Mutually Uncorrelated Codes for DNA Storage

Maya Levy
Technion - Israel Institute of Technology
Haifa 32000, Israel
mayalevy@cs.technion.ac.il

Eitan Yaakobi
Technion - Israel Institute of Technology
Haifa 32000, Israel
yaakobi@cs.technion.ac.il

***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 work 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. Then, we present an efficient algorithm for MU codes with linear encoding and decoding complexity. Next, we extend these results for (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. Particularly we show an efficient construction of these codes with nearly optimal redundancy and draw connections to the problem of comma-free and prefix synchronized codes. Lastly, we provide similar results for the edit distance and balanced MU 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 [13] for the purpose of synchronization, and has received attention recently due to its relevance and applicability for DNA storage [21]. 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. [7] in 2012. Since then a few similar systems were implemented but they did not support random access to the memory. Recently, in [5], [21] two random access DNA storage systems were proposed. To enable random access, the authors suggested to equip the DNA information blocks with unique addresses, which are aimed to identify the corresponding blocks during the reading process.

Obtaining a *good* set of addresses 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 addresses set should satisfy are listed by Yazdi et al. in [21]. 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 addresses, 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 addresses to be balanced since balanced DNA sequences increase the chances of successful reads. In addition, the authors of [19] 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* [15], [16], *non-overlapping codes* [4] and *cross-bifix-free codes* [1], [3], [6]. 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 [15], states that $A_{MU}(n, q) < q^n/(en)$, while the best known constructive lower bound, given independently in [6], [8], [13], states that $A_{MU}(n, q) \gtrsim \frac{q-1}{qe} \cdot \frac{q^n}{n}$. Hence, for the binary case there is still a gap of factor 2 between the lower and upper bound. The construction of MU codes is explicit. Given some $k < n$, one fixes the first k bits to be zero, followed by a single one, and the last bit is one as well. The sequence of the remaining $n-k-2$ needs to satisfy the constraint that it does not have a zeros 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 show that in the binary case indeed it is not possible to achieve larger codes using this construction. Lastly, we present efficient encoding and decoding algorithms with linear complexity which results in MU codes with $\lceil \log(n) \rceil + 4$ redundancy bits.

We then extend our study for 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 $\lfloor \frac{d_h+1}{2} \rfloor \log(n) + (d_m - 1)\log\log(n) + \mathcal{O}(d_m \log d_m)$, 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. 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.

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 $[1, k]$. Let Σ_q be a finite alphabet of size q and Σ_q^n be the set of vectors of size n over the alphabet Σ_q. We also denote by $\mathbb{F}_2^n$ the set of binary vectors of length n. 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 *zeros 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 $\mathbf{a}$ i times and $\mathbf{ab}$ is the concatenation of the two vectors $\mathbf{a}$ and $\mathbf{b}$. For two functions $f(n), g(n)$ we say $f(n) \lesssim g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} \le 1$, and $f(n) \approx g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$.

Definition 1 *Two not necessarily distinct words $\mathbf{a}, \mathbf{b} \in \Sigma_q^n$ are **mutually uncorrelated** if any non-trivial prefix of $\mathbf{a}$ does not* *match a non-trivial suffix of* $\boldsymbol{b}$ *and vice versa. 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.*

Let us denote by $A_{MU}(n,q)$ the largest cardinality of an MU code. Levenshtein showed in [15] that for all n and q

$$A_{MU}(n,q) \leq \left(\frac{n-1}{n}\right)^{n-1} \frac{q^n}{n} < \frac{q^n}{en}. \qquad (1)$$

We now recall a family of MU codes which was suggested independently first by Gilbert in [8], later by Levenshtein in [13] and recently by Chee et al. in [6].

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

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

It was proved in [6], [8], [13] that there exists an appropriate choice of k for which the following lower bound holds

$$|\mathcal{C}_1(n,k)| \gtrsim \frac{q-1}{qe} \cdot \frac{q^n}{n} \qquad (2)$$

where $n \to \infty$ over the sub sequence $\frac{q^i-1}{q-1}, i \in \mathbf{N}$.

However, it was not established in these works what the maximal asymptotic cardinality of $\mathcal{C}_1(n,k)$ is.

Additional interesting approach for constructing binary MU codes was proposed in [3] and was later generalized in [1] to alphabets of size $q > 2$. In this case, it was shown that for some small values of n the approach in [1] achieves larger cardinality than Construction 1. Both works [1], [3] do not offer asymptotic analysis of the constructions' redundancy, however, for the binary case it is commonly believed that Construction 1 provides the best known asymptotic code size.

III. Mutually Uncorrelated Codes

In this section we expand the study of MU codes. Specifically, we show that the lower bound in (2) is asymptotically tight. We also show an efficient encoding scheme for MU codes, which improves upon a recent result from [17].

Theorem 1 *For* $q = 2$,

$$\max_{1\leq k<n}\{|\mathcal{C}_1(n,k)|\} \approx \frac{2^n}{n} 2^{F(n)} \leq \frac{2^n}{2en},$$

where $F(n) = \Delta_n - \min\left\{\log(e)2^{\Delta_n}+1, 2\log(e)2^{\Delta_n}\right\}$, *and* $\Delta_n = \log(n) - \lceil\log(n)\rceil$.

Proof Sketch: It can be shown that the size of $\mathcal{C}_1(n,k)$ is maximized by choosing $k = \lceil\log(n)\rceil + a$ for some integer a. We therefore assume without loss of generality that $k = \lceil\log(n)\rceil + a$ with $a \in \mathbf{Z}$. Let $S(n,k)$ be the set of all binary vectors with no zeros runs of length k,

$$S'(n,k) = \{\boldsymbol{s} \in S(n,k) \mid \boldsymbol{s} \text{ starts or ends with } k/2 \text{ zeros}\},$$

and $s(n,k) = |S(n,k)|, s'(n,k) = |S'(n,k)|$. Note that $s'(n,k) \leq 2 \cdot 2^{n-k/2} = 2^{n-k/2+1}$. Lastly, we denote $\widehat{S}(n,k) = S(n,k) \setminus S'(n,k)$ and $\widehat{s}(n,k) = |\widehat{S}(n,k)| = s(n,k) - s'(n,k) \geq s(n,k) - 2^{n-k/2+1}$. For all m large enough, we consider the set $T(m) = \widehat{S}(n,k)^m = \widehat{S}(n,k) \times \cdots \times \widehat{S}(n,k)$, and $t(m) = |T(m)| = \widehat{s}(n,k)^m$. Note that the set $T(m)$ satisfies the $(0,k-1)$-RLL constraint [9], and hence

$$\lim_{m\to\infty} \frac{\log(t(m))}{nm} = \lim_{m\to\infty} \frac{\log(\widehat{s}(n,k))}{n} = \frac{\log(\widehat{s}(n,k))}{n} \leq E_{0,k-1}.$$

where $E_{0,k-1}$ is the capacity of the $(0,k-1)$-RLL constraint. We get $s(n,k) \leq 2^{nE_{0,k-1}} + 2^{n-k/2+1}$ and it is also possible to show that $2^{nE_{0,k-1}} \leq s(n,k)$. We use the result $\lim_{k\to\infty} \frac{1-E_{0,k}}{\log(e)2^{-k-2}} = 1$ from [10] to get $s(n,k) \approx 2^{nE_{0,k-1}}$ and since $|\mathcal{C}_1(n,k)| = s(n-k-2,k)$, we have

$$\frac{n|\mathcal{C}_1(n,k)|}{2^n} \approx 2^{\log(n)+(n-\lceil\log(n)\rceil-a-2)E_{0,\lceil\log(n)\rceil+a-1}-n}.$$

We denote

$$f_a(n) = \log(n) + (n - \lceil\log(n)\rceil - a - 2)E_{0,\lceil\log(n)\rceil+a-1} - n,$$

and

$$f_a(n) \approx \Delta_n - \frac{\log(e)}{2^{a+1}}2^{\Delta_n} - a - 2.$$

In order to maximize the size of the code it is possible to show that $\max_{a\in\mathbf{Z}}\{f_a(n)\} = F(n)$, and conclude

$$\max_{1\leq k<n}\{|\mathcal{C}_1(n,k)|\} \approx \frac{2^n}{n}2^{F(n)} \leq \frac{2^n}{2en},$$

where the last step is verified showing that $\max_{x\in[-1,0]}\{x - \min\{\log(e)2^x+1, 2\log(e)2^x\}\} = -\log(e) - 1$. ■

Construction 1 does not provide MU codes with efficient encoder and decoder. To show such a construction with linear complexity, one has to choose the value of k and define a code of length $n-k-2$ without zeros runs of length k. This result is shown in Algorithm 1, where we choose $k = \lceil\log(n)\rceil + 1$.

Algorithm 1 Zero Run-Length Encoding

```
Input: Sequence $\mathbf{x} \in \mathbb{F}_2^{n'}$
Output: $\mathbf{y} \in \mathbb{F}_2^{n'+1}$ with zeros run length $\leq \lceil\log(n)\rceil$
 1: Define $\mathbf{y} = \mathbf{x}1 \in \mathbb{F}_2^{n'+1}$
 2: Set $i = 1$ and $i_{end} = n'$
 3: while $i \leq i_{end} - \lceil\log(n)\rceil$ do
 4:     if $w_H(\mathbf{y}_i^{i+\lceil\log(n)\rceil}) = 0$ then
 5:         remove the zeros run $\mathbf{y}_i^{i+\lceil\log(n)\rceil}$ from $\mathbf{y}$
 6:         $p(i)$: binary representation of $i$ with $\lceil\log(n)\rceil$ bits
 7:         append $p(i)0$ to the right of $\mathbf{y}$
 8:         set $i_{end} = i_{end} - \lceil\log(n)\rceil - 1$
 9:     else
10:         set $i = i + 1$
11:     end if
12: end while
```

The following lemma proves the correctness of Algorithm 1.

Lemma 1 *For all* $n' \leq n$, *given any vector* $\boldsymbol{x} \in \mathbb{F}_2^{n'}$, *Algorithm 1 outputs a sequence* $\mathbf{y} \in \mathbb{F}_2^{n'+1}$, *where any zeros run has length at most* $\lceil\log(n)\rceil$ *and such that* $\boldsymbol{x}$ *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 $\lceil\log(n)\rceil+1$ zeros 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 $1 \leq t \leq n'$ 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 zeros runs and $\mathbf{y}_{t+2}^{n'+1}$ is the list of the pointers $p(i)0$ representing the indices of the removed zeros runs.

To reconstruct $\mathbf{x}$ given $\mathbf{y}$ we start by locating the separating bit 1 on position $t+1$. We start from the right and check whether the rightmost bit is 1 or 0. In case it is 0, then 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 zeros 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 zeros 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 zeros 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 $p(i)0$. It suffices to show that any sub-vector of $\mathbf{y}_{t+2}^{n'+1}$ of the form $(p(i_1), 0, p(i_2))$ does not consist a zeros run of length greater than $\lceil\log(n)\rceil$. We do not write the index zero and from the algorithm, $0 < i_1 \le i_2$. We consider the leftmost ones in $p(i_1)$ and $p(i_2)$. Since $i_1 \le i_2$, the position of the leftmost one within $p(i_1)$ is lower or equal to the position of the leftmost one within $p(i_2)$, thus any window of length $\lceil\log(n)\rceil + 1$ must contain at least one of the leftmost ones from $p(i_1)$ and $p(i_2)$. ■

Constructing an MU code using Algorithm 1 and Construction 1 gives us the following result.

Corollary 1 *There exists a construction of an MU code with redundancy $\lceil\log(n)\rceil + 4$ and linear encoding and decoding time and space complexities.*

According to Theorem 1 the lowest asymptotic redundancy that can be achieved by $\mathcal{C}_1(n,k)$ when $q = 2$ is $\log(2en)$. Our result shows that with roughly 1.56 additional redundancy bits it is possible to have an efficient implementation of MU codes. A similar algorithm to the problem solved by Algorithm 1 was recently proposed in [17] to efficiently encode sequences that do not contain runs of zeros and ones of length k with a single redundancy bit. Specifically, in [17] the authors 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 [17] with $k = \lceil\log(n)\rceil + 2$. Lastly, we note that Kauts presented in [11] an algorithm which encodes all words avoiding zeros runs of any specific length with optimal redundancy. However, the space complexity of this algorithm is $\Theta(n^2)$.

IV. (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 mutual uncorrelatedness constraint, by requiring the prefixes and suffixes to not only differ but also have a large distance.

Definition 2 *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. Motivated by Levenshtein's upper bound [14], we first provide an upper bound on the value $A_{MU}(n, q, d_h, d_m)$.

Theorem 2 *For all n, q, d_h, d_m,*

$$A_{MU}(n, q, d_h, d_m) \le \frac{M(n, q, d)}{\lfloor n/d_m \rfloor}, \qquad (3)$$

where $d = \min\{d_h, 2d_m\}$. 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

$$\widehat{\mathcal{C}} = \{(\mathbf{aa})_{i+1}^{n+i} | \mathbf{a} \in \mathcal{C}, i = \alpha \cdot d_m, \alpha \in [0, \lfloor n/d_m \rfloor - 1]\}. \quad (4)$$

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 $i = j$ then $\mathbf{a} \ne \mathbf{b}$ and $d_H(\mathbf{a}, \mathbf{b}) = d_H(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \ge d_h$. Otherwise, $i \ne j$ and we assume without loss of generality that $i > j$. Notice that $\widehat{\mathbf{a}}_{n-i+1}^{n-j}$ 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 $d_H(\widehat{\mathbf{a}}_{n-j+1}^{n}\widehat{\mathbf{a}}_1^{n-i}, \widehat{\mathbf{b}}_{n-j+1}^{n}\widehat{\mathbf{b}}_1^{n-i}) \ge d_m$. Therefore,

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

We showed that $d_H(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \ge \min\{d_h, 2d_m\}$, thus $|\widehat{\mathcal{C}}| = \lfloor n/d_m \rfloor \cdot |\mathcal{C}| \le M(n, q, d)$ where $d = \min\{d_h, 2d_m\}$ and the theorem follows directly. ■

Before presenting a construction of (d_h, d_m)-MU codes, we introduce a new constraint that will be used in this construction.

Definition 3 *Let d be a positive integer. 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 $\forall i \in [n-k+1] : w_H(\boldsymbol{a}_i^{i+k-1}) \ge d$.*

This constraint states that a vector $\mathbf{a} \in \Sigma_q^n$ is a (d,k)-WWL vector if the weight of each consecutive subsequence of length k in $\mathbf{a}$ is at least d. A set of (d,k)-WWL vectors will be called a (d,k)-*WWL code*.

For the rest of the paper we let $F(n,d)$ be

$$F(n,d) = \lceil\log(n)\rceil + (d-1)(\lceil\log\lceil\log(n)\rceil\rceil + C) + 2, \quad (5)$$

where C is a constant equal to the minimum integer such that $2^{\lceil\log\lceil\log(n)\rceil\rceil + C} \ge \lceil\log(F(n,d))\rceil + 2$. We present an explicit algorithm for encoding and decoding $(d, F(n,d))$-WWL binary vectors with $n' \le n$ information bits and d redundancy bits.

Algorithm 2 Window Weight Limited Encoding

```
Input: $\mathbf{x} \in \mathbb{F}_2^{n'}$ and an integer $d > 1$
Output: $(d, F(n,d))$-WWL vector $\mathbf{y} \in \mathbb{F}_2^{n'+d}$
 1: Define $\mathbf{y} = \mathbf{x}1^d \in \mathbb{F}_2^{n'+d}$
 2: Set $i = 1$ and $i_{end} = n'$
 3: while $i \le i_{end} - F(n,d) + 1$ do
 4:     if $w_H(\mathbf{y}_i^{i+F(n,d)-1}) < d$ then
 5:         remove $\mathbf{y}_i^{i+F(n,d)-1}$ from $\mathbf{y}$
 6:         $p(i)$: binary representation of $i$ with $\lceil\log(n)\rceil$ bits
 7:         for $j = 1, \ldots, d-1$ do
 8:             $t(j)$ : binary representation of the index of the $j$th 1
                    in $\mathbf{y}_i^{i+F(n,d)-1}$with $\lceil\log\lceil\log(n)\rceil\rceil + C$ bits
 9:         end for
10:         append $p(i)t(1)\cdots t(d-1)01$ to the right of $\mathbf{y}$
11:         set $i_{end} = i_{end} - F(n,d)$
12:         set $i = i - F(n,d) + 1$
13:     else
14:         set $i = i + 1$
15:     end if
16: end while
```

Lemma 2 *For all $n' \le n$, given any vector $\boldsymbol{x} \in \mathbb{F}_2^{n'}$ Algorithm 2 outputs a $(d, F(n,d))$-WWL vector $\boldsymbol{y} \in \mathbb{F}_2^{n'+d}$ such that $\boldsymbol{x}$ can be uniquely reconstructed given $\boldsymbol{y}$. The time and space complexity of the algorithm and its inverse is $\Theta(n)$.*

We are now ready to show a construction of (d_h, d_m)-MU codes. We say that a vector $\mathbf{u} \in \mathbb{F}_2^\ell$ is a d-*auto-cyclic vector* if for every $1 \le i \le d$, $d_H(\mathbf{u}, 0^i\mathbf{u}_1^{\ell-i}) \ge d$. In the next construction we use the following d-auto-cyclic vector $\mathbf{u}$ of length $\ell(d) = d\lceil\log d\rceil + 2d$, which is given by $\mathbf{u} = 1^d\mathbf{u}_0 \cdots \mathbf{u}_{\lceil\log d\rceil} \in \mathbb{F}_2^{\ell(d)}$ such that $\mathbf{u}_i = ((1^{2^i}0^{2^i})^d)_1^d$.

Construction 2 *Let n, k be two integers such that $k \geq \ell(d_m)$ and $n \geq k+\ell(d_m)+2d_m$. Denote $n' = n-k-\ell(d_m)-2d_m$. Let $\mathcal{C}_H$ be a length-n' (d_m, k)-WWL code with minimum Hamming distance d_h.*

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

Theorem 3 *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.

Claim 1 *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 $\mathbf{b}_{d_m+1}^n = 0^{k-d_m}1^{d_m}\mathbf{u}1^{d_m}\mathbf{c}1^{d_m}$, where 0^{k-d_m}, $\mathbf{u}$ and $\mathbf{c} \in \mathcal{C}_H$ are all (d_m, k)-WWL vectors. From Claim 1 $\mathbf{b}_{d_m+1}^n$ is also a (d_m, k)-WWL vector and since $n-i+1 > d_m, \mathbf{b}_{n-i+1}^n$ is also a (d_m, k)-WWL vector. Therefore in its first k positions there are at least d_m ones and $d_H(\mathbf{a}_1^i = 0^k\mathbf{a}_{k+1}^i, \mathbf{b}_{n-i+1}^n) \geq d_m$.
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 $d_H(\mathbf{a}_1^i, \mathbf{b}_{n-i+1}^n) = d_H(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \geq d_H(\widehat{\mathbf{a}}_{k-j}^{k-j+\ell(d_m)}, \widehat{\mathbf{b}}_{k-j}^{k-j+\ell(d_m)}) \geq d_m$.

■

We can use Algorithm 2 together with a systematic code with minimum distance d_h in order to explicitly construct the code $\mathcal{C}_H$ with $k = F(n, d_m)+1$ in Construction 2.

Corollary 2 *There exist (d_h, d_m)-MU codes with redundancy $\lfloor \frac{d_h+1}{2} \rfloor \log(n) + (d_m-1)\log\log(n) + \mathcal{O}(d_m \log d_m)$ and linear time and space complexity.*

Proof Sketch: We use Algorithm 2 to generate a $(d_m, F(n, d_m))$-WWL code of length $\tilde{n} < n$, where the choice of $\tilde{n}$ will be explained later. We then guarantee minimum Hamming distance d_h by applying a systematic BCH encoder on the output of Algorithm 2. The approximately $\lfloor \frac{d_h-1}{2} \rfloor \log(\tilde{n})$ redundancy bits added by the BCH encoder are positioned within the $\tilde{n}$ bits such that every two redundancy bits are located at least $F(n, d_m)+1$ positions apart. We denote the resulting set of vectors by $\mathcal{C}'$. Notice that the code $\mathcal{C}'$ is a $(d_m, F(n, d_m)+1)$-WWL code with minimum Hamming distance d_h and its length is a function of $\tilde{n}$ which we denote by $f(\tilde{n})$. We construct the code $\mathcal{C}_2(n,k,d_h,d_m)$ by choosing $k = F(n, d_m)+1$ and using $\mathcal{C}'$ as $\mathcal{C}_H$. The choice of $\tilde{n}$ is determined such that $f(\tilde{n}) = n' = n-k-\ell(d_m)-2d_m$ is satisfied. The total redundancy of this construction is upper bounded by

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

■

An interesting related problem is discussed by Levenshtein in [14], where he described prefix synchronized codes with index ρ. A code $\mathcal{C} \subseteq \mathbb{F}_2^n$ is said to be prefix synchronized with prefix $\mathbf{h} \in \mathbb{F}_2^m, m \leq n$ and index ρ if for any $\mathbf{a} \in \mathcal{C}$, $i \in [2,n], d_H((\mathbf{ah})_i^{i+m-1}, \mathbf{h}) \geq \rho$. Levenshtein stated in [14] that when n goes to infinity, the redundancy of the maximal prefix synchronized code with index ρ is $\log n + (\rho-1)\log\log n - o(\log\log n)$. Notice that $\mathcal{C}_2(n,k,1,d_m)$ is prefix synchronized with $\mathbf{h} = 0^k\mathbf{u}$ and $\rho = d_m$, hence by Corollary 2 we provide an efficient construction to the problem with only $o(\log\log n)$ additional bits of redundancy. Similarly an extension to comma-free codes by Levenshtein [14] states that $\mathcal{C} \subseteq \Sigma_q^n$ is a (d,ρ)-comma-free code if for any $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathcal{C}$, $i \in [2,n]$, $d_H((\mathbf{ab})_i^{i+n-1}, \mathbf{c}) \geq \rho$, and $\mathcal{C}$ has a minimum Hamming distance d. Note that any (d,ρ)-MU code is also a (d,ρ)-comma-free code, so we can use the result of Corollary 2 to construct efficient (d,ρ)-comma-free codes with $\lfloor \frac{d+1}{2} \rfloor \log n + (\rho-1)\log\log n + o(\log\log n)$ redundancy bits. To the best of our knowledge, the lowest redundancy that an efficient construction to this problem achieved so far has been $\sqrt{\rho n}$ [2], [16].

Lastly, we note that Construction 2 improves upon Construction 3 from [20], which solves the problem of $(d_h, d_m = 1)$-MU codes but requires $\mathcal{O}(\sqrt{n})$ redundancy bits.

V. 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 [19] which report deletion errors during the synthesis of DNA strands.

The edit distance $d_E(\mathbf{a}, \mathbf{b})$ 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 maximal d such that for any two distinct words $\mathbf{a}, \mathbf{b} \in \mathcal{C}, d_E(\mathbf{a}, \mathbf{b}) \geq d$.

Definition 4 *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 4 is different than the MU constraints we introduced so far since we consider also suffixes and prefixes of different lengths. 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 a (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.

Theorem 4 *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 $d = \min\{d_e, d_m\}$.*

It can be proved that the code $\mathcal{C}_2(n,k,d_e,d_m)$ is a $(2, d_m)$-EMU code. In order to increase the minimum edit distance, we slightly modify Construction 2 as follows.

Construction 3 *Let n, k be two integers such that $k \geq d_m$ and $n \geq k+2d_m$. Denote $n' = n-k-2d_m$. Let $\mathcal{C}_E$ be a*

Table I
REDUNDANCY SUMMARY FOR BINARY CODES

Property	MU	(d_h, d_m)- MU	$(4, d_m)$- EMU	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 1	Construction 2	Construction 3	Construction 4
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		$d = \min\{2d_m, d_h\}$	$d_m \geq 4$	

(d_m, k)-WWL code of length n' with minimum edit distance d_e. The code $\mathcal{C}_3(n, k, d_e, d_m)$ is defined as follows,

$$\mathcal{C}_3(n, k, d_e, d_m) = \{0^k 1^{d_m} \boldsymbol{c} 1^{d_m} \in \mathbb{F}_2^n \mid \boldsymbol{c} \in \mathcal{C}_E\}.$$

Theorem 5 *The code $\mathcal{C}_3(n, k, d_e, d_m)$ is a (d_e, d_m)-EMU code.*

There exists an explicit efficient method to construct a $(d_m, F(n, d_m) + 1)$-WWL codes with minimum edit distance 4, which will be used as the code $\mathcal{C}_E$ in Construction 3. For this, we use Algorithm 2 and the well known Varshamov and Tenengolts (VT) codes with edit distance four in their systematic version [18].

Theorem 6 *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.*

VI. Balanced Mutually Uncorrelated Codes

A binary word of length n, when n is even, is said to be *balanced* if its Hamming weight is $n/2$. For the extension of $q > 2$, when q is even, we follow the balanced definition from [20] and say that a code $\mathcal{C} \subseteq \Sigma_q^n$ is 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 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 *balanced MU code* if $\mathcal{C}$ is balanced and also an MU code. Let $A_{BMU}(n, q)$ denote the maximum cardinality of balanced MU codes.

Theorem 7 *For all n, q, $A_{BMU}(n, q) \leq \frac{\binom{n}{n/2}\left(\frac{q}{2}\right)^n}{n} \approx \frac{2q^n}{n\sqrt{2\pi n}}$. In particular, the minimum redundancy of balanced MU codes is $1.5\log(n) + \mathcal{O}(1)$.*

Next is a construction of binary balanced MU codes.

Construction 4 *Let n, k be two integers such that $1 \leq k < n$. The code $\mathcal{C}(n, k) \subseteq \mathbb{F}_2^n$ is defined as follows,*

$$\mathcal{C}_4(n, k) = \{0^k 1 \boldsymbol{c} 1 \mid w_H(\boldsymbol{c}) = \frac{n}{2} - 2, \boldsymbol{c} \text{ is a } (1, k)\text{ -WWL vector}\}.$$

The correctness and redundancy result of Construction 4 are stated in the next theorem.

Theorem 8 *The code $\mathcal{C}_4(n, k)$ is a 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 $C = \frac{2^a - 1}{2^{2a+1}\sqrt{2\pi}}$.*

The extension of Construction 4 for non-binary is direct, since every symbol can store $q/2$ values after the assignment of binary values. Hence the number of redundancy symbols remains the same, i.e., $1.5\log_q(n) + \mathcal{O}(1)$. This meets the result from [20] where a 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 one in [20].

An efficient implementation of balanced MU codes can be achieved by adopting Knuth's algorithm for balancing binary words [12]. Due to the lack of space, we only state our result for such a construction.

Theorem 9 *There exists a construction of a balanced MU code with $2\log(n) + \log\log(n) + o(\log\log n)$ redundancy bits and linear time and space complexity.*

VII. Conclusion

In this work we studied MU codes as well as their variations for codes with minimum Hamming distance, minimum edit distance, and balanced codes. Similar techniques can be applied to construct balanced MU codes together with minimum Hamming distance and thereby satisfying three of the constraints listed in [20]. The results in the paper are summarized in Table I. 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 complexity.

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 1.

References

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

[2] L. A. Bassalygo, "On the separation of comma-free codes (in Russian)," *Probl. Pered. Informatsii*, vol. 2, no. 4, pp. 78–79, 1966.

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

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

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

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

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

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

[9] K. A. S. Immink, *Coding techniques for digital recorders*, Prentice Hall, College Div., 1991.

[10] A. Kato and K. Zeger, "A comment regarding: On the capacity of two-dimensional run length constrained channels," *personal manuscript*, 2005.

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

[12] D. E. Knuth, "Efficient balanced codes," *IEEE Trans. on Inf. Theory*, vol. 32, no. 1, pp. 51–53, 1986.

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

[14] V. I. Levenshtein, "Bounds for codes ensuring error correction and synchronization," *Probl. Pered. Informatsii*, vol. 5, no. 2, pp. 3–13, 1969.

[15] V. I. Levenshtein, "Maximum number of words in codes without overlaps," *Prob. Inform. Transmission*, vol. 6, pp. 355–357, 1970.

[16] V. I. Levenshtein, "Combinatorial problems motivated by comma-free codes," *J. of Comb. Designs*, vol. 12, no.3, pp. 184–196, 2004.

[17] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes for correcting a burst of deletions or insertions," in *Proc. IEEE Int. Symp. Inf. Theory*, pp. 630–634, Barcelona, Spain, Jul. 2016.

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

[19] S. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol 1, no. 3, pp. 230–248, 2015.

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

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