcal{C}'$ is a $(d_m, \mathcal{F}(n, d_m) + 1)$-WWL code with minimum distance d_h and its length is a function of $\tilde{n}$ denoted by $f(\tilde{n})$. The redundancy of $\mathcal{C}'$ is $\lfloor \frac{d_h-1}{2} \rfloor \log(\tilde{n}) + \mathcal{O}(1)$. We construct the code $\mathcal{C}_2(n, k, d_h, d_m)$ by choosing $k = \mathcal{F}(n, d_m) + 1$ and using $\mathcal{C}'$ as $\mathcal{C}_H$. The choice of $\tilde{n}$ is determined in a way that $f(\tilde{n}) = n' = n - k - \ell(d_m) - 2d_m$ is satisfied. Thus, the total redundancy of this construction is upper bounded by

$$\mathcal{F}(n, d_m) + \ell(d_m) + 2d_m + 1 + \left\lfloor \frac{d_h - 1}{2} \right\rfloor \log n + \mathcal{O}(1)$$
$$= \left\lfloor \frac{d_h + 1}{2} \right\rfloor \log n + (d_m - 1) \log\log n + \mathcal{O}(d_m \log d_m).$$

■

Lastly, we note that Construction II improves upon Construction 3 from [34], which solves the problem of $(d_h, d_m = 1)$-MU codes but requires $\mathcal{O}(\sqrt{n})$ redundancy bits.

VII. MU Codes With Edit Distance

In this section we turn to another extension of MU codes which imposes a minimum *edit distance* between prefixes and suffixes as well as on the code. This extension is motivated by several works such as [33] which report on deletion errors during the synthesis process of DNA strands.

The *edit distance*, denoted by $d_E(\mathbf{a}, \mathbf{b})$, of two words $\mathbf{a}, \mathbf{b}$ is the minimum number of insertions and deletions that transform $\mathbf{a}$ to $\mathbf{b}$. The *minimum edit distance* of a code $\mathcal{C}$ is the minimal d such that for any two distinct words $\mathbf{a}, \mathbf{b} \in \mathcal{C}, d_E(\mathbf{a}, \mathbf{b}) \geq d$. For a word $\mathbf{a}$ we say that $(a_{i_1} a_{i_2} \ldots a_{i_k})$ is a subsequence of $\mathbf{a}$ if $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. A common subsequence of two words $\mathbf{a}, \mathbf{b}$ is a sequence which is a subsequence of $\mathbf{a}$ and $\mathbf{b}$. We say that a sequence $\mathbf{c}$ is a *largest common subsequence (lcs)* of $\mathbf{a}, \mathbf{b}$ if there is no common subsequence of $\mathbf{a}$ and $\mathbf{b}$ with length greater than the length of $\mathbf{c}$. Note that every lcs of $\mathbf{a}, \mathbf{b}$ has the same length, which will be denoted by $\ell(\mathbf{a}, \mathbf{b})$.

Next, we list several commonly known claims that will be helpful in the following proofs of this section. The claims' proofs appear in Appendix D for completeness.

Claim 27: For $\boldsymbol{a} \in \Sigma_q^n, \boldsymbol{b} \in \Sigma_q^m$: $d_E(\boldsymbol{a}, \boldsymbol{b}) = n + m - 2\ell(\boldsymbol{a}, \boldsymbol{b})$.

Claim 28: For $\boldsymbol{a} \in \Sigma_q^n, \boldsymbol{b} \in \Sigma_q^n, \boldsymbol{c} \in \Sigma_q^m, \boldsymbol{d} \in \Sigma_q^m$: $d_E(\boldsymbol{ac}, \boldsymbol{bd}) \geq \max\{d_E(\boldsymbol{a}, \boldsymbol{b}), d_E(\boldsymbol{c}, \boldsymbol{d})\}$.

Claim 29: For $\boldsymbol{a} \in \Sigma_q^n, \boldsymbol{b} \in \Sigma_q^n, \boldsymbol{c} \in \Sigma_q^m, d_E(\boldsymbol{ac}, \boldsymbol{b}) \geq d_E(\boldsymbol{a}, \boldsymbol{b})/2$.

Claim 30: For $\boldsymbol{a} = 0^n, \boldsymbol{b} \in \Sigma_q^n, d_E(\boldsymbol{a}, \boldsymbol{b}) = 2w_H(\boldsymbol{b})$.

We are now ready to present the definition of MU codes with edit distance.

*Definition 31: A code $\mathcal{C}$ is called a (d_e, d_m)-**EMU code** if*

1) *the minimum edit distance of the code is d_e,*
2) *for every two not necessarily distinct words $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$, and $i, j \in [n-1]$, if $i, j \in [d_m, n-d_m]$: $d_E(\boldsymbol{a}_1^i, \boldsymbol{b}_{n-j+1}^n) \geq d_m$, otherwise $d_E(\boldsymbol{a}_1^i, \boldsymbol{b}_{n-j+1}^n) \geq \min\{i, j, n-i, n-j\}$.*

The second condition in Definition 31 is different from the MU constraints we introduced so far since now we require large distance of suffixes and prefixes of different lengths, while in Section VI we required large hamming distance between prefixes and suffixes of the same length. This choice of the constraint will assure that suffixes and prefixes of the addresses will not get confused even if they experienced deletions and insertions.

We set $A_{EMU}(n, q, d_e, d_m)$ to be the largest cardinality of any (d_e, d_m)-EMU code over Σ_q^n, and $E(n, q, d)$ is the largest cardinality of a code over Σ_q^n with minimum edit distance d. The following is an upper bound on the value $A_{EMU}(n, q, d_e, d_m)$.

Theorem 32: For all n, q, d_e, d_m,

$$A_{EMU}(n, q, d_e, d_m) \leq \frac{E(n, q, d)}{\lfloor n/d_m \rfloor},$$

where $d = \min\{d_e, d_m\}$.

Proof: Given a (d_e, d_m)-EMU code, $\mathcal{C} \subseteq \Sigma_q^n$, we define $\widehat{\mathcal{C}}$ and $\widehat{\mathbf{a}}, \widehat{\mathbf{b}} \in \widehat{\mathcal{C}}$ similarly to the proof of Theorem 22. The code $\widehat{\mathcal{C}}$ is defined as the code of all cyclic shifts of $\alpha \cdot d_m$ bits of codewords of $\mathcal{C}$. If $\widehat{\mathbf{a}}, \widehat{\mathbf{b}} \in \widehat{\mathcal{C}}$ are of the same shift, their edit distance is at least d_e. Otherwise, we denote $\widehat{\mathbf{a}} = (\mathbf{aa})_{i+1}^{n+i}, \widehat{\mathbf{b}} = (\mathbf{bb})_{j+1}^{n+j}$, where $\mathbf{a}, \mathbf{b} \in \mathcal{C}$, i, j are multiples of d_m, and we assume without loss of generality that $i > j$. The subsequence $\widehat{\mathbf{a}}_{n-i+1}^{n-j}$ is a prefix of $\mathbf{a}$ and $\widehat{\mathbf{b}}_{n-i+1}^{n-j}$ is a suffix of $\mathbf{b}$. Moreover, the length of $\widehat{\mathbf{a}}_{n-i+1}^{n-j}$ is at least d_m, therefore, from the definition of (d_e, d_m)-EMU codes, $d_E(\widehat{\mathbf{a}}_{n-i+1}^{n-j}, \widehat{\mathbf{b}}_{n-i+1}^{n-j}) \geq d_m$. We now apply Claim 28 twice and get that

$$d_E(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \geq d_E(\widehat{\mathbf{a}}_{n-i+1}^{n-j}, \widehat{\mathbf{b}}_{n-i+1}^{n-j}) \geq d_m.$$

We showed that $d_E(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \geq \min\{d_e, d_m\}$, hence all cyclic shifts by $\lfloor n/d_m \rfloor$ symbols of codewords from $\mathcal{C}$ are distinct and the size of $\widehat{\mathcal{C}}$ is $\lfloor n/d_m \rfloor \cdot |\mathcal{C}|$. Moreover, $\widehat{\mathcal{C}}$ is a code with minimum edit distance $d = \min\{d_e, d_m\}$ and we summarize

$$|\widehat{\mathcal{C}}| = \lfloor n/d_m \rfloor \cdot |\mathcal{C}| \leq E(n, q, d),$$

where $d = \min\{d_h, d_e\}$ and the theorem follows directly. ■

In [19], it was shown that

$$E(n, q, 4) \leq \frac{q^n - q}{(q-1)n}$$

which aligns with the asymptotic upper bounds by [21] and [30] $E(n, q, 4) \lesssim \frac{q^n}{(q-1)n}$. Theorem 32 therefore implies that the minimum redundancy when $\min\{d_e, d_m\} = 4$ is at least $2\log_q n + \Theta(1)$.

The following lemma is given without a proof as it shares similar ideas with the proof of Theorem 34 that follows.

Lemma 33: The code $\mathcal{C}_2(n, k, 1, d_m)$ is a $(2, d_m)$-EMU code.

Next, we slightly modify Construction II to allow values of d_e greater than 2.

Construction III: Let n, k be two integers such that $k \geq d_m$ and $n \geq k + 2\,d_m$. Let $n' = n - k - 2d_m$ and $\mathcal{C}_E$ be a (d_m, k)-WWL code of length n' with minimum edit distance d_e. The code $\mathcal{C}_3(n, k, d_e, d_m) \subseteq \{0, 1\}^n$ is defined as follows,

$$\mathcal{C}_3(n, k, d_e, d_m) = \{0^k 1^{d_m} \boldsymbol{c} 1^{d_m} |\ \boldsymbol{c} \in \mathcal{C}_E\}.$$

The correctness of Construction III is proved in the next theorem.

Theorem 34: The code $\mathcal{C}_3(n, k, d_e, d_m)$ is a (d_e, d_m)-EMU code.

Proof: We use the notation $\mathcal{C} = \mathcal{C}_3(n, k, d_e, d_m)$ for simplicity. Any two words $\mathbf{a}, \mathbf{b} \in \mathcal{C}$ satisfy $d_E(\mathbf{a}_{k+d_m+1}^{n-d_m}, \mathbf{b}_{k+d_m+1}^{n-d_m}) \geq d_E$. Applying Claim 28 twice gives us $d_E(\mathbf{a}, \mathbf{b}) \geq d_E$.

We denote $\mathbf{a}_1^i = \mathbf{x}, \mathbf{b}_{n-j+1}^n = \mathbf{y}$. Notice that a word $\mathbf{b} \in \mathcal{C}$ has the following structure $\mathbf{b}_{d_m+1}^n = 0^{k-d_m}1^{d_m}\mathbf{b}_{k+d_m+1}^{n-d_m}1^{d_m}$ such that $\mathbf{b}_{k+d_m+1}^{n-d_m}$ is a (d_m, k)-WWL vector. Therefore, according to Claim 24, $\mathbf{b}_{d_m+1}^n$ is also a (d_m, k)-WWL vector, and since $\mathbf{y}$ is a subsequence of it, $\mathbf{y}$ is a (d_m, k)-WWL vector as well. The following cases are considered,

(1) $j \in [n-d_m],\ i = j$: in that case we show a stronger property of $\mathbf{x}, \mathbf{y}$ which claims that $d_E(\mathbf{x}, \mathbf{y}) \geq 2\min\{d_m, i, j\}$. If $i = j \leq k$ then $\mathbf{x} = 0^i$ and $\omega_H(\mathbf{y}) \geq \min\{d_m, i, j\}$ since $\mathbf{y}$ ends with $\min\{d_m, i, j\}$ 1s. From Claim 30 we get $d_E(\mathbf{x}, \mathbf{y}) \geq 2\min\{d_m, i, j\}$. If $i = j > k$, $\mathbf{x}_1^k = 0^k$ and $\omega_H(\mathbf{y}_1^k) \geq d_m$ as $\mathbf{y}$ is a (d_m, k)-WWL vector. Claim 30 yields $d_E(\mathbf{x}_1^k, \mathbf{y}_1^k) \geq 2d_m$ and $d_E(\mathbf{x}_1^k\mathbf{x}_{k+1}^i, \mathbf{y}_1^k\mathbf{y}_{k+1}^i) = d_E(\mathbf{x}, \mathbf{y}) \geq \max\{2d_m, \omega_H(\mathbf{x}_{k+1}^i, \mathbf{y}_{k+1}^i)\} \geq 2d_m$ according to Claim 28.

(2) $j \in [n-d_m],\ i \neq j$ we assume that $i > j$. $\mathbf{x}_1^j = \mathbf{a}_1^j$ and $\mathbf{y} = \mathbf{b}_{n-j+1}^n$, hence $d_E(\mathbf{x}_1^j, \mathbf{y}) \geq 2\min\{d_m, i, j\}$ following the previous case. Claim 29 implies $d_E(\mathbf{x}_1^j\mathbf{x}_{j+1}^i, \mathbf{y}) = d_E(\mathbf{x}, \mathbf{y}) \geq 2\min\{d_m, i, j\}/2 = \min\{d_m, i, j\}$. The proof for $j > i$ is similar.

(3) $j \in [n-d_m+1, n-1]$: $\mathbf{y} = \mathbf{b}_{n-j+1}^n = 0^{k-(n-j)}1^{d_m}\mathbf{b}_{k+d_m+1}^{n-d_m}1^{d_m}$. Since $n-j < d_m$ and $\mathbf{b}_{k+d_m+1}^{n-d_m}$ is a (d_m, k)-WWL vector, we deduce that $\mathbf{y}$ is a $(n-j, k)$-WWL vector. In the case of $i \geq k$, following a similar path to the proof when $j \in [n-d_m]$ leads us to $d_E(\mathbf{x}, \mathbf{y}) \geq n-j \geq \min\{i, j, n-i, n-j\}$. If $i < k$ then $j - i > d_m$ and hence $d_E(\mathbf{x}, \mathbf{y}) \geq d_m > n-j$. ■

Even though Construction III provides a construction of (d_e, d_m)-EMU codes for all d_e and d_m it heavily depends on the existence of codes with edit distance. The knowledge of codes with large minimum edit distance is quiet limited, in the sense that there exist codes with rate 1, however their structure is complex and there is no explicit expression for their redundancy [9]. Hence, for the rest of this section we focus on the case of edit distance four, i.e. codes correcting a single deletion or insertion.

There exists an explicit efficient method to construct a $(d_m, \mathcal{F}(n, d_m)+1)$-WWL codes with minimum edit distance 4, which will be used as the code $\mathcal{C}_E$ in Construction III. For this, we use Algorithm 2 and the well known Varshamov Tenengolts (VT) codes with edit distance four in their systematic version [1], [31]. Namely, the VT code is defined for all n and $b \in [n+1]$ by

$$VT(b) = \{\mathbf{a} = (a_1, \dots, a_n) \in \Sigma^n \mid \Sigma_{i=1}^n i \cdot a_i \equiv b \bmod (n+1)\}.$$

The systematic version of a VT code converts any vector of length $n' = n - \lceil \log(n+1) \rceil$ to a VT vector of length n by adding $\lceil \log(n+1) \rceil$ redundancy bits in locations corresponding to powers of 2 [1]. Any integer $i \in [0, n]$ can be represented by a sum of a subset of those indices (the subset corresponding to its binary representation). We choose to represent in those indices the integer that guarantees the fulfillment of the VT constraint by the resulting vector. The complexity of the encoding and decoding of this method is linear, we use it to achieve the following result.

Theorem 35: There exists a construction of $(d_m, \mathcal{F}(n, d_m)+1)$-WWL code of length n, with minimum edit distance 4, redundancy $\log n + \mathcal{O}(d_m)$ and linear time and space encoding and decoding complexities.

Proof Sketch: To construct the code $\mathcal{C}$ which is a $(d_m, \mathcal{F}(n, d_m)+1)$-WWL code with minimum edit distance 4, we start with a $(d_m, \mathcal{F}(n, d_m))$-WWL code $\mathcal{C}_{WWL}$ of length n', where $n' + 2d_m + \lceil \log(n' + 2d_m) \rceil = n$. An efficient algorithm for encoding and decoding of such a code was presented in Lemma 19. We then define

$$\mathcal{C}_{EWWL} = \{\mathbf{a}_1^{i_1}1^{d_m}\mathbf{a}_{i_1+1}^{i_2}1^{d_m}\mathbf{a}_{i_2+1}^{n'} | \mathbf{a} \in \mathcal{C}_{WWL}\}$$

where $i_1 \leq i_2$ and the choice of their values will be explained later. The extension to $\mathcal{C}_{EWWL}$ is aimed to ensure that the final resulting code satisfies the $(d_m, \mathcal{F}(n, d_m)+1)$-WWL constraint. It is readily verified that the code $\mathcal{C}_{EWWL}$ is also a $(d_m, \mathcal{F}(n, d_m))$-WWL code. We now apply the systematic VT code on $\mathcal{C}_{EWWL}$ to get the code $\mathcal{C}$ with minimum edit distance 4. The length of $\mathcal{C}_{EWWL}$ is $n' + 2d_m$, hence the length of $\mathcal{C}$ is $n' + 2d_m + \lceil \log(n' + 2d_m) \rceil = n$. We are only left with showing that $\mathcal{C}$ is a $(d_m, \mathcal{F}(n, d_m)+1)$-WWL code.

Recall that the redundancy bits are located in indices which are powers of 2. As a result, for any $\mathbf{a} \in \mathcal{C}$, $\mathbf{a}_{2\lceil \log n \rceil+1}^n$ does not include a window of length $\mathcal{F}(n, d_m) = \log n + o(\log n)$ that contains more than one redundancy bit. Combining it with the fact that $\mathcal{C}_{EWWL}$ is a $(d_m, \mathcal{F}(n, d_m))$-WWL code we get that $\mathbf{a}_{2\lceil \log n \rceil+1}^n$ is a $(d_m, \mathcal{F}(n, d_m)+1)$-WWL vector.

Lastly, we can choose i_1 and i_2 when constructing the code $\mathcal{C}_{EWWL}$ such that a vector $\mathbf{a} \in \mathcal{C}$ satisfies $\mathbf{a}_{\lceil \log n \rceil - d_m}^{\lceil \log n \rceil} = 1^{d_m}$ and $\mathbf{a}_{2\lceil \log n \rceil - d_m}^{2\lceil \log n \rceil} = 1^{d_m}$, or in other words $\mathbf{a} = \mathbf{x}1^{d_m}\mathbf{y}1^{d_m}\mathbf{z}$ where $\mathbf{x}, \mathbf{y}$ are of length $\lceil \log n \rceil - d_m$ and $\mathbf{z}$ of length $n - 2\lceil \log n \rceil$. In that case, we showed that $\mathbf{z}$ is a $(d_m, \mathcal{F}(n, d_m)+1)$-WWL vector and since the length of $\mathbf{x}, \mathbf{y}$ is smaller than $\mathcal{F}(n, d_m)$ they are also $(d_m, \mathcal{F}(n, d_m)+1)$-WWL vectors. Using Claim 24 we conclude that $\mathbf{a} \in \mathcal{C}$ is a $(d_m, \mathcal{F}(n, d_m)+1)$-WWL vector. ■

We use the code $\mathcal{C}$ from the proof above as $\mathcal{C}_E$ in Construction III and deduce the following corollary.

Theorem 36: There exists a construction of $(4, d_m)$-EMU codes with redundancy $2\log n + (d_m - 1)\log\log n + \mathcal{O}(d_m)$ and linear time and space complexity.

To summarize, in this section we first introduced, in Theorem 32, a lower bound on the redundancy of (d_e, d_m)-EMU codes. We then presented a general structure of a (d_e, d_m)-EMU code in Construction III, and used this structure to obtain an explicit construction, with efficient encoder and decoder, when $d_e = 4$. For this case, the construction is $(d_m - 1)\log\log n + \Theta(1)$ redundancy bits away from the lower bound in Theorem 32.

VIII. Polarity Balanced MU Codes

In this section we study yet another extension of MU codes. Under this setup we seek the codes to be *polarity balanced*. A binary word of length n, when n is even, is said to be polarity balanced, or *p-balanced* for short if its Hamming weight is $n/2$. It is well known that the number of p-balanced words is $\binom{n}{n/2} \approx \frac{2^{n+1}}{\sqrt{2\pi n}}$. For the extension of $q > 2$, when q is even, we follow the polarity balanced definition from [32] and say that a code $\mathcal{C} \subseteq \Sigma_q^n$ is p-balanced if for any $\mathbf{a} \in \mathcal{C}$ the number of positions i such that $a_i \in [0, \frac{q}{2}-1]$ is $n/2$. Hence, the number of q-ary p-balanced words is $\binom{n}{n/2}(q/2)^n \approx \frac{2q^n}{\sqrt{2\pi n}}$. For the rest of this section we assume that n and q are even. A code $\mathcal{C} \subseteq \Sigma_q^n$ is said to be a *p-balanced MU code* if $\mathcal{C}$ is p-balanced and is also an MU code. Let $A_{BMU}(n,q)$ denote the maximum cardinality of p-balanced MU codes.

Theorem 37: For all n, q, $A_{BMU}(n,q) \leq \frac{\binom{n}{n/2}(\frac{q}{2})^n}{n} \approx \frac{2q^n}{n\sqrt{2\pi n}}$. In particular, the redundancy of p-balanced MU codes is at least $1.5\log n + \mathcal{O}(1)$.

Proof: Assume that $\mathcal{C}$ is a p-balanced MU code and let $\widehat{\mathcal{C}}$ be the following code.

$$\widehat{\mathcal{C}} = \{(\mathbf{aa})_i^{i+n-1} \mid \mathbf{a} \in \mathcal{C}, i \in [1,n]\}.$$

That is, $\widehat{\mathcal{C}}$ is the code of all cyclic shifts of words from $\mathcal{C}$. The code $\mathcal{C}$ is p-balanced and hence $\widehat{\mathcal{C}}$ is p-balanced as well. Furthermore, since the code $\mathcal{C}$ is an MU code, it is easy to show that all cyclic shifts of its codewords are distinct, hence the cardinality of $\widehat{\mathcal{C}}$ is $n \cdot |\mathcal{C}|$, and we obtain $|\widehat{\mathcal{C}}| = n \cdot |\mathcal{C}| \leq \binom{n}{n/2}(q/2)^n$. ■

Next we present a construction of binary p-balanced MU codes.

Construction IV: Let n, k be two integers such that $1 \leq k < n$. The code $\mathcal{C}_4(n,k) \subseteq \{0,1\}^n$ is defined as follows,

$$\mathcal{C}_4(n,k) = \{0^k 1\boldsymbol{c}1 \mid w_H(\boldsymbol{c}) = \frac{n}{2} - 2, \quad \boldsymbol{c} \text{ has no zero run of length } k\}.$$

The correctness and redundancy result of Construction IV are stated in the next theorem. It follows from the proof that the redundancy of the maximal $\mathcal{C}_4(n,k)$ is $1.5\log n + \mathcal{O}(1)$ and from Theorem 37, this result is optimal up to a constant number of redundancy bits.

Theorem 38: The code $\mathcal{C}_4(n,k)$ is a binary p-balanced MU code, and for an integer $k = \log n + a$, $|\mathcal{C}_4(n, \log n + a)| \gtrsim C\frac{2^n}{n\sqrt{n}}$, where $C = \frac{2^a - 1}{2^{2a+1}\sqrt{2\pi}}$.

Proof: The cardinality of $\mathcal{C}_4(n,k)$ is the number of binary words of length $n' = n-k-2$ which consist of $n/2-2$ ones and do not contain a zero run of length k. To lower bound the number of these words we count all words of length n' and weight $n/2-2$, $\binom{n'}{n/2-2}$, and reduce an upper bound on the number of such words that contain a zero run of length k. For any word $\mathbf{a} \in \Sigma^{n'}$ of weight $n/2-2$ with zero run of length k there exists an index $i \in [1, n'-k+1]$ such that $\mathbf{a}_i^{i+k-1} = 0^k$ and the remaining $n'-k$ bits consist of exactly $n/2-2$ 1s. There are $n'-k+1$ possibilities for i, and for any fixed i, there are $\binom{n'-k}{n/2-2}$ possibilities for the remaining $n'-k$ symbols. Therefore, the number of vectors of weight $n/2-2$ that have a zero run of length k is upper bounded by $(n'-k+1)\binom{n'-k}{n/2-2}$. Based on this observation we show that $|\mathcal{C}_4(n,k)| \gtrsim \frac{2^n}{n\sqrt{n}}\frac{2^a-1}{2^{2a+1}\sqrt{2\pi}}$. The technical details of this step appear in Appendix E. ■

The extension of Construction IV to the non-binary case is straightforward, since every binary symbol can store $q/2$ values and by that a binary p-balanced MU code of size X can be translated to a q-ary p-balanced MU code of size $X \cdot \left(\frac{q}{2}\right)^n$. Applying this approach on Construction IV yields redundancy of $1.5\log_q n + \mathcal{O}(1)$. This meets the result from [34] where a p-balanced MU code over the alphabet $\{A, C, G, T\}$ was suggested, with redundancy of $1.5\log_4(n) + \mathcal{O}(1)$ symbols. However, our construction is also applicable for the binary case as opposed to the construction in [34].

Lastly, we discuss efficient implementation of Construction IV. For a vector $\mathbf{a}$ we denote by $\overline{\mathbf{a}}$ its logical complement. Knuth [18] presented an efficient (linear complexity) algorithm to construct p-balanced words. Knuth's algorithm is based on the observation that for every binary vector $\mathbf{a}$ there exists an index $i \in [1,n]$ such that the vector $\overline{\mathbf{a}}_1^i\mathbf{a}_{i+1}^n$ is p-balanced. To convert an arbitrary vector $\mathbf{a}$ to a p-balanced vector, one can store the p-balanced vector $\overline{\mathbf{a}}_1^i\mathbf{a}_{i+1}^n$, and append a binary p-balanced representation of the index i. The total redundancy of this method is $\log n + \log\log n + o(\log\log n)$. We extend Knuth's method to provide an efficient construction of p-balanced MU codes with linear encoding and decoding complexity and redundancy $2\log n + \log\log n + o(\log\log n)$.

Theorem 39: There exists a construction of p-balanced MU codes with $2\log n + \log\log n + o(\log\log n)$ redundancy bits and linear time and space complexity.

Proof: We describe an algorithm for efficiently encoding words of length n, that agrees with the structure presented in Construction IV. We assume for simplicity that $\log n, \log\log n, \log\log\log n$ are integers and we denote by $\#_1(\mathbf{a}), \#_0(\mathbf{a})$ the number of ones and zeros in $\mathbf{a}$, respectively. We start with a vector x of length

$$n' = n - 2\log n - \log\log n - 2\log\log\log n - 14$$

and apply Algorithm 3.

After Step 1, $\mathbf{x}$ has no zero runs of length $\log n + 1$. We say that a vector $\mathbf{a}$ is the *integral* vector of a vector $\mathbf{b}$ if for all $1 \leq i \leq n$, $\mathbf{a}_i = \sum_{j=1}^{i}\mathbf{b}_j$. It can be shown that if a vector does not contain a zero run of length ℓ, then its integral vector does not contain a zero or one run of length $\ell+1$. According to Step 2, $\mathbf{v}$ is the integral vector of $\mathbf{x}$, therefore, it does not contain a zero or one run of length $\log n + 2$. Then, we add in Step 3 the required prefix $0^{\log n+3}1$. We are now left with balancing the vector $\mathbf{v}$. For that purpose, we use Knuth's Algorithm with some adaptations that ensure the overall structure of the output vector remains as required by Construction IV, i.e., with a prefix $0^{\log n+3}1$, followed by a sequence with no zero run of length $\log n + 3$, and ends with 1. As in Knuth's algorithm, we first find an index i in $\mathbf{v}$ such that $\mathbf{v}_1^i\overline{\mathbf{v}}_{i+1}^{n'}$ is p-balanced. We consider the following cases in Step 4:

(1) $i > \log n + 4$: in Step 8 we set $\mathbf{w} = \mathbf{v}10$ and in Step 10, $i > \log n + 4$ is still satisfied. We then set $\mathbf{y}$ to $\mathbf{w}_1^i 1\overline{\mathbf{w}}_{i+1}^{n'}$. Since $\mathbf{v}$ originally did not have a zeros or ones run of length $\log n + 2$ other than its prefix, $\mathbf{w}$ does not contain a run of length $\log n + 3$ and $\mathbf{y} = \mathbf{w}_2^i 1\overline{\mathbf{w}}_{i+1}^{n'}$ does not contain a zero run of length $\log n + 3$. Similarly

Algorithm 3 Extended Knuth's Algorithm for p-balanced MU codes

Input: $\mathbf{x} \in \Sigma^{n'}$
Output: p-balanced $\mathbf{y} \in \Sigma^n$, $\mathbf{y} = 0^{\log n+3}1\mathbf{y'}1$ where $\mathbf{y'}$ does not contain a run of zeros of length $\log n + 3$.

```
1: Execute Algorithm 1 to remove zero runs of length log n+1 from x
2: Let v ∈ Σ^{n'}, v_i = Σ_{j=1}^{i} x_j
3: v = 0^{log n+3} 1v
4: If v is p-balanced set i = 0, otherwise find an index i such that v_1^i v̄_{i+1}^{n'} is p-balanced
5: if 0.5 log n + 2 < i ≤ log n + 4 then
6:     w = v_1^{log n+4} v̄_{log n+5}^{n'} 01
7: else
8:     w = v10
9: end if
10: If w is p-balanced set i = 0, otherwise find an index i such that w_1^i w̄_{i+1}^{n'} is p-balanced
11: if i > log n + 4 then
12:     p(i): p-balanced binary representation of i with log n + log log n + 2 log log log n + 2 bits, such that it does not contain a zero run of length log n
13:     y = w_1^i 1 w̄_{i+1}^{n'} 1 p(i) 0001
14: else (i ≤ 0.5 log n + 2)
15:     ℓ = #_0(w) − #_1(w)
16:     y = w1^ℓ z11 such that z11 is p-balanced and y ∈ Σ^n
17: end if
```

to Knuth's algorithm, we also append a p-balanced binary representation of i. Such a representation is available with $\log n + \log\log n + 2\log\log\log n$ bits. Since we additionally require it to not include a zero run of length $\log n + 3$ we insert 10 in its $\log n$'th position to get $\mathbf{p(i)}$. The returned vector is appended with 0001 for balancing purposes.

(2) $i < \log n + 4, i \neq 0$: here, we do not simply apply the same approach as in the previous case because we want to guarantee $0^{\log n+3}1$ is a prefix of $\mathbf{y}$. We denote by $\#_1(\mathbf{a}), \#_0(\mathbf{a})$ the number of ones and zeros in $\mathbf{a}$, respectively. $\mathbf{v}_1^i\overline{\mathbf{v}}_{i+1}^{n'}$ is p-balanced, hence

$$\#_0(\mathbf{v}_1^i\overline{\mathbf{v}}_{i+1}^{n'}) = \#_1(\mathbf{v}_1^i\overline{\mathbf{v}}_{i+1}^{n'}).$$

Since

$$\#_0(\mathbf{v}_1^i\overline{\mathbf{v}}_{i+1}^{n'}) = i + \#_1(\mathbf{v})$$

and

$$\#_1(\mathbf{v}_1^i\overline{\mathbf{v}}_{i+1}^{n'}) = \#_0(\mathbf{v}) - i$$

we have that

$$2i = \#_0(\mathbf{v}) - \#_1(\mathbf{v}). \tag{13}$$

(a) If $i \leq 0.5\log n + 2$, then $\mathbf{w} = \mathbf{v}10$ and in Step 10 i remains the same. In Step 16,

$$\ell = \#_0(\mathbf{w}) - \#_1(\mathbf{w}) \leq \log n + 4$$

and therefore we balance $\mathbf{y}$ by simply appending the sequence $1^\ell\mathbf{z}11$.

(b) If $0.5\log n+1 < i < \log n+4$ then ℓ might exceed our desirable redundancy number of bits. Our alternative solution is to set $\mathbf{w}$ in Step 6 to $\mathbf{v}_1^{\log n+4}\overline{\mathbf{v}}_{\log n+5}^{n'}01$. After Step 6

$$\#_0(\mathbf{w}) = \#_0(\mathbf{v}_1^{\log n+4}\overline{\mathbf{v}}_{\log n+5}^{n'}01) = \log n + 3 + \#_1(\mathbf{v}),$$
$$\#_1(\mathbf{w}) = \#_1(\mathbf{v}_1^{\log n+4}\overline{\mathbf{v}}_{\log n+5}^{n'}01) = \#_0(\mathbf{v}) - \log n - 1$$

and we have that

$$\begin{aligned}\#_0(\mathbf{w}) - \#_1(\mathbf{w}) &= \log n + 3 + \#_1(\mathbf{v}) \\ &\quad - (\#_0(\mathbf{v}) - \log n - 1) \\ &= 2\log n + 4 - (\#_0(\mathbf{v}) - \#_1(\mathbf{v})) \\ &\overset{Eq.\ (13)}{=} 2\log n + 4 - 2i \\ &< \log n + 2.\end{aligned}$$

Therefore, after Step 10, similarly to Equation (13),

$$2i = \#_0(\mathbf{w}) - \#_1(\mathbf{w}) < \log n + 3$$

and the algorithm follows the path as in case (a).

(3) $i = \log n+4$ or $i = 0$: in both cases $\mathbf{w}$ is p-balanced after the if clause in Step 5. In Step 10 $i = 0$, the if statement in Step 11 is not satisfied and in Step 16 $\ell = 0$. We append some p-balanced suffix of the form $\mathbf{z}11$ and of suitable length such that $\mathbf{y} = \mathbf{w}1^\ell\mathbf{z}11$ is a p-balanced length-n vector as expected.

The vector $\mathbf{y}$ is uniquely decodable in the following manner. By looking at the two rightmost bits we can detect whether the if statement in Step 11 was satisfied and reconstruct $\mathbf{w}$. Then, again, we look at the two rightmost bits to detect whether the if statement in Step 5 was satisfied and reconstruct $\mathbf{v}$. Finally, $\mathbf{x}$ is derived from $\mathbf{v}$ by removing the prefix 0^k1, and computing the differences vector of $\mathbf{v}$. ■

To conclude, we showed in Theorem 37 that the problem of p-balanced MU codes can be solved with at least $1.5\log n + \Theta(1)$ redundancy bits. Construction IV and Theorem 38 establish the existence and provide the structure of a p-balanced MU code with such a redundancy. Lastly, in Theorem 39 we present an explicit construction with efficient encoding and decoding algorithms, with $2\log n + \Theta(1)$ redundancy bits.

IX. Other Related Families of Codes

In this section we present two families of codes which are closely related to MU codes. Namely, *comma-free codes* [14] and *prefixed synchronized codes* [13], [24]. We discuss these codes and their connection with MU codes.

A. Comma-Free Codes

Comma-free codes were studied in [14], motivated by the problem of word synchronization for block codes. A code $\mathcal{C} \subseteq \Sigma_q^n$ is a comma-free code if for any two not necessarily distinct vectors $\mathbf{a}, \mathbf{b} \in \mathcal{C}$ and $i \in [2, n]$, $(\mathbf{ab})_i^{n+i-1} \notin \mathcal{C}$. We denote by

$A_{CF}(n,q)$ the maximal cardinality of a comma-free code of length n over Σ_q. It was shown in [14] that

$$A_{CF}(n,q) \le \frac{1}{n}\sum_{d|n}\mu(d)q^{n/d},$$

where $\mu(d)$ is the Möbius function and the sum is taken over all divisors of n. Later in [12], an optimal construction of odd length comma-free codes was introduced leading to

$$A_{CF}(n,q) = \frac{1}{n}\sum_{d|n}\mu(d)q^{n/d} \approx \frac{q^n}{n}$$

for odd n.

Note that the comma-free property is weaker than the MU property, therefore an MU-code is a comma-free code but not vice versa; in particular, $A_{MU}(n,q) \le A_{CF}(n,q)$.

An extension to comma-free codes by Levenshtein [22] states that $\mathcal{C} \subseteq \Sigma_q^n$ is a (d,ρ) comma-free code if for any $\mathbf{a},\mathbf{b},\mathbf{c} \in \mathcal{C}$, $i \in [2,n]$, $d_H((\mathbf{ab})_i^{i+n-1},\mathbf{c}) \ge \rho$, and $\mathcal{C}$ has a minimum Hamming distance d.

Levenshtein proved that there exists a construction of (d,ρ)-comma-free codes with redundancy of approximately $\lfloor\frac{d+1}{2}\rfloor\log n + c(\rho)$ bits, where $c(\rho)$ is a constant that depends on ρ. However, the encoding and decoding of such codes are complex. To allow efficient encoding and decoding, Levenshtein suggested in [24] to use constructions based upon cosets of linear codes. But, in this case, it was shown in [3] that the redundancy is at least $\sqrt{\rho n}$. Note that any (d,ρ)-MU code is also a (d,ρ)-comma-free code, so Construction II is a (d,ρ)-comma-free code and we can use the result of Corollary 26 to construct an efficient (d,ρ)-comma-free code with significantly less redundancy:

Corollary 40: There exists a construction of a (d,ρ)-comma-free code with efficient encoding and decoding and $\lfloor\frac{d+1}{2}\rfloor\log n + (\rho-1)\log\log n + o(\log\log n)$ redundancy bits.

For the case of $\rho \ge d$ this construction is $\mathcal{O}(\log\log n)$ away from the optimal possible redundancy according to [22]. The details of the construction are described in the proof of Corollary 26.

B. Prefix Synchronized Codes

For a set $H \subseteq \Sigma_q^m$, a prefix synchronized code $\mathcal{C}_H \subseteq \Sigma_q^n$ is defined to be the set of all words $\mathbf{a} \in \Sigma_q^n$ such that for any $h \in H$, the word $\mathbf{a}h$ contains a word from H only in the first and last m positions [13], [24]. Construction I is in fact a prefix synchronized code with the set $H = \{0^k\}$. Prefix synchronized codes can be defined for any set of prefixes H. Another related problem is discussed in [22], namely, prefix synchronized codes with index ρ. A code $\mathcal{C} \subseteq \Sigma^n$ is said to be prefix synchronized with a set $H \subseteq \Sigma_q^m$, $m \le n$ and index ρ if for any $\mathbf{a} \in \mathcal{C}$, $\mathbf{h} \in H$, $i \in [2,n]$, $d_H((\mathbf{ah})_i^{i+m-1},\mathbf{h}) \ge \rho$. Levenshtein stated in [22] that when n goes to infinity, a lower bound on the redundancy of a prefix synchronized code with index ρ is $\log n + (\rho-1)\log\log n + \log\log\log n$. The next theorem provides a prefix synchronized code which is close to optimal.

Theorem 41: The code $\mathcal{C}_2(n,k,1,d_m)$ defined in Construction II, is prefix synchronized with index $\rho = d_m$ for the set $H = \{0^k\boldsymbol{u}\}$, where the vector $\boldsymbol{u}$ is defined in Eq. (12).

Hence, by Corollary 26 we provide an efficient construction to this problem with only $o(\log\log n)$ additional bits of redundancy.

X. Conclusion

In this work we studied MU codes and their extension to MU codes with Hamming distance. For that purpose we looked into two interesting constraints, the k-RLL and the (d,k)-WWL constraints when k is a function of the word's length, n. The results of this study are presented in Table I. We then continued to additional variations of MU codes, that is MU codes with minimum Edit distance and p-balanced MU codes. Similar techniques can be applied to construct p-balanced MU codes together with minimum Hamming distance and thereby satisfy three of the constraints listed in [34]. The results on the variations of MU codes are summarized in Table II. For each case we first give the lower bound on the redundancy, then the construction that solves this case, and finally the best redundancy we could get with linear encoding and decoding complexity.

Appendix A

Proposition 6: Let $f(n)$, $g(n)$ be functions such that $\lim_{n\to\infty} g(n) = 1$ and $1 \le f(n) \le C$ for a constant C then

$$f(n)^{g(n)} \approx f(n)$$

Proof: For any $0 < \delta < 1$ we choose

$$\delta' = \min\left\{-\frac{\log(1-\delta)}{\log C}, \frac{\log(1+\delta)}{\log C}\right\} > 0$$

such that $1-\delta \le C^{-\delta'}$ and $C^{\delta'} \le 1+\delta$ are satisfied. There exists N' such that for every $n \ge N'$

$$1-\delta' \le g(n) \le 1+\delta'.$$

Therefore, for every $n \ge N'$

$$C^{-\delta'} \le f(n)^{-\delta'} \le \frac{f(n)^{g(n)}}{f(n)} \le f(n)^{\delta'} \le C^{\delta'}$$

and from the choice of δ' we get that for every $n \ge N'$

$$1-\delta \le \frac{f(n)^{g(n)}}{f(n)} \le 1+\delta$$

hence

$$\lim_{n\to\infty}\frac{f(n)^{g(n)}}{f(n)} = 1.$$

■

Appendix B

Lemma 14: For $k = \lceil\log_q n\rceil + z, z \in \mathbb{Z}$,

$$2^{n' E_{k,q}} \approx \frac{q^n}{n}\cdot\frac{q^{\Delta_n - z-2}}{e^{(q-1)q^{\Delta_n - z-1}}},$$

where $\Delta_n = \log_q n - \lceil\log_q n\rceil$.

TABLE II
REDUNDANCY SUMMARY FOR BINARY MU CODES

Property	MU	(d_h, d_m)- MU	$(4, d_m)$- EMU	p-balanced MU
Lower bound	$\log n + \log(e)$	$\lfloor \frac{d+1}{2} \rfloor \log n - \log d_m + \mathcal{O}(1)$	$2\log n - \log d_m + \mathcal{O}(1)$	$1.5\log n + \log\sqrt{2\pi} - 1$
Construction	Construction I	Construction II	Construction III	Construction IV
Efficient upper bound	$\lceil \log n \rceil + 4$	$\lfloor \frac{d_h+1}{2} \rfloor \log n + (d_m-1)\log\log n$ $+\mathcal{O}(d_m \log d_m)$	$2\log n + (d_m-1)\log\log n$ $+\mathcal{O}(d_m)$	$2\log n + \log\log n$ $+o(\log\log n)$
Comments		$d = \min\{2d_m, d_h\}$	$d_m \geq 4$	

Proof: From Lemma 7, when $k = \lceil \log_q n \rceil + z$, and for n large enough there exists a constant $C \geq 0$ such that

$$1 \leq 2^{n'(\log q - E_{k,q})} \leq 2^{n'\frac{C}{n}} \leq 2^C.$$

We use Proposition 6 with $f(n) = 2^{n'(\log q - E_{k,q})}$ and $g(n) = \frac{(q-1)\log e q^{-k-2}}{\log q - E_{k,q}}$ to get

$$2^{n'(\log q - E_{k,q})} \approx 2^{n'(q-1)\log e q^{-(k-1)-2}}$$

and conclude that

$$\begin{aligned} 2^{n'E_{k,q}} &= 2^{n'\log q + n'(E_{k,q} - \log q)} \\ &\approx 2^{n'\log q - n'(q-1)\log e q^{-k-1}} \\ &= q^{n'} e^{-n'(q-1)q^{-k-1}}. \end{aligned}$$

The term $n'(q-1)q^{-k-1}$ satisfies

$$\begin{aligned} &n'(q-1)q^{-k-1} \\ &\quad = (n - (\lceil \log_q n \rceil + z) - 2)(q-1)q^{-(\lceil \log_q n \rceil + z) - 1} \\ &\quad = (n - (\lceil \log_q n \rceil + z) - 2)(q-1)\frac{q^{\Delta_n - z - 1}}{n} \\ &\quad = (q-1)q^{\Delta_n - z - 1} + o(1). \end{aligned}$$

Finally, we conclude that

$$\begin{aligned} 2^{n'E_{k,q}} &\approx q^{n'} e^{-(q-1)q^{\Delta_n - z - 1} + o(1)} \\ &\approx \frac{q^n}{n} \frac{q^{\Delta_n - z - 2}}{e^{(q-1)q^{\Delta_n - z - 1}}} \end{aligned}$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$. ∎

Lemma 15: For $z \in \mathbb{Z}$,

$$\begin{aligned} &|\mathcal{C}_1(n, q, \lceil \log_q n \rceil + z)| \\ &\quad \approx \frac{q^n}{n}\left(\frac{q-1}{q}\right)^2 q^{\Delta_n - z - \log_q e(q-1)q^{\Delta_n - z - 1}} \end{aligned}$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$.

Proof: From (10) we have

$$1 \leq \frac{a_q(n', k)}{2^{n'E_{k,q}}} \leq 1 + \frac{2q^{n' - \lceil k/2 \rceil}}{2^{n'E_{k,q}}}.$$

By using Lemma 14 we get that for $k = \lceil \log_q n \rceil + z$

$$\lim_{n\to\infty} \frac{2q^{n' - \lceil k/2 \rceil}}{2^{n'E_{k,q}}} = \lim_{n\to\infty} \frac{2q^{n' - \lceil k/2 \rceil}}{q^n e^{-q^{\Delta_n - z - 1}}} = 0,$$

and conclude that

$$1 \leq \lim_{n\to\infty} \frac{a_q(n', k)}{2^{n'E_{k,q}}} \leq \lim_{n\to\infty} 1 + \frac{2q^{n' - \lceil k/2 \rceil}}{2^{n'E_{k,q}}} = 1.$$

Hence,

$$a_q(n', k) \approx 2^{n'E_{k,q}}.$$

Recall that $|\mathcal{C}_1(n, q, k)| = (q-1)^2 a_q(n', k)$. Together with Lemma 14 the result follows directly. ∎

Theorem 16:

$$\mathcal{C}_1(n, q) \approx \frac{q^n}{n} \cdot \left(\frac{q-1}{q}\right)^2 q^{F(\Delta_n)} \leq \frac{q^n}{n} \cdot \frac{q-1}{eq},$$

where $\Delta_n = \log_q n - \lceil \log_q n \rceil$ and

$$F(\Delta_n) = \max_{z \in \{-2,-1,0\}} \left\{ \Delta_n - z - \log_q(e)(q-1)q^{\Delta_n - z - 1} \right\}.$$

The inequality is tight when $n \to \infty$ over any subsequence of n that satisfies $\Delta_n = -\log_q(q-1)$.

Completion of the Proof of Theorem 16: In this part, we complete the proof of Theorem 16 by analyzing the function $F(\Delta_n)$. Recall that

$$f(\Delta_n, z) = \Delta_n - z - \log_q e(q-1)q^{\Delta_n - z - 1}$$

and the value of z that achieves the only maximum of $f(\Delta_n, z)$ over the real numbers is $z_0 = \log_q(q-1) - 1 + \Delta_n$. Since we are interested in integers, we next investigate which $z \in \mathbb{Z}$ maximizes $f(\Delta_n, z)$. We consider the following two cases.

(1) $q = 2$: In this case, $z_0 = \Delta_n - 1$, and therefore $-2 < z_0 \leq -1$, so the maximal value is achieved for $z \in \{-2, -1\}$. Note that

$$\frac{\partial f}{\partial \Delta_n} = 1 - (q-1)q^{\Delta_n - z - 1} = 1 - 2^{\Delta_n - z - 1},$$

and

$$\begin{aligned} \left.\frac{\partial f}{\partial \Delta_n}\right|_{z=-1} &= 1 - 2^{\Delta_n} \geq 0, \\ \left.\frac{\partial f}{\partial \Delta_n}\right|_{z=-2} &= 1 - 2^{\Delta_n + 1} \leq 0. \end{aligned}$$

Hence $f(\Delta_n, -1)$ is increasing and $f(\Delta_n, -2)$ is decreasing when $-1 < \Delta_n \leq 0$. Moreover, they meet in $\Delta_n = \log(\ln 2) \approx -0.53$. From the analysis above we summarize that when $q = 2$,

$$F(\Delta_n) = \begin{cases} f(\Delta_n, -2), & for \;\; -1 < \Delta_n \leq \log(\ln 2) \\ f(\Delta_n, -1), & otherwise \end{cases}$$

To understand which Δ_n achieves the maximal $F(\Delta_n)$ we can consider the only two options $\Delta_n \in \{-1, 0\}$ and since we get $f(-1, -2) = f(0, -2) = 1 - \log e$ we can conclude that

$$\mathcal{C}_1(n, q) \lesssim \frac{2^n}{4n} \cdot 2^{1 - \log e} = \frac{2^n}{2en}.$$

Lastly, when $n \to \infty$ over any subsequence of n that satisfies $\lim_{n\to\infty} \Delta_n \in \{-1, 0\}$, we have that

$$\mathcal{C}_1(n, q) \approx \frac{2^n}{2en}.$$

(2) $q > 2$:
For any $q > 2$ and $-1 < \Delta_n \le 0$ we have that $-2 < z_0 < 0$, and so the maximum over $z \in \mathbb{Z}$ is achieved by one of the options $z \in \{-2, -1, 0\}$. Note that

$$\begin{aligned} f(\Delta_n, -1) - f(\Delta_n, -2) &= \Delta_n + 1 - \log_q e(q-1)q^{\Delta_n} \\ &\quad - (\Delta_n + 2 - \log_q e(q-1)q^{\Delta_n+1}) \\ &= -1 + \log_q e(q-1)^2 q^{\Delta_n} \\ &\ge -1 + \frac{(q-1)^2}{q \ln q} > 0, \end{aligned}$$

where the last step holds for any $q > 2$. Therefore, we are only left with maximizing over $z \in \{-1, 0\}$. The functions $f(\Delta_n, -1)$ and $f(\Delta_n, 0)$ meet when $\Delta_n = \delta_0 = -\log_q \frac{(q-1)^2}{q \ln q}$. Since for $q > 2$, $1 \le \frac{(q-1)^2}{q \ln q} < q$, we have $-1 < \delta_0 \le 0$. In addition,

$$\left.\frac{\partial f}{\partial \Delta_n}\right|_{z=0} = 1 - \frac{q-1}{q} \cdot q^{\Delta_n} > 0,$$

hence $f(\Delta_n, 0)$ is increasing for $-1 < \Delta_n \le 0$. Also,

$$\left.\frac{\partial f}{\partial \Delta_n}\right|_{z=-1, \Delta_n=\delta_0} = 1 - (q-1) \cdot q^{\delta_0} = 1 - \frac{q \ln q}{q-1} < 0,$$

so $f(\Delta_n, -1)$ is decreasing at the point $\Delta_n = \delta_0$. We conclude that for $\Delta_n \le \delta_0$, $f(\Delta_n, 0) \le f(\Delta_n, -1)$ and for $\Delta_n \ge \delta_0$, $f(\Delta_n, -1) \le f(\Delta_n, 0)$. That is,

$$F(\Delta_n) = \begin{cases} f(\Delta_n, -1), & for\ -1 < \Delta_n \le \delta_0 \\ f(\Delta_n, 0), & otherwise \end{cases}$$

We next study that values of Δ_n that maximize the function $F(\Delta_n)$. We showed that $f(\Delta_n, 0)$ is increasing so its maximal value is achieved when $\Delta_n = 0$ and it is $F(0) = f(0, 0) = \frac{q-1}{q} - 1$. The maximal value of $f(\Delta_n, -1)$ is achieved when

$$\left.\frac{\partial f}{\partial \Delta_n}\right|_{z=-1} = 0,$$

that is, when $\Delta_n = \delta_1 = -\log_q(q-1)$. Since $-1 \le \delta_1 \le \delta_0$ we get $F(\delta_1) = f(-1, \delta_1) = 1 - \log_q(e(q-1))$ and $F(\delta_1) \ge F(0)$. We are now ready to summarize and conclude that

$$\mathcal{C}_1(n, q) \lesssim \frac{q^n}{n}\left(\frac{q-1}{q}\right)^2 \cdot q^{F(\delta_1)} = \frac{q^n}{n}\frac{q-1}{eq}$$

and when $n \to \infty$ over any subsequence of n that satisfies $\lim_{n\to\infty} \Delta_n \in \{-1, 0\}$ we have

$$\mathcal{C}_1(n, q) \approx \frac{q^n}{n}\frac{q-1}{eq}.$$

■

Appendix C

Lemma 18: Let n, k, d be positive integers such that $d \le k \le n$. Then, there exists a constant $C > 0$ such that for n large enough

$$a_q(n, k, d) \le q^{n - C\frac{(n-2k)k^{d-1}}{q^k}}.$$

Proof: We consider the set $A_q(2k, k, d)^{\lfloor \frac{n}{2k} \rfloor}$, that is, the set of vectors which are a concatenation of $\lfloor \frac{n}{2k} \rfloor$ vectors from $A_q(2k, k, d)$. We then append it with the set of all length-$\langle n \rangle_{2k}$ q-ary vectors. The resulting set of length-n vectors is denoted by $B_q(n, k, d) = A_q(2k, k, d)^{\lfloor \frac{n}{2k} \rfloor} \Sigma_q^{\langle n \rangle_{2k}}$. Note that $A_q(n, k, d) \subseteq B_q(n, k, d)$ and

$$|B_q(n, k, d)| = a_q(2k, k, d)^{\lfloor \frac{n}{2k} \rfloor} q^{\langle n \rangle_{2k}}.$$

Hence,

$$a_q(n, k, d) \le a_q(2k, k, d)^{\lfloor \frac{n}{2k} \rfloor} q^{\langle n \rangle_{2k}}. \tag{14}$$

Let $b(k)$ be the number of vectors of length $2k$ with a subsequence of the form $[1, q-1]^d \mathbf{u} [1, q-1]^d$ where $\mathbf{u}$ is a vector of length k and weight smaller than d and $[1, q-1]^d$ corresponds to a sequence of d non zero symbols. The value of $b(k)$ is given by

$$\begin{aligned} b(k) &= q^{2k-(k+2d)}(q-1)^{2d}(2k - (k+2d) + 1)\sum_{i=0}^{d-1}\binom{k}{i} \\ &= q^{k-2d}(q-1)^{2d}(k - 2d + 1)\sum_{i=0}^{d-1}\binom{k}{i}. \end{aligned}$$

All those length $2k$ vectors are not included in the set $A_q(2k, k, d)$. Therefore,

$$\begin{aligned} a_q(2k, k, d) &\le q^{2k} - b(k) \\ &\le q^{2k} - q^{k-2d}(q-1)^{2d}(k - 2d + 1)\sum_{i=0}^{d-1}\binom{k}{i}. \end{aligned} \tag{15}$$

We denote $B = \sum_{i=0}^{d-1}\binom{k}{i}$. Note that $B = \Theta(k^{d-1})$, when d is fixed and k is arbitrary large. Combining inequalities (14) and (15) we get

$$\begin{aligned} &a_q(n, k, d) \\ &\le \left(q^{2k} - q^{k-2d}(q-1)^{2d}(k - 2d + 1)B\right)^{\lfloor \frac{n}{2k} \rfloor} \cdot q^{\langle n \rangle_{2k}} \\ &= \left(q^{2k}\left(1 - \frac{(k-2d+1)(q-1)^{2d}B}{q^{k+2d}}\right)\right)^{\lfloor \frac{n}{2k} \rfloor} \cdot q^{\langle n \rangle_{2k}} \\ &= q^n\left(1 - \frac{(k-2d+1)(q-1)^{2d}B}{q^{k+2}}\right)^{\lfloor \frac{n}{2k} \rfloor} \\ &\overset{(a)}{\le} q^n\left(e^{-\frac{(k-2d+1)(q-1)^{2d}B}{q^{k+2}}}\right)^{\lfloor \frac{n}{2k} \rfloor} \\ &\le q^{n - \log_q e \frac{(k-2d+1)(q-1)^{2d}B}{q^{k+2}}\left(\frac{n}{2k} - 1\right)} \\ &\overset{(b)}{\le} q^{n - C\frac{(n-2k)k^{d-1}}{q^k}}, \end{aligned}$$

where (a) results from the inequality $1 - x \le e^{-x}$ for all x and (b) since there exists a constant C such that for n large enough the inequality holds. ■

Lemma 19: For all $n' \le n$, given any vector $\boldsymbol{x} \in \Sigma_q^{n'}$ Algorithm 2 outputs a $(d, \mathcal{F}(n,d))$-WWL vector $\boldsymbol{y} \in \Sigma_q^{n'+d}$ such that $\boldsymbol{x}$ can be uniquely reconstructed given $\boldsymbol{y}$. The time and space complexity of the algorithm and its inverse is $\Theta(n)$.

Proof: First, we notice that according to the choice of $\mathcal{F}(n,d)$ and C the length of $\mathbf{y}$ does not change throughout the execution of the algorithm, therefore $\mathbf{y} \in \Sigma_q^{n'+d}$. There exists an index $1 \le t \le n'$ such that the output vector $\mathbf{y}$ satisfies $\mathbf{y} = (\mathbf{y}_1^t, 1^d, \mathbf{y}_{t+d+1}^{n'+d})$, where $\mathbf{y}_1^t$ is the remainder of $\mathbf{x}$ after removing the low weight vectors and $\mathbf{y}_{t+d+1}^{n'+d}$ is the list of pointers of the form $p(i)t(1)\cdots t(d-1)01$ representing the indices of the low weight subvectors and the positions of the ones inside each subvector.

To reconstruct $\mathbf{x}$ we first locate the index t by scanning $\mathbf{y}$ from the right. We read the two rightmost symbols of $\mathbf{y}$, if they are 01 we conclude that the following $\mathcal{F}(n,d) - 2$ symbols are a pointer of the form $p(i)t(1)\cdots t(d-1)01$, we skip them and repeat that process until we encounter with two symbols 11. We then construct the original $\mathbf{x}$ by inserting proper low weight vectors of length $\mathcal{F}(n,d)$ to the remainder part $\mathbf{y}_1^t$. The positions of the vectors we insert, the positions of the non-zeros within them, and the non-zeros symbols, are all determined according to the indices listed in the pointers part in $\mathbf{y}_{t+d+1}^{n'+d}$.

We next show that $\mathbf{y}$ does not contain a vector of length $\mathcal{F}(n,d)$ of weight less than d. The remainder part $\mathbf{y}_1^t$ does not contain such a vector since we removed all low weight vectors within the while loop. Also, the separating part 1^d ensures that there is no vector of length $\mathcal{F}(n,d)$ that originates in $\mathbf{y}_1^t$ and ends in $\mathbf{y}_{t+d+1}^{n'+d}$. Next we contradict the case of low weight vector in the addressing part. Recall that the structure of $\mathbf{y}_{t+d+1}^{n'+d}$ is a concatenation of pointers of the form $p(i)t(1)\cdots t(d-1)01$. Note that the weight of every index $p(i)$ or $t(j)$ is at least 1. Every vector of length $\mathcal{F}(n,d)$ in $\mathbf{y}_{t+d+1}^{n'+d}$ consists at least $d-1$ full indices (counting both p_i and t_js) and an additional one from the appended 01 pairs. Therefore the total weight of such vectors is at least d as required.

Lastly, the algorithm's complexity is $\Theta(n)$ since the complexity of every pointer update $\Theta(\log n)$ and there are at most $n/\log n$ updating operations. ■

Appendix D

Claim 27: For $\boldsymbol{a} \in \Sigma_q^n, \boldsymbol{b} \in \Sigma_q^m$: $d_E(\boldsymbol{a},\boldsymbol{b}) = n + m - 2\ell(\boldsymbol{a},\boldsymbol{b})$.

Proof: We say a series of insertions and deletions that transforms $\mathbf{a}$ to $\mathbf{b}$ is *canonic* if all of its deletions are of symbols from the original $\mathbf{a}$. In other words, there were no new symbols that were inserted and then deleted. To determine the edit distance of $\mathbf{a}$ and $\mathbf{b}$ we are interested in the minimal length of a series that transforms $\mathbf{a}$ to $\mathbf{b}$. We therefore imit our discussion, without loss of generality, to canonic series, because if a series is not canonic there exists a shorter equivalent canonic series to transform $\mathbf{a}$ to $\mathbf{b}$.

Any canonic series is associated with a common subsequence of $\mathbf{a}$ and $\mathbf{b}$ which is received by applying all the deletions in the series on $\mathbf{a}$. We denote this common subsequence by $\mathbf{x} \in \Sigma_q^\ell$. The number of deletions in the initial canonic series is $n - \ell$ and number of insertions is $m - \ell$, so the length of the series is $n + m - 2\ell$. Equivalently, any common subsequence of $\mathbf{a}$ and $\mathbf{b}$, denoted by $\mathbf{x} \in \Sigma_q^\ell$ is associated with a canonic series of length $n+m-2\ell$ which consists deletions of symbols of $\mathbf{a}$ that do not belong to $\mathbf{x}$ followed by insertions of symbols of $\mathbf{b}$ that do not belong to $\mathbf{x}$. Thus, there exists a common subsequence of $\mathbf{a}$ and $\mathbf{b}$ of length ℓ if and only if there exists a canonic series from $\mathbf{a}$ to $\mathbf{b}$ of length $n+m-2\ell$. From the definition of edit distance, the claim follows. ■

Claim 28: For $\boldsymbol{a} \in \Sigma_q^n, \boldsymbol{b} \in \Sigma_q^n, \boldsymbol{c} \in \Sigma_q^m, \boldsymbol{d} \in \Sigma_q^m$: $d_E(\boldsymbol{ac},\boldsymbol{bd}) \ge \max\{d_E(\boldsymbol{a},\boldsymbol{b}), d_E(\boldsymbol{c},\boldsymbol{d})\}$.

Proof: Without loss of generality we assume that $d_E(\mathbf{a},\mathbf{b}) = \max\{d_E(\mathbf{a},\mathbf{b}), d_E(\mathbf{c},\mathbf{d})\}$. We set $\ell(\mathbf{a},\mathbf{b}) = \ell_1, \ell(\mathbf{ac},\mathbf{bd}) = \ell_2$. For any common subsequence $\mathbf{x}$ of $\mathbf{ac}, \mathbf{bd}$ of length $\ell \ge m$, the prefix of length $\ell - m$, $\mathbf{x}_1^{\ell-m}$ is a common subsequence of $\mathbf{a}, \mathbf{b}$ hence $\ell_1 \ge \ell - m$. If we take $\mathbf{x}$ to be an lcs of $\mathbf{ac}, \mathbf{bd}$ we get that $\ell_1 \ge \ell_2 - m$. Combining this with Claim 27 we have

$$\begin{aligned} d_E(\mathbf{ac},\mathbf{bd}) &= 2n + 2m - 2\ell_2 \ge 2n + 2m - 2\ell_1 - 2m \\ &= d_E(\mathbf{a},\mathbf{b}). \end{aligned}$$

■

Claim 29: For $\boldsymbol{a} \in \Sigma_q^n, \boldsymbol{b} \in \Sigma_q^n, \boldsymbol{c} \in \Sigma_q^m, d_E(\boldsymbol{ac},\boldsymbol{b}) \ge d_E(\boldsymbol{a},\boldsymbol{b})/2$.

Proof: We will show that

$$\max\{d_E(\mathbf{a},\mathbf{b}) - m, m\} \le d_E(\mathbf{ac},\mathbf{b}),$$

since every m satisfies $d_E(\mathbf{a},\mathbf{b})/2 \le \max\{d_E(\mathbf{a},\mathbf{b}) - m, m\}$ the claim follows directly. An lcs of $\mathbf{ac}, \mathbf{b}$ is a subsequence of $\mathbf{b}$, therefore $\ell(\mathbf{ac},\mathbf{b}) \le n$ and from Claim 27, $d_E(\mathbf{ac},\mathbf{b}) = n + n + m - 2\ell(\mathbf{ac},\mathbf{b}) \ge 2n + m - 2n = m$.

We set $\ell(\mathbf{a},\mathbf{b}) = \ell_1, \ell(\mathbf{ac},\mathbf{b}) = \ell_2$. Note that since $\mathbf{a}$ and $\mathbf{b}$ are of the same length, $d_E(\mathbf{a},\mathbf{b})$ is even. As in the proof of claim 28, $\ell_1 \ge \ell_2 - m$. Combining it with Claim 27 we have $d_E(\mathbf{ac},\mathbf{b}) = 2n + m - 2\ell_2 \ge 2n + m - 2\ell_1 - 2m = d_E(\mathbf{a},\mathbf{b}) - m$. ■

Claim 30: For $\boldsymbol{a} = 0^n, \boldsymbol{b} \in \Sigma_q^n, d_E(\boldsymbol{a},\boldsymbol{b}) = 2w_H(\boldsymbol{b})$.

Proof: Since $\mathbf{a} = 0^n$ the lcs of $\mathbf{a}, \mathbf{b}$ is $0^{n - w_H(\mathbf{b})}$ and $\ell(\mathbf{a},\mathbf{b}) = n - w_H(\mathbf{b})$. From Claim 27 we get $d_E(\mathbf{a},\mathbf{b}) = 2w_H(\mathbf{b})$. ■

Appendix E

Theorem 38: The code $\mathcal{C}_4(n,k)$ is a p-balanced MU code, and for an integer $k = \log n + a$, $|\mathcal{C}_4(n, \log n + a)| \gtrsim C\frac{2^n}{n\sqrt{n}}$, where $C = \frac{2^a - 1}{2^{2a+1}\sqrt{2\pi}}$.

Completion of the Proof of Theorem 38:

$$\begin{aligned} |\mathcal{C}_4(n,k)| &\ge \binom{n'}{n/2-2} - (n'-k+1)\binom{n'-k}{n/2-2} \\ &= \binom{n'-k}{n/2-2}\left[\prod_{i=0}^{k-1}\frac{n'-i}{n'-(n/2-2)-i} - n'+k-1\right] \end{aligned}$$

$$= \binom{n-2k-2}{n/2-2}\left[\prod_{i=0}^{k-1}\frac{n-k-2-i}{n/2-k-i} - n + 2k + 1\right]$$
$$\geq \binom{n-2k-2}{n/2-2}\left[2^k - n + 2k + 1\right]$$
$$= \binom{n-2k-2}{n/2-2} n\left[\frac{2^k+2k+1}{n} - 1\right].$$

Note that

$$\binom{n-2k-2}{n/2-2} = \binom{n-2k-2}{\frac{n-2k-2}{2}}\frac{\frac{n-2k-2}{2}!}{(\frac{n}{2}-2)!}\frac{\frac{n-2k-2}{2}!}{(\frac{n}{2}-2k)!}$$
$$= \binom{n-2k-2}{\frac{n-2k-2}{2}}\frac{(\frac{n}{2}-k-1)!}{(\frac{n}{2}-2)!}\frac{(\frac{n}{2}-k-1)!}{(\frac{n}{2}-2k)!}$$
$$= \binom{n-2k-2}{\frac{n-2k-2}{2}}\prod_{i=1}^{k-1}\frac{\frac{n}{2}-k-i}{\frac{n}{2}-1-i}$$
$$\geq \binom{n-2k-2}{\frac{n-2k-2}{2}}\left(\frac{\frac{n}{2}-2k+1}{\frac{n}{2}}\right)^{k-1}$$
$$= \binom{n-2k-2}{\frac{n-2k-2}{2}}\left(1-\frac{4k-2}{n}\right)^{k-1}.$$

For $k = \log n + a$ we get the following

$$= \binom{n-2k-2}{\frac{n-2k-2}{2}}\left(1-\frac{4k-2}{n}\right)^{\frac{n}{4k-2}\frac{(4k-2)(k-1)}{n}}$$
$$\approx \binom{n-2k-2}{\frac{n-2k-2}{2}} e^{-\frac{(4k-2)(k-1)}{n}}$$
$$\approx \binom{n-2k-2}{\frac{n-2k-2}{2}}$$
$$\approx \frac{2^{n-2k-2+1}}{\sqrt{2\pi n-2k-2}}$$
$$\geq \frac{2^n}{2^{2a+1}n^2\sqrt{2\pi n}}.$$

Finally, we conclude that

$$|\mathcal{C}_4(n,k)| \gtrsim \frac{2^n}{2^{2a+1}n^2\sqrt{2\pi n}} n\left[\frac{2^k+2k+1}{n} - 1\right]$$
$$= \frac{2^n}{2^{2a+1}n\sqrt{2\pi n}}\left[\frac{n2^a+2\log n+2a+1}{n} - 1\right]$$
$$\approx \frac{2^n}{n\sqrt{n}}\frac{2^a-1}{2^{2a+1}\sqrt{2\pi}}.$$

■

ACKNOWLEDGMENTS

The authors would like to thank Olgica Milenkovic and Ryan Gabrys for valuable discussions on DNA storage and Ron M. Roth for his contribution to the proof of Theorem 16. They also thank the three anonymous reviewers and the associate editor Chaoping Xing for their valuable comments and suggestions, which have contributed for the clarity of the paper and its presentation.

REFERENCES

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

[2] E. Barcucci, S. Bilotta, E. Pergola, R. Pinzani, and J. Succi, "Cross-bifix-free sets generation via Motzkin paths," *RAIRO-Theor. Inform. Appl.*, vol. 50, no. 1, pp. 81–91, Jun. 2016.

[3] L. A. Bassalygo, "On separated comma-free codes," (in Russian), *Problemy Peredachi Inform.*, vol. 2, no. 4, pp. 78–79, 1966.

[4] A. Bernini, S. Bilotta, R. Pinzani, A. Sabri, and V. Vajnovszki, "Prefix partitioned Gray codes for particular cross-bifix-free sets," *Cryptogr. Commun.*, vol. 6, no. 4, pp. 359–369, 2014.

[5] A. Bernini, S. Bilotta, R. Pinzani, and V. Vajnovszki, "A Gray code for cross-bifix-free sets," *Math. Struct. Comput. Sci.*, vol. 27, no. 2, pp. 184–196, 2017.

[6] S. Bilotta, E. Pergola, and R. Pinzani, "A new approach to cross-bifix-free sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 6, pp. 4058–4063, Jun. 2012.

[7] S. R. Blackburn, "Non-overlapping codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4890–4894, Sep. 2015.

[8] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," in *Proc. ASPLOS*, Atlanta, GA, USA, Apr. 2016, pp. 637–649.

[9] J. Brakensiek, V. Guruswami, and A. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, pp. 3403–3410, May 2018.

[10] Y. M. Chee, H. M. Kiah, P. Purkayastha, and C. Wang, "Cross-bifix-free codes within a constant factor of optimality," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4668–4674, Jul. 2013.

[11] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, p. 1628, 2012.

[12] W. Eastman, "On the construction of comma-free codes," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 2, pp. 263–267, Apr. 1965.

[13] E. Gilbert, "Synchronization of binary messages," *IRE Trans. Inf. Theory*, vol. 6, no. 4, pp. 470–477, Sep. 1960.

[14] S. W. Golomb, B. Gordon, and L. R. Welch, "Comma-free codes," *Can. J. Math.*, vol. 10, no. 2, pp. 202–209, 1958.

[15] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Duplication-correcting codes for data storage in the DNA of living organisms," *IEEE Trans. Inf. Theory*, vol. 63, no. 8, pp. 4996–5010, Aug. 2017.

[16] A. Kato and K. Zeger, "A comment regarding: On the capacity of two-dimensional run length constrained channels," Tech. Rep., 2005.

[17] W. Kautz, "Fibonacci codes for synchronization control," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 2, pp. 284–292, Apr. 1965.

[18] D. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 51–53, Jan. 1986.

[19] A. A. Kulkarni and N. Kiyavash, "Nonasymptotic upper bounds for deletion correcting codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5115–5130, Aug. 2013.

[20] V. I. Levenshtein, "Decoding automata which are invariant with respect to their initial state," (in Russian), *Problems Cybern.*, vol. 12, pp. 125–136, 1964.

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

[22] V. I. Levenshtein, "Bounds for codes ensuring error correction and synchronization," *Problemy Peredachi Inform.*, vol. 5, no. 2, pp. 3–13, 1969.

[23] V. I. Levenshtein, "Maximum number of words in codes without overlaps," *Problems Inform. Transmiss.*, vol. 6, no. 4, pp. 355–357, 1970.

[24] V. I. Levenstein, "Combinatorial problems motivated by comma-free codes," *J. Combinat. Des.*, vol. 12, no. 3, pp. 184–196, Dec. 2004.

[25] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, Germany, Jun. 2017, pp. 3115–3119.

[26] B. H. Marcus, R. M. Roth, and P. H. Siegel, *An Introduction to Coding for Constrained Systems*. 2001.

[27] M. Qin, E. Yaakobi, and P. H. Siegel, "Time–space constrained codes for phase-change memories," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5102–5114, Aug. 2013.

[28] M. F. Schilling, "The surprising predictability of long runs," *Math. Mag.*, vol. 85, no. 2, pp. 141–149, 2012.

[29] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes for correcting a burst of deletions or insertions," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 1971–1985, Apr. 2017.

[30] G. M. Tenengolts, "Nonbinary codes, correcting single deletion or insertion (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 5, pp. 766–769, Sep. 1984.

[31] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors," (in Russian), *Autom. Telemkhanika*, vol. 161, no. 3, pp. 288–292, 1965.

[32] J. H. Weber, K. A. S. Immink, P. H. Siegel, and T. G. Swart, "Polarity-balanced codes," in *Proc. Inf. Theory Appl. Workshop (ITA)*, San Diego, CA, USA, Feb. 2013, pp. 1–5.

[33] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Trans. Mol. Biol. Multi-Scale Commun.*, vol. 1, no. 3, pp. 230–248, Sep. 2015.

[34] S. M. Hossein, T. Yazdi, H. M. Kiah, and O. Milenkovic, "Weakly mutually uncorrelated codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 2649–2653.

[35] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Nature Sci. Rep.*, vol. 5, Aug. 2015, Art. no. 14138.

Maya Levy received B.Sc. in Computer Engineering and M.Sc. in Computer Science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2014 and 2018, respectively. Her research interests are algorithms and coding for DNA-based storage. From 2011 to 2014 Maya has worked at Intel as an intern in GPU architecture team. She also participated twice in summer internship program at Microsoft, Redmond, Washington in 2016 and 2017. Maya has recently joined Google as a software engineer.

Eitan Yaakobi (S'07–M'12–SM'17) is an Assistant Professor at the Computer Science Department at the Technion — Israel Institute of Technology. He received the B.A. degrees in computer science and mathematics, and the M.Sc. degree in computer science from the Technion — Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011. Between 2011–2013, he was a postdoctoral researcher in the department of Electrical Engineering at the California Institute of Technology. His research interests include information and coding theory with applications to non-volatile memories, associative memories, data storage and retrieval, and voting theory. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010-2011.