(C) 2017 Seyed Mohammadhossein Tabatabaei Yazdi

# BY <br> <br> SEYED MOHAMMADHOSSEIN TABATABAEI YAZDI 

 <br> <br> SEYED MOHAMMADHOSSEIN TABATABAEI YAZDI}

## DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:
Professor Olgica Milenkovic, Chair
Professor Iwan Duursma
Professor Andrew Singer
Professor Venugopal V. Veeravalli
Associate Professor Jian Ma, Carnegie Mellon University

## ABSTRACT

Despite the many advances in traditional data recording techniques, the surge of Big Data platforms and energy conservation issues has imposed new challenges to the storage community in terms of identifying extremely high volume, non-volatile and durable recording media. The potential for using macromolecules for ultra-dense storage was recognized as early as 1959 when Richard Feynman outlined his vision for nanotechnology in a lecture, "There is plenty of room at the bottom". Among known macromolecules, DNA is unique insofar as it lends itself to implementations of non-volatile recording media of outstanding integrity and extremely high storage capacity.

The basic system implementation steps for DNA-based data storage systems include synthesizing DNA strings that contain user information and subsequently retrieving them via high-throughput sequencing technologies. Existing architectures enable reading and writing but do not offer random-access and error-free data recovery from low-cost, portable devices, which is crucial for making the storage technology competitive with classical recorders.

In this work we advance the field of macromolecular data storage in three directions. First, we introduce the notion of weakly mutually uncorrelated (WMU) sequences. WMU sequences are characterized by the property that no sufficiently long suffix of one sequence is the prefix of the same or another sequence. For this purpose, WMU sequences used for primer design in DNAbased data storage systems are also required to be at large mutual Hamming distance from each other, have balanced compositions of symbols, and avoid primer-dimer byproducts. We derive bounds on the size of WMU and various constrained WMU codes and present a number of constructions for balanced, error-correcting, primer-dimer free WMU codes using Dyck paths, prefixsynchronized and cyclic codes.

Second, we describe the first DNA-based storage architecture that enables random access to data blocks and rewriting of information stored at arbitrary
locations within the blocks. The newly developed architecture overcomes drawbacks of existing read-only methods that require decoding the whole file in order to read one data fragment. Our system is based on the newly developed WMU coding techniques and accompanying DNA editing methods that ensure data reliability, specificity and sensitivity of access, and at the same time provide exceptionally high data storage capacity. As a proof of concept, we encoded parts of the Wikipedia pages of six universities in the USA, and selected and edited parts of the text written in DNA corresponding to three of these schools. The results suggest that DNA is a versatile media suitable for both ultrahigh density archival and rewritable storage applications.

Third, we demonstrate for the first time that a portable, random-access platform may be implemented in practice using nanopore sequencers. Every solution for DNA-based data storage systems so far has exclusively focused on Illumina sequencing devices, but such sequencers are expensive and designed for laboratory use only. Instead, we propose using a new technology, MinION-Oxford Nanopore's handheld sequencer. Nanopore sequencing is fast and cheap, but it results in reads with high error rates. To deal with this issue, we designed an integrated processing pipeline that encodes data to avoid costly synthesis and sequencing errors, enables random access through addressing, and leverages efficient portable sequencing via new iterative alignment and deletion error-correcting codes. As a proof of concept, we stored and sequenced around 3.6 kB of binary data that includes two compressed images (a Citizen Kane poster and a smiley face emoji), using a portable data storage system, and obtained error-free read-outs.

To my lovely parents
Maman Maryam $\mathcal{E B}^{\mathcal{J} \text { Baba Mohsen, for their love and support }}$ and my beloved siblings
Fatemeh, Sadegh, Javad and Reyhaneh.

## ACKNOWLEDGMENTS

Towards this truly exciting accomplishment in my life, my most sincere gratitude goes to my advisor, Professor Olgica Milenkovic, for constant and generous support and guidance during my education at UIUC. Beside her thoughtful ideas and exceptional knowledge of the field I would like to mostly appreciate her uniquely outstanding style of research. Her mind-provoking questions and thoughtful discussions have greatly influenced my thought processes during the completion of my doctoral thesis. She has taught me the true meaning of scientific research. Also, her creative approach to teaching and giving engaging presentations was particularly enlightening.

I profusely thank Doctor Alvaro Hernandez for sharing his knowledge and experience with me. My gratitude to Huimin Zhao for his great comments and suggestions since the beginning of my doctoral studies. His discussions and comments have greatly improved the quality of my work.

I would like to also extend my gratitude to Professor Jian Ma for his comments and suggestions on my research.

I also thank my fellows and colleagues at UIUC, especially Ryan Gabrys, Han Mao Kiah, Vida Ravanmehr, Nasir Paravi, Christopher Fields and Minji Kim. Also, special thanks to my friend Pooyan Kabir for his wonderful comments and suggestions.

Finally, I would like to extend my deepest gratitude, love and affection to my beloved family, for loving me, believing in me and wishing the best for me. I owe all my achievements to their pure hearts; such pure hearts that anything they dream for me comes true.

## CONTENTS

Chapter 1 INTRODUCTION ..... 1
1.1 Motivation ..... 1
1.2 Contributions of This Dissertation ..... 2
Chapter 2 MUTUALLY UNCORRELATED PRIMERS FOR DNA- BASED DATA STORAGE ..... 5
2.1 Introduction ..... 5
2.2 Roadmap of Approaches and Results ..... 7
2.3 MU and WMU Codes: Definitions, Bounds and Constructions ..... 10
2.4 Error-Correcting WMU Codes ..... 18
2.5 Balanced $\kappa$-WMU Codes ..... 26
2.6 APD-MU Codes ..... 29
2.7 APD, Balanced, Error-Correcting and WMU Codes ..... 31
2.8 Information Encoding with WMU Addresses ..... 35
2.9 Conclusions ..... 38
Chapter 3 A REWRITABLE, RANDOM-ACCESS DNA-BASED STORAGE SYSTEM ..... 39
3.1 Introduction ..... 39
3.2 Results ..... 42
3.3 Methods ..... 45
Chapter 4 PORTABLE AND ERROR-FREE DNA-BASED DATA STORAGE ..... 56
4.1 Introduction ..... 56
4.2 The Encoding Step ..... 58
4.3 The Postprocessing Step ..... 59
4.4 System Implementation ..... 63
4.5 Supplementary Information ..... 64
BIBLIOGRAPHY ..... 89

## Chapter 1

## INTRODUCTION

### 1.1 Motivation

DNA digital data storage refers to any scheme to store digital data in the base sequence of DNA.

Among the macromolecules that may potentially be used as a storage media, DNA molecules, which may be abstracted as strings over the four symbol alphabet $\{A, T, G, C\}$, stand out due to a number of unique properties:

- DNA has been successfully used as a building block of a number of small scale self-assembly based computers [1].
- DNA lends itself to implementations of non-volatile recoding media of outstanding integrity, as one can still recover the DNA of 30,000 year old Neanderthal and 700,000 year old horse bones [2].
- DNA allows for extremely high storage capacities - a single human cell, with a mass of roughly 3 picograms, hosts DNA strands encoding 6.4 GBs of information.
- The technologies for synthesizing (writing) artificial DNA and for massive sequencing (reading) have reached unprecedented levels of efficiency and accuracy [3].

As a result, DNA-based storage systems may be the most plausible storage platform to materialize in the near future.

Building upon the rapid growth of DNA synthesis and sequencing technologies, two laboratories recently outlined architectures for archival DNA-based storage in $[4,5]$. The first architecture achieved a density of $700 \mathrm{~TB} / \mathrm{gram}$, while the second approach raised the density to $2 \mathrm{~PB} /$ gram. The increase in density achieved by the second method was largely attributed to the use of
four elementary coding schemes: Huffman coding, differential coding, single parity-check coding, and repetition coding. Nevertheless, data fidelity levels required by modern storage systems were not met, nor did the two architectures enable accurate random access and portable system architecture.

### 1.2 Contributions of This Dissertation

The coding techniques used in the two aforementioned architectures [4, 5] are off-the-shelf solutions that do not take into account the special nature of synthesis and sequencing errors. For example, in [5], Huffman coding may lead to catastrophic error propagation in the presence of storage system errors. Furthermore, given the ultrahigh storage density achievable by DNA storage systems, compression may not be needed and it also may prevent efficient information access and rewriting. Differential encoding is only used to avoid errors which arise when encountering long runs of the same DNA symbol, termed homopolymers; but such errors are confined to older sequencing technologies (such as Roche454) which are no longer in widespread use, or to nanopore sequencers, in which case they are coupled with many other types of indel and substitution errors. Finally, single parity-check coding only allows for rudimentary error-detection and cannot actually be used to correct errors; repetition coding is achieved by synthesizing multiple copies of the same subsequences shifted by a fixed number of positions. This shifting ensures what is known in the biology literature as "high sequence coverage," but high coverage is very costly. To overcome the last issue, the more recent work [6] extended the coding approach of [5] by replacing single parity-check codes with Reed-Solomon codes [7].

Another important issue regarding DNA storage systems is the need to enable random access to data. The three architectures which confirmed the plausibility of DNA-based storage did not account for precise random access, which renders them impractical, as the whole DNA content has to be read at once and assembled in order to retrieve the desired DNA blocks. The obvious solution to this problem is to equip DNA blocks with addresses that may aid in accurate access and amplification of desired DNA strings without perturbing other blocks in the DNA pool. Unlike in classical storage systems, addresses are "recognized" through Watson-Crick complementary bonding
between DNA strings in a process that is known as $D N A$ hybridization. As an illustrative example, in order to select a block addressed by $A A A G C C T$, one would use a primer $A G G C T T T$ (the reverse complement of the address) to amplify the strand that contains the address $A A A G C C T$. In other words, selection reduces to exponential amplification of the desired string. Clearly, this approach may fail if other addresses are "hybridization similar" to $A A A G C C T$ - a simple example being the string $T G G C T T A$ which only differs from the true selection primer $A G G C T T T$ in the first and last position. The problem is further aggravated by the fact that undesired selection and amplification may occur between address primers and substrings in the information blocks themselves.

To mitigate these problems, in Chapter 2 we design addresses that satisfy a number of constraints and encode information in a manner that prevents undesired selection. To accomplish this task, novel coding solutions are needed, including generalizations of prefix-synchronized coding [8-10], running digital sum coding [11, 12], and autocorrelation analysis [13]. We hence study the problem of Enumerating and constructing addresses that satisfy constant GC content, Hamming distance and mutual uncorrelatedness constraints.

In molecular biology and genetics, GC-content is the percentage of bases on a DNA string that are either G or C . Here, the constant $G C$ content is imposed on sufficiently long prefixes of the addresses, for reasons that will be discuss in Chapter 2 and Chapter 3. Constant composition codes under the Hamming metric have been studied in depth in the literature, but the novelty of the problem comes from requiring the words to be mutually uncorrelated. A set of addresses $\mathcal{A}=\left\{\mathbf{a}_{1}, \ldots, \mathbf{a}_{M}\right\}$, each of length $n$, is termed to be mutually uncorrelated if for any two, not necessarily distinct, addresses $\mathbf{a}_{i}, \mathbf{a}_{j} \in \mathcal{A}$, no prefix of $\mathbf{a}_{i}$ of length $\leq n-1$ appears as a suffix of $\mathbf{a}_{j}$. Note that long undesired prefix-suffix matches may lead to assembly errors in blocks during joint sequencing and more importantly, uncorrelated sequences may be avoided jointly in the information parts of the blocks via simple coding methods we termed address coding. Hence, as part of our thesis research, we found bounds on the size of addresses satisfying the aforementioned constraints and combinatorial construction methods that approach these bounds. We also studied a relaxation of the address coding problem in which the prefix-suffix constraints are only imposed on substrings of sufficient length. We term such sequences weakly mutually uncorrelated. In addition, we coupled the above
constraints with secondary structure constraints [14]. The term secondary structure is used to describe a folded formation that results from hybridization of Watson-Crick complementary substrings on the same single-stranded DNA sequence. Secondary structures in primers are known to inhibit their function or cause errors in the process of PCR amplification and fragment rewriting [15].

To overcome random access and rewriting issues, in Chapter 3 we propose a (hybrid) DNA rewritable storage architecture with random access capabilities. The new DNA-based storage scheme encompasses a number of coding features, including constrained coding, ensuring that DNA patterns prone to sequencing errors are avoided; prefix synchronized coding ensures that blocks of DNA may be accurately accessed without perturbing other blocks in the DNA pool. We use the result from Chapter 2 to design WMU address sequences, and once they are constructed, to encode user information into codewords that avoid any of the addresses, sufficiently long substrings of the addresses, or substrings "similar" to the addresses.

In the past few years, there has been a rapid growth of new generation of portable sequencing devices using nanopore technology. In Chapter 4 we introduce and design the first DNA storage platform that offers error-free and random-access readouts from a portable device. We show through experimental and theoretical verification that such a platform may be easily implemented in practice. The gist of the approach is an integrated bioinformatics and coding-theoretics pipeline that includes new anchored iterative alignment techniques and insertion/deletion error-correcting codes. Our work represents the only known random access DNA-based data storage system that uses error-prone MinION sequencers and produces error-free readouts.

Parts of the dissertation results were published in [16-18].

## Chapter 2

# MUTUALLY UNCORRELATED PRIMERS FOR DNA-BASED DATA STORAGE 

### 2.1 Introduction

Mutually uncorrelated (MU) codes are a class of fixed length block codes in which no proper prefix of one codesequence is a suffix of the same or another codesequence. MU codes have been extensively studied in the coding theory and combinatorics literature under a variety of names. Levenshtein introduced the codes in 1964 under the name 'strongly regular codes' [19], and suggested that the codes be used for synchronization. For frame synchronization applications described by van Wijngaarden and Willink in [20], Bajić and Stojanović [21] rediscovered MU codes, and studied them under the name of 'cross-bifix-free' codes. Constructions and bounds on the size of MU codes were also reported in a number of recent contributions [22,23]. In particular, Blackburn [23] analyzed these sequences under the name of 'nonoverlapping codes', and provided a simple construction for a class of codes with optimal cardinality.

MU codes have recently found new applications in DNA-based data storage [4,5]: In this setting, Yazdi et al. [17,18] developed a new, random-access and rewritable DNA-based data storage architecture that uses MU address sequences that allow selective access to encoded DNA blocks via Polymerase Chain Reaction (PCR) amplification with primers complementary to the address sequences. In a nutshell, DNA information-bearing sequences are prepended with address sequences used to access strings of interest via PCR amplification. To jump start the amplification process, one needs to 'inject' complements of the sequences into the storage system, and those complementary sequences are referred to as DNA primers. Primers attach themselves to the user-selected address strings and initiate the amplification reaction. In order to ensure accurate selection and avoid expensive postprocessing, the
information sequences following the address are required to avoid sequences that resemble the addresses, thereby imposing a special coding constraint that may be met through the use of MU addresses. In addition, the addressing scheme based on MU codes may be used in conjunction with other specialized DNA-based data storage codes like the ones outlined in [24-26]. Detailed descriptions of implementations of DNA-based data storage systems and their underlying synthetic biology principles are mentioned in Chapter 3; the interested reader is referred to [27] for a discussion of system components and constraints.

The goal of this work is to generalize the family of MU codes by introducing weakly mutually uncorrelated (WMU) codes. WMU codes are block codes in which no sufficiently long prefix of one codesequence is a suffix of the same or another codesequence. In contrast, MU codes prohibit suffixprefix matches of any length. This relaxation of prefix-suffix constraints was motivated in [17], with the purpose of improving code rates and allowing for increased precision DNA fragment assembly and selective addressing. A discussion of the utility of WMU codes in DNA-based data storage may be found in the overview paper [18,27] and the paper describing recent practical implementations of portable DNA-based data storage systems which make use of WMU codes [18].

Here, we are concerned with determining bounds on the size of specialized WMU codes and efficient WMU code constructions. Of interest are both binary and quaternary WMU codes, as the former may be used to construct the latter, while the latter class may be adapted for encoding over the four letter DNA alphabet $\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$. Our contributions include bounds on the largest size of unconstrained and constrained WMU codes, constructions of WMU codes that meet the derived upper bounds as well as results on several important constrained versions of WMU codes: Error-correcting WMU codes, balanced WMU codes, balanced error-correcting WMU codes, and WMU codes that avoid primer-dimer byproducts. The aforementioned constraints arise due to the following practical considerations.

A binary sequence is called balanced if half of its symbols are zero. On the other hand, a DNA sequence is termed balanced if it has a $50 \% \mathrm{GC}$ content (i.e., if $50 \%$ of the symbols in the sequence are either G or C). Balanced DNA sequences are more stable than DNA sequences with lower or higher GC content and they have lower sequencing error-rates. Balanced

DNA sequences are also easier to synthesize than unbalanced sequences [28]. In addition, WMU codes at large Hamming distance limit the probability of erroneous codesequence selection dues to address errors. When referring to primer dimer (PD) issues [29], we consider potential problems that may arise during random access when two primers used for selection bond to each other, thereby prohibiting amplification of either of the two corresponding information-bearing sequences. PD byproducts can be eliminated by restricting the WMU codes to avoid simultaneous presence of long substrings and their complements in the codesequences.

This chapter is organized as follows. Section 2.2 contains an overview of the topics and results discussed in the chapter and some formal definitions needed to follow the material in subsequent sections. In Section 2.3 we review MU and introduce WMU codes, and derive bounds on the maximum size of the latter family of combinatorial objects. In addition, we outline a construction of WMU codes that meets the derived upper bound. We also describe a construction that uses binary MU component codes and other constrained codes in order to obtain families of WMU codes that obey different combinations of primer constraints. In Section 2.4 we describe constructions for error-correcting WMU codes, while in Section 2.5 we discuss balanced WMU codes. Primer-dimer constraints are discussed in Section 2.6. Our main results are presented in Section 2.7, where we first propose to use cyclic codes to devise WMU codes that are both balanced and have error correcting capabilities. We then proceed to improve the cyclic code construction in terms of coding rate through decoupled constrained and error-correcting coding for binary strings. In this setting, we use DC-balanced codes [30]. Encoding of information with WMU address codes is described in Section 2.8.

### 2.2 Roadmap of Approaches and Results

Throughout the chapter we use the following notation: $\mathbb{F}_{q}$ stands for a finite field of order $q \geq 2$. Two cases of special interest are $q=2$ and $q=4$. In the latter case, we tacitly identify the elements of $\mathbb{F}_{4}$ with the four letters of the DNA code alphabet, $\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$. We let $\mathbf{a}=\left(a_{1}, \ldots, a_{n}\right) \in \mathbb{F}_{q}^{n}$ stand for a sequence of length $n$ over $\mathbb{F}_{q}$, and let $\mathbf{a}_{i}^{j}, 1 \leq i, j \leq n$, stand for a substring
of a starting at position $i$ and ending at position $j$, i.e.,

$$
\mathbf{a}_{i}^{j}= \begin{cases}\left(a_{i}, \ldots, a_{j}\right) & i \leq j \\ \left(a_{i}, a_{i-1}, \ldots, a_{j}\right) & i>j\end{cases}
$$

Moreover, for two arbitrary sequences $\mathbf{a} \in \mathbb{F}_{q}^{n}, \mathbf{b} \in \mathbb{F}_{q}^{m}$, we use $\mathbf{a b}=\left(a_{1}, \ldots, a_{n}, b_{1}, \ldots, b_{m}\right)$ to denote a sequence of length $n+m$ generated by appending $\mathbf{b}$ to the right of $\mathbf{a}$. Thus, $\mathbf{a}^{l}$ stands for a sequence of length $l n$ comprising $l$ consecutive copies of the sequence a.

We say that a sequence $\overline{\mathbf{a}}=\left(\overline{a_{1}}, \ldots, \overline{a_{n}}\right) \in \mathbb{F}_{q}^{n}$ represents the complement of sequence $\mathbf{a} \in \mathbb{F}_{q}^{n}$ if:

- For $q=2$, and $1 \leq i \leq n$,

$$
\bar{a}_{i}= \begin{cases}1 & \text { if } a_{i}=0  \tag{2.1}\\ 0 & \text { if } a_{i}=1\end{cases}
$$

- For $q=4$, and $1 \leq i \leq n$,

$$
\bar{a}_{i}= \begin{cases}\mathrm{T} & \text { if } a_{i}=\mathrm{A},  \tag{2.2}\\ \mathrm{~A} & \text { if } a_{i}=\mathrm{T}, \\ \mathrm{G} & \text { if } a_{i}=\mathrm{C}, \\ \mathrm{C} & \text { if } a_{i}=\mathrm{G}\end{cases}
$$

The notion of complement used for $\mathbb{F}_{4}$ is often referred to as the Watson-Crick (W-C) complement.

In this work, we define an (address) code $\mathcal{C}$ of length $n$ as a collection of sequences from $\mathbb{F}_{q}^{n}$, for $q \in\{2,4\}$, satisfying a set of specific combinatorial constraints described below.

The goal is to describe new constructions for address sequences used for DNA-based data storage. Address sequences should enable reliable access to desired information content. This is accomplished by making the addresses as distinguishable from each other as possible via a simple minimum Hamming distance constraint; recall that the Hamming distance $d_{H}$ between any two sequences of length $n, \mathbf{a}=\left(a_{1}, \ldots, a_{n}\right)$ and $\mathbf{b}=\left(b_{1}, \ldots, b_{n}\right)$, over some finite alphabet $\mathcal{A}$ equals

$$
d_{H}(\mathbf{a}, \mathbf{b})=\sum_{i=1}^{n} \mathbb{1}\left(a_{i} \neq b_{i}\right),
$$

where $\mathbb{1}(\cdot)$ stands for the indicator function. One may also use the Levenshtein distance instead, as discussed in the context of MU codes in [31].

Access to desired sequences is accomplished by exponentially amplifying them within the pool of all sequences via addition of primer sequences corresponding to the W-C complement of their addresses. As primers have to be synthesized, they need to satisfy constraints that enable simplified synthesis, such as having a balanced GC-content, formally defined for a sequence a over $\mathbb{F}_{4}^{n}$ as $\sum_{i=1}^{n} \mathbb{1}\left(a_{i} \in\{\mathrm{G}, \mathrm{C}\}\right)=\frac{n}{2}$. This constraint directly translates to a balancing property for the address sequences. Furthermore, as one may require simultaneous amplification of multiple sequences, multiple primers need to be added in which case it is undesirable for different pairs of primers to bond to each other via W-C complementarity. The PD byproducts of this binding may be significantly reduced if one imposes an additional PD constraint on the primers, and hence on the address sequences, as defined below.

Definition 1. A set of sequences $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$, for $q \in\{2,4\}$, is said to avoid primer dimer (APD) byproducts of effective length $f$ if substrings of sequences in $\mathcal{C}$ with length $\geq f$ cannot hybridize with each other in the forward or the reverse direction. More precisely, we say that $\mathcal{C}$ is an $f$-APD code if for any two sequences $\mathbf{a}, \mathbf{b} \in \mathcal{C}$, not necessarily distinct, and $1 \leq i, j \leq n+1-f$, we have $\overline{\mathbf{a}}_{i}^{f+i-1} \neq \mathbf{b}_{j}^{f+j-1}, \mathbf{b}_{f+j-1}^{j}$. We refer to the sequence $\mathbf{b}_{f+j-1}^{j}$ as the reverse of the sequence $\mathbf{b}_{j}^{f+j-1}$.

For practical reasons, we only focus on the parameter regime $f=\Theta(n)$, as only sufficiently long complementary sequences may bond with each other. Furthermore, we defer the study of the related problem of secondary structure formation $[14,32]$ to future work.

In certain DNA-based data storage systems, one may be interested in restricting the address sequences by imposing only one or two of the above constraints. For example, if the addresses are relatively short $(\leq 10)$, one may dispose of the requirement to make the sequences balanced, as short sequences are significantly easier to synthesize than longer ones. If one allows for postprocessing of the readouts, then the Hamming distance constraint may be relaxed or completely removed. It is for this reason that we also consider a more general class of code constructions that accommodate only a subset of the three previously described constraints.

By far the most important constraint imposed on the address sequences
is that they enable a simple construction of information-bearing sequences (assumed to be of length $N \gg n$ ) that do not contain any of the address sequences of length $n$ as substrings. It is in this context of forbidden substring coding that MU codes were introduced in $[8,13]$. WMU codes may be used in the same setting, but they are less restrictive than MU codes, and therefore allow for larger codebooks. This is why our main results pertain to constructions of WMU codes with various subsets of primer constraints, and we formally define and discuss these codes in the next section. For some related questions pertaining to MU codes, the interested reader is referred to [31].

### 2.3 MU and WMU Codes: Definitions, Bounds and Constructions

For simplicity of notation, we adopt the following naming convention for codes: If a code $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ has properties Property $_{1}$, Property $_{2}, \ldots$, Property $_{s}$, then we say that $\mathcal{C}$ is a Property $_{1} \_$Property $_{2} \ldots \ldots$, Property $_{\text {s_q_ }}$ n code, and use the previous designation in the subscript.

### 2.3.1 Mutually Uncorrelated Codes

We say that a sequence $\mathbf{a}=\left(a_{1}, \ldots, a_{n}\right) \in \mathbb{F}_{q}^{n}$ is self-uncorrelated if no proper prefix of a matches its suffix, i.e., if $\left(a_{1}, \ldots, a_{i}\right) \neq\left(a_{n-i+1}, \ldots, a_{n}\right)$, for all $1 \leq i<n$. This definition may be extended to a set of sequences as follows: Two not necessarily distinct sequences $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{q}^{n}$ are said to be mutually uncorrelated if no proper prefix of $\mathbf{a}$ appears as a suffix of $\mathbf{b}$ and vice versa. We say that $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ is a mutually uncorrelated (MU) code if any two not necessarily distinct codesequences in $\mathcal{C}$ are mutually uncorrelated.

The maximum cardinality of MU codes was determined up to a constant factor by Blackburn [23, Theorem 8]. For completeness, we state the modified version of this result for alphabet size $q \in\{2,4\}$ below:

Theorem 1. Let $A_{\text {MU_q_n }}$ denote the maximum size of a MU_q_n code, with
$n \geq 2$ and $q \in\{2,4\}$. Then

$$
c_{q} \frac{q^{n}}{n} \leq A_{\mathrm{MU} \_\mathrm{q}_{-} \mathrm{n}} \leq \frac{q^{n}}{2 n}
$$

where $c_{q}=\frac{(q-1)^{2}(2 q-1)}{4 q^{4}}$, which for $q=2$ and $q=4$ equal $c_{2}=0.04688$ and $c_{4}=0.06152$, respectively.

We also briefly outline two known constructions of MU codes, along with a new and simple construction for error-correcting MU codes that will be used in our subsequent derivations.

Bilotta et al. [22] described an elegant construction for MU codes based on well-known combinatorial objects termed Dyck sequences. A Dyck sequence of length $n$ is a binary sequence composed of $\frac{n}{2}$ zeros and $\frac{n}{2}$ ones such that no prefix of the sequence has more zeros than ones. By definition, a Dyck sequence is balanced and it necessarily starts with a one and ends with a zero. The number of Dyck word of length $n$ is the $\frac{n}{2}$-th Catalan number, equal to $\frac{2}{n+2}\binom{n}{\frac{n}{2}}$.

Construction 1. (BAL_MU_2_n Codes) Consider a set $\mathcal{D}$ of Dyck sequences of length $n-2$ and define the following set of sequences of length $n$ :

$$
\mathcal{C}=\{1 \mathbf{a} 0: \mathbf{a} \in \mathcal{D}\} .
$$

It is straightforward to show that $\mathcal{C}$ is balanced and MU code. Size of $\mathcal{C}$ is also equal to $\frac{n-2}{2}$-th Catalan number, or $|\mathcal{C}|=\frac{1}{2(n-1)}\binom{n}{\frac{n}{2}}$.

An important observation is that MU codes constructed using Dyck sequences are inherently balanced, as they contain $\frac{n}{2}$ ones and $\frac{n}{2}$ zeros. The balancing property also carries over to all prefixes of certain subsets of Dyke sequences. To see this, recall that a Dyck sequence has height at most $D$ if for any prefix of the sequence, the difference between the number of ones and the number of zeros is at most $D$. Hence, the disbalance of any prefix of a Dyck sequence of height $D$ is at most $D$. Let $\operatorname{Dyck}(n, D)$ denote the number of Dyck sequences of length $2 n$ and height at most $D$. For fixed values of $D$, de Bruijn et al. [33] proved that

$$
\begin{equation*}
\operatorname{Dyck}(n, D) \sim \frac{4^{n}}{D+1} \tan ^{2}\left(\frac{\pi}{D+1}\right) \cos ^{2 n}\left(\frac{\pi}{D+1}\right) \tag{2.3}
\end{equation*}
$$

Here, $f(n) \sim g(n)$ is used to denote the following asymptotic relation $\lim _{m \rightarrow \infty} f(n) / g(n)=1$.

Bilotta's construction also produces nearly prefix-balanced MU codes, provided that one restricts his/her attention to subsets of sequences with small disbalance $D$; equation 2.3 establishes the existence of large subsets of Dyck sequences with small disbalance. By mapping 0 and 1 to $\{\mathrm{A}, \mathrm{T}\}$ and $\{\mathrm{C}, \mathrm{G}\}$, respectively, one may enforce a similar GC balancing constraint on DNA MU codes.

The next construction of MU codes was proposed by Levenshtein [19] and Gilbert [8].

Construction 2. (MU_q_n Codes) Let $n \geq 2$ and $1 \leq \ell \leq n-1$, be two integers and let $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ be the set of all sequences $\mathbf{a}=\left(a_{1}, \ldots, a_{n}\right)$ such that

- The sequence a starts with $\ell$ consecutive zeros, i.e., $\mathbf{a}_{1}^{\ell}=0^{\ell}$.
- It holds that $a_{\ell+1}, a_{n} \neq 0$.
- The subsequence $\mathbf{a}_{\ell+2}^{n-1}$ does not contain $\ell$ consecutive zeros as a subsequence.

Then, $\mathcal{C}$ is an MU code. Blackburn [23, Lemma 3] showed that when $\ell=\left\lceil\log _{q} 2 n\right\rceil$ and $n \geq 2 \ell+2$ the above construction is optimal. His proof relies on the observation that the number of strings $\mathbf{a}_{\ell+2}^{n-1}$ that do not contain $\ell$ consecutive zeros as a subsequence exceeds $\frac{(q-1)^{2}(2 q-1)}{4 n q^{4}} q^{n}$, thereby establishing the lower bound of Theorem 1. The aforementioned result is a simple consequence of the following lemma.

Lemma 1. The number of $q$-ary sequences of length $n$ that avoid $t$ specified sequences in $\mathbb{F}_{q}^{n_{s}}$ as substrings is greater than $q^{n}\left(1-\frac{n t}{q^{n_{s}}}\right)$.

Proof. The result obviously holds for $n \leq n_{s}$. If $n \geq n_{s}$, then the number of bad strings, i.e., $q$-ary strings of length $n$ that contain at least one of the specified $t$ strings as a substring, is bounded from above by:

$$
\begin{aligned}
\# \text { bad strings } & \leq\left(n-n_{s}+1\right) t q^{n-n_{s}} \\
& \leq n t q^{n-n_{s}} .
\end{aligned}
$$

Hence, the number of good sequences, i.e., the number of $q$-ary sequences of length $n$ that avoid $t$ specified strings in $\mathbb{F}_{q}^{n_{s}}$ as substrings, is bounded from
below by

$$
\begin{aligned}
\# \text { good strings } & \geq q^{n}-\# \text { bad strings } \\
& \geq q^{n}\left(1-\frac{n t}{q^{n_{s}}}\right)
\end{aligned}
$$

It is straightforward to modify Construction 2 so as to incorporate errorcorrecting redundancy. Our constructive approach to this problem is outlined in what follows.

Construction 3. (Error - Correcting_MU_2_n Codes) Fix two positive integers $t$ and $\ell$ and consider a binary $\left(n_{H}, s, d\right)$ code $\mathcal{C}_{H}$ of length $n_{H}=$ $t(\ell-1)$, dimension $s$, and Hamming distance $d$. For each codesequence $\mathbf{b} \in \mathcal{C}_{H}$, we map $\mathbf{b}$ to a sequence of length $n=(t+1) \ell+1$ given by

$$
\mathbf{a}(\mathbf{b})=0^{\ell} 1 \mathbf{b}_{1}^{\ell-1} 1 \mathbf{b}_{\ell}^{2(\ell-1)} 1 \cdots \mathbf{b}_{(t-1)(\ell-1)+1}^{t(\ell-1)} 1 .
$$

Let $\mathcal{C}_{\text {parse }} \triangleq\left\{\mathbf{a}(\mathbf{b}): \mathbf{b} \in \mathcal{C}_{H}\right\}$.
It is easy to verify that $\left|\mathcal{C}_{\text {parse }}\right|=\left|\mathcal{C}_{H}\right|$, and that the code $\mathcal{C}_{\text {parse }}$ has the same minimum Hamming distance as $\mathcal{C}_{H}$, i.e., $d\left(\mathcal{C}_{\text {parse }}\right)=d\left(\mathcal{C}_{H}\right)$. As $n_{H}=t(\ell-1)$, we also have $\mathcal{C}_{\text {parse }} \subseteq\{0,1\}^{n}$, where $n=(t+1) \ell+1$. In addition, the parsing code $\mathcal{C}_{\text {parse }}$ is an MU code, since it satisfies all the constraints required by Construction 2. To determine the largest asymptotic size of a parsing code, we recall the Gilbert-Varshamov bound.

Theorem 2. (Asymptotic Gilbert-Varshamov bound [34, 35]) For any two positive integers $n$ and $d \leq \frac{n}{2}$, there exists a block code $\mathcal{C} \subseteq\{0,1\}^{n}$ of minimum Hamming distance $d$ with normalized rate

$$
R(\mathcal{C}) \geq 1-h\left(\frac{d}{n}\right)-o(1)
$$

where $h(\cdot)$ is the binary entropy function, i.e., $h(x)=x \log _{2} \frac{1}{x}+(1-x) \log _{2} \frac{1}{1-x}$, for $0 \leq x \leq 1$.

Recall that the parameters $s$ (dimension) and $d$ (minimum Hamming distance) of the codes $\mathcal{C}_{H}$ and $\mathcal{C}_{\text {parse }}$ are identical. Their lengths, $n_{H}$ and $n$,
respectively, equal $n_{H}=t(\ell-1)$ and $n=(t+1) \ell+1$, where $t, \ell$ are positive integers. We next aim to optimize the parameters of the parsing code for fixed $s$ and fixed $n$, which amounts to maximizing $d$. Since $d$ is equal to the corresponding minimum distance of the code $\mathcal{C}_{H}$, and both codes have the same dimension $s$, in order to maximize $d$ we maximize $n_{H}$ under the constraint that $n$ is fixed. More precisely, we optimize the choice of $\ell, t$ and then use the resulting parameters in the Gilbert-Varshamov lower bound.

To maximize $n_{H}=t(\ell-1)$ given $n=(t+1) \ell+1$ and $t, \ell \geq 1$, we write

$$
n_{H}=n-(\ell+t+1) \leq n-2 \sqrt{\ell(t+1)}=n-2 \sqrt{n-1} .
$$

Here, the inequality follows from the arithmetic and geometric mean inequality, i.e., $\frac{\ell+t+1}{2} \geq \sqrt{\ell(t+1)}$. On the other hand, it is easy to verify that this upper bound is achieved by setting $\ell=\sqrt{n-1}$ and $t=\sqrt{n-1}-1$. Hence, the maximum value of $n_{H}$ is $n_{H}^{*}=n-2 \sqrt{n-1}$.

By using a code $\mathcal{C}_{H}$ with parameters $\left[n_{H}^{*}, s, d\right]$ as specified by the GV bound, where $d \leq \frac{n_{H}^{*}}{2}$ and $s=n_{H}^{*}\left(1-h\left(\frac{d}{n_{H}^{*}}\right)-o(1)\right)$, we obtain an errorcorrecting MU code $\mathcal{C}_{\text {parse }}$ with parameters $\left[n_{H}^{*}+2 \sqrt{n_{H}^{*}+2 \sqrt{n_{H}^{*}-1}-1}, n_{H}^{*}(1-\right.$ $\left.\left.h\left(\frac{d}{n_{H}^{*}}\right)-o(1)\right), d\right]$.

### 2.3.2 Weakly Mutually Uncorrelated Codes: Definitions, Bounds and Constructions

The notion of mutual uncorrelatedness may be relaxed by requiring that only sufficiently long prefixes of one sequence do not match sufficiently long suffixes of the same or another sequence. A formal definition of this property is given next.

Definition 2. Let $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ and $1 \leq \kappa<n$. We say that $\mathcal{C}$ is a $\kappa$-weakly mutually uncorrelated ( $\kappa$-WMU) code if no proper prefix of length $l$, for all $l \geq \kappa$, of a codesequence in $\mathcal{C}$ appears as a suffix of another codesequence, including itself.

Our first result pertains to the size of the largest WMU code.
Theorem 3. Let $A_{\kappa-\text { WMU_q_n }}$ denote the maximum size of a $\kappa$-WMU code
over $\mathbb{F}_{q}^{n}$, for $1 \leq \kappa<n$ and $q \in\{2,4\}$. Then,

$$
c_{q} \frac{q^{n}}{n-\kappa+1} \leq A_{\kappa-\text { WMU_q_n }} \leq \frac{q^{n}}{n-\kappa+1},
$$

where the constant $c_{q}$ is as described in Theorem 1.
Proof. To prove the upper bound, we use an approach first suggested by Blackburn in [23, Theorem 1], for the purpose of analyzing MU codes. Assume that $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ is a $\kappa$-WMU code. Let $L=(n+1)(n-\kappa+1)-1$, and consider the set $X$ of pairs $(\mathbf{a}, i)$, where $i \in\{1, \ldots, L\}$, and where $\mathbf{a} \in \mathbb{F}_{q}^{L}$ is such that the (possibly cyclically wrapped) substrings of a of length $n$ starting at position $i$ belongs to $\mathcal{C}$. Note that our choice of the parameter $L$ is governed by the overlap length $\kappa$.

Clearly, $|X|=L|\mathcal{C}| q^{L-n}$, since there are $L$ different possibilities for the index $i,|\mathcal{C}|$ possibilities for the string starting at position $i$ of $\mathbf{a}$, and $q^{L-n}$ choices for the remaining $L-n \geq 0$ symbols in $\mathbf{a}$. Moreover, if $(\mathbf{a}, i) \in X$, then $(\mathbf{a}, j) \notin X$ for $j \in\{i \pm 1, \ldots, i \pm n-\kappa\}_{\bmod L}$ due to the weak mutual uncorrelatedness property. Hence, for a fixed string $\mathbf{a} \in \mathbb{F}_{q}^{L}$, there are at most $\left\lfloor\frac{L}{n-\kappa+1}\right\rfloor$ different pairs $\left(\mathbf{a}, i_{1}\right), \ldots,\left(\mathbf{a}, i_{\left\lfloor\frac{L}{n-\kappa+1}\right\rfloor}\right) \in X$. This implies that

$$
|X| \leq\left\lfloor\frac{L}{n-\kappa+1}\right\rfloor q^{L}
$$

Combining the two derived constraints on the size of $X$, we obtain

$$
|X|=L|\mathcal{C}| q^{L-n} \leq\left\lfloor\frac{L}{n-\kappa+1}\right\rfloor q^{L} .
$$

Therefore, $|\mathcal{C}| \leq \frac{q^{n}}{n-\kappa+1}$.
To prove the lower bound, we describe a simple WMU code construction, outlined in Construction 4.

Construction 4. ( $\kappa-$ WMU_q_n Codes) Let $\kappa, n$ be two integers such that $1 \leq \kappa \leq n$. A $\kappa$-WMU code $\mathcal{C} \in \mathbb{F}_{q}^{n}$ may be constructed using a simple concatenation of the form $\mathcal{C}=\left\{\mathbf{a b} \mid \mathbf{a} \in \mathcal{C}_{1}, \mathbf{b} \in \mathcal{C}_{2}\right\}$, where $\mathcal{C}_{1} \subseteq \mathbb{F}_{q}^{n-\kappa+1}$ is an MU code, and $\mathcal{C}_{2} \subseteq \mathbb{F}_{q}^{\kappa-1}$ is unconstrained.

It is easy to verify that $\mathcal{C}$ is an $\kappa$-WMU code with $\left|\mathcal{C}_{1}\right|\left|\mathcal{C}_{2}\right|$ codesequences. Let $\mathcal{C}_{2}=\mathbb{F}_{q}^{\kappa-1}$ and let $\mathcal{C}_{1} \subseteq \mathbb{F}_{q}^{n-\kappa+1}$ be the largest MU code of size $A_{\text {MU_q_n }}{ }^{n+\kappa+1}$.

Then, $|\mathcal{C}|=q^{\kappa-1} A_{\text {MU }}^{\text {q_n }-\kappa+1}{ }^{\text {. }}$. The claimed lower bound now follows from the lower bound of Theorem 1, establishing that $|\mathcal{C}| \geq c_{q} \frac{q^{n}}{n-\kappa+1}$.

As described in the Introduction, $\kappa$-WMU codes used in DNA-based storage applications are required to satisfy a number of additional combinatorial constraints in order to be used as blocks addresses. These include the errorcorrecting, balancing and primer dimer constraints. Balancing and errorcorrecting properties of codesequences have been studied in great depth, but not in conjunction with MU or WMU codes. The primer dimer constraint has not been previously considered in the literature.

In what follows, we show that all the above constraints can be imposed on $\kappa$-WMU codes via a simple decoupled binary code construction. To this end, let us introduce a mapping $\Psi$ as follows. For any two binary sequences $\mathbf{a}=\left(a_{1}, \ldots, a_{n}\right), \mathbf{b}=\left(b_{1}, \ldots, b_{n}\right) \in\{0,1\}^{n}, \Psi(\mathbf{a}, \mathbf{b}):\{0,1\}^{n} \times\{0,1\}^{n} \rightarrow$ $\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}^{n}$ is an encoding function that maps the pair $\mathbf{a}, \mathbf{b}$ to a DNA string $\mathbf{c}=\left(c_{1}, \ldots, c_{n}\right) \in\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}^{n}$, according to the following rule:

$$
\text { for } 1 \leq i \leq n, c_{i}= \begin{cases}\mathrm{A} & \text { if }\left(a_{i}, b_{i}\right)=(0,0)  \tag{2.4}\\ \mathrm{T} & \text { if }\left(a_{i}, b_{i}\right)=(0,1) \\ \mathrm{C} & \text { if }\left(a_{i}, b_{i}\right)=(1,0) \\ \mathrm{G} & \text { if }\left(a_{i}, b_{i}\right)=(1,1)\end{cases}
$$

Clearly, $\Psi$ is a bijection and $\Psi(\mathbf{a}, \mathbf{b}) \Psi(\mathbf{c}, \mathbf{d})=\Psi(\mathbf{a c}, \mathbf{b d})$. The next lemma lists a number of useful properties of $\Psi$.

Lemma 2. Suppose that $\mathcal{C}_{1}, \mathcal{C}_{2} \subseteq\{0,1\}^{n}$ are two binary block codes of length $n$. Encode pairs of codesequences $(\mathbf{a}, \mathbf{b}) \in \mathcal{C}_{1} \times \mathcal{C}_{2}$ into a code $\mathcal{C}=$ $\left\{\Psi(\mathbf{a}, \mathbf{b}) \mid \mathbf{a} \in \mathcal{C}_{1}, \mathbf{b} \in \mathcal{C}_{2}\right\}$. Then:
(i) If $\mathcal{C}_{1}$ is balanced, then $\mathcal{C}$ is balanced.
(ii) If either $\mathcal{C}_{1}$ or $\mathcal{C}_{2}$ are $\kappa$-WMU codes, then $\mathcal{C}$ is also an $\kappa$-WMU code.
(iii) If $d_{1}$ and $d_{2}$ are the minimum Hamming distances of $\mathcal{C}_{1}$ and $\mathcal{C}_{2}$, respectively, then the minimum Hamming distance of $\mathcal{C}$ is at least $\min \left(d_{1}, d_{2}\right)$.
(iv) If $\mathcal{C}_{2}$ is an $f$ - APD code, then $\mathcal{C}$ is also an $f$ - APD code.

Proof. (i) Any $\mathbf{c} \in \mathcal{C}$ may be written as $\mathbf{c}=\Psi(\mathbf{a}, \mathbf{b})$, where $\mathbf{a} \in \mathcal{C}_{1}, \mathbf{b} \in \mathcal{C}_{2}$. According to (2.4), the number of G, C symbols in $\mathbf{c}$ equals the number of ones in a. Since a is balanced, exactly half of the symbols in $\mathbf{c}$ are Gs and Cs. This implies that $\mathcal{C}$ has balanced GC content.
(ii) We prove the result by contradiction. Suppose that $\mathcal{C}$ is not a $\kappa$-WMU code while $\mathcal{C}_{1}$ is a $\kappa$-WMU code. Then, there exist sequences $\mathbf{c}, \mathbf{c}^{\prime} \in \mathcal{C}$ such that a proper prefix of $\mathbf{c}$ of length at least $\kappa$ appears as a suffix of $\mathbf{c}^{\prime}$. Alternatively, there exist sequences $\mathbf{p}, \mathbf{c}_{0}, \mathbf{c}_{0}^{\prime}$ such that $\mathbf{c}=\mathbf{p c}_{0}, \mathbf{c}^{\prime}=\mathbf{c}_{0}^{\prime} \mathbf{p}$ and the length of $\mathbf{p}$ is at least $\kappa$. Next, we use the fact $\Psi$ is a bijection and find binary strings $\mathbf{a}, \mathbf{b}, \mathbf{a}_{0}, \mathbf{b}_{0}$ such that

$$
\mathbf{p}=\Psi(\mathbf{a}, \mathbf{b}), \mathbf{c}_{0}=\Psi\left(\mathbf{a}_{0}, \mathbf{b}_{0}\right), \mathbf{c}_{0}^{\prime}=\Psi\left(\mathbf{a}_{0}^{\prime}, \mathbf{b}_{0}^{\prime}\right) .
$$

Therefore,

$$
\begin{aligned}
& \mathbf{c}=\mathbf{p c}_{0}=\Psi(\mathbf{a}, \mathbf{b}) \Psi\left(\mathbf{a}_{0}, \mathbf{b}_{0}\right)=\Psi\left(\mathbf{a a}_{0}, \mathbf{b b}_{0}\right), \\
& \mathbf{c}^{\prime}=\mathbf{c}_{0}^{\prime} \mathbf{p}=\Psi\left(\mathbf{a}_{0}^{\prime}, \mathbf{b}_{0}^{\prime}\right) \Psi(\mathbf{a}, \mathbf{b})=\Psi\left(\mathbf{a}_{0}^{\prime} \mathbf{a}, \mathbf{b}_{0}^{\prime} \mathbf{b}\right),
\end{aligned}
$$

where $\mathbf{a a}_{0}, \mathbf{a}_{0}^{\prime} \mathbf{a} \in \mathcal{C}_{1}$. This implies that the string $\mathbf{a}$ of length at least $\kappa$ appears both as a proper prefix and suffix of two not necessarily distinct elements of $\mathcal{C}_{1}$. This contradicts the assumption that $\mathcal{C}_{1}$ is a $\kappa$-WMU code. The same argument may be used for the case that $\mathcal{C}_{2}$ is a $\kappa$-WMU code.
(iii) For any two distinct sequences $\mathbf{c}, \mathbf{c}^{\prime} \in \mathcal{C}$ there exist $\mathbf{a}, \mathbf{a}^{\prime} \in \mathcal{C} 1, \mathbf{b}, \mathbf{b}^{\prime} \in \mathcal{C}_{2}$ such that $\mathbf{c}=\Psi(\mathbf{a}, \mathbf{b}), \mathbf{c}^{\prime}=\Psi\left(\mathbf{a}^{\prime}, \mathbf{b}^{\prime}\right)$. The Hamming distance between $\mathbf{c}, \mathbf{c}^{\prime}$ equals

$$
\begin{aligned}
\sum_{1 \leq i \leq n} \mathbb{1}\left(c_{i} \neq c_{i}^{\prime}\right) & =\sum_{1 \leq i \leq n} \mathbb{1}\left(a_{i} \neq a_{i}^{\prime} \vee b_{i} \neq b_{i}^{\prime}\right) \\
& \geq\left\{\begin{array}{ll}
d_{1} & \text { if } \mathbf{a} \neq \mathbf{a}^{\prime} \\
d_{2} & \text { if } \mathbf{b} \neq \mathbf{b}^{\prime}
\end{array} \geq \min \left(d_{1}, d_{2}\right) .\right.
\end{aligned}
$$

This proves the claimed result.
(iv) By combining (2.1), (4.1) and (2.4), one can easily verify that $\overline{\Psi(\mathbf{a}, \mathbf{b})}=$ $\Psi(\mathbf{a}, \overline{\mathbf{b}})$. We again prove the result by contradiction. Suppose that $\mathcal{C}$
is not an $f$-APD code. Then, there exist $\mathbf{c}, \mathbf{c}^{\prime} \in \mathcal{C}, \mathbf{a}, \mathbf{a}^{\prime} \in \mathcal{C}_{1}, \mathbf{b}, \mathbf{b}^{\prime} \in \mathcal{C}_{2}$ such that $\mathbf{c}=\Psi(\mathbf{a}, \mathbf{b}), \mathbf{c}^{\prime}=\Psi\left(\mathbf{a}^{\prime}, \mathbf{b}^{\prime}\right)$ and $\overline{\mathbf{c}}_{i}^{f+i-1}=\left(\mathbf{c}^{\prime}\right)_{j}^{f+j-1}$ or $\left(\mathbf{c}^{\prime}\right)_{f+j-1}^{j}$, for some $1 \leq i, j \leq n+1-f$. This implies that $\overline{\mathbf{b}}_{i}^{f+i-1}=\left(\mathbf{b}^{\prime}\right)_{j}^{f+j-1}$ or $\left(\mathbf{b}^{\prime}\right)_{f+j-1}^{j}$, which contradicts the assumption that $\mathcal{C}_{2}$ is an $f$-APD code.

In the next sections, we devote our attention to establishing bounds on the size of WMU codes with error-correction, balancing and primer dimer constraints, and to devising constructions that use the decoupling principle or more specialized methods that produce larger codebooks. As the codes $\mathcal{C}_{1}$ and $\mathcal{C}_{2}$ in the decoupled construction have to satisfy two or more properties in order to accommodate all required constraints, we first focus on families of binary codes that satisfy one or two primer constraints.

### 2.4 Error-Correcting WMU Codes

The decoupled binary code construction result outlined in the previous section indicates that in order to construct an error-correcting $\kappa$-WMU code over $\mathbb{F}_{4}$, one needs to combine a binary error-correcting $\kappa$-WMU code with a classical error-correcting code. To the best of our knowledge, no results are available on error-correcting MU or error-correcting $\kappa$-WMU codes.

We start by establishing lower bounds on the coding rates for error-correcting WMU codes using the constrained Gilbert-Varshamov bound [34, 35].

For $\mathbf{a} \in \mathbb{F}_{q}^{n}$ and an integer $r \geq 0$, let $\mathcal{B}_{\mathbb{F}_{q}^{n}}(\mathbf{a}, r)$ denote the Hamming sphere of radius $r$ centered around $\mathbf{a}$, i.e.,

$$
\mathcal{B}_{\mathbb{F}_{q}^{n}}(\mathbf{a}, r)=\left\{\mathbf{b} \in \mathbb{F}_{q}^{n} \mid d_{H}(\mathbf{a}, \mathbf{b}) \leq r\right\}
$$

where, as before, $d_{H}$ denotes the Hamming distance. Clearly, the cardinality of $\mathcal{B}_{\mathbb{F}_{q}^{n}}(\mathbf{a}, r)$ equals

$$
\mathcal{V}_{q}(n, r)=\sum_{i=0}^{r}\binom{n}{i}(q-1)^{i}
$$

independently of the choice of the center of the sphere. For the constrained version of Gilbert-Varshamov bound, let $X \subseteq \mathbb{F}_{q}^{n}$ denote an arbitrary subset
of $\mathbb{F}_{q}^{n}$. For a sequence $\mathbf{a} \in X$, define the Hamming ball of radius $r$ in $X$ by

$$
\mathcal{B}_{X}(\mathbf{a}, r)=\mathcal{B}_{\mathbb{F}_{q}^{n}}(\mathbf{a}, r) \cap X
$$

The volumes of the spheres in $X$ may depend on the choice of $\mathbf{a} \in X$. Of interest is the maximum volume of the spheres of radius $r$ in $X$,

$$
\mathcal{V}_{X, \max }(r)=\max _{\mathbf{a} \in X}\left|\mathcal{B}_{X}(\mathbf{a}, r)\right|
$$

The constrained version of the GV bound asserts that there exists a code of length $n$ over $X$, with minimum Hamming distance $d$ that contains

$$
M \geq \frac{|X|}{\mathcal{V}_{X, \max }(d-1)}
$$

codesequences. Based on the constrained GV bound, we can establish the following lower bound for error-correcting WMU codes. The key idea is to use a $\kappa$-WMU subset as the ground set $X \subseteq \mathbb{F}_{q}^{n}$.

Theorem 4. (Lower bound on the maximum size of error-correcting WMU codes.) Let $\kappa$ and $n$ be two integers such that $n-\kappa-1 \geq 2 \ell$, for $\ell=$ $\left\lceil\log _{q} 2(n-\kappa+1)\right\rceil, q \in\{2,4\}$. Then there exists a $\kappa$-WMU code $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ with minimum Hamming distance $d$ and cardinality

$$
\begin{equation*}
|\mathcal{C}| \geq c_{q} \frac{q^{n}}{(n-\kappa+1)\left(\mathcal{L}_{0}-\mathcal{L}_{1}-\mathcal{L}_{2}\right)}, \tag{2.5}
\end{equation*}
$$

where

$$
c_{q}=\frac{(q-1)^{2}(2 q-1)}{4 q^{4}}
$$

and $\mathcal{L}_{0}, \mathcal{L}_{1}$, and $\mathcal{L}_{2}$, are given by

$$
\begin{aligned}
& \mathcal{L}_{0}=\mathcal{V}_{q}(n-\ell-1, d-1)+(q-2) \mathcal{V}_{q}(n-\ell-1, d-2), \\
& \mathcal{L}_{1}=(q-1) \sum_{i=\ell+2}^{n-\kappa-\ell+1}\left[\sum _ { j = 0 } ^ { i - \ell - 2 } \left(\binom{i-\ell-2}{j}(q-2)^{j}\right.\right. \\
& \left.\left.\quad \times \mathcal{V}_{q}(n-i-\ell+1, d-\ell-j-2)\right)\right]
\end{aligned}
$$

and

$$
\mathcal{L}_{2}=\sum_{i=0}^{n-\kappa-\ell}\binom{n-\kappa-\ell}{i}(q-2)^{i} \mathcal{V}_{q}(\kappa-1, d-i-2) .
$$

Proof. Assume that $X$ is a $\kappa$-WMU code over $\mathbb{F}_{q}^{n}$ generated according to Construction 4 and such that it has the cardinality at least $c_{q} \frac{q^{n}}{n-\kappa+1}$. Recall that in this case, $X$ is the set of sequences $\mathbf{a} \in \mathbb{F}_{q}^{n}$ that start with $\ell=$ $\left\lceil\log _{q} 2(n-\kappa+1)\right\rceil$ consecutive zeros $\left(\mathbf{a}_{1}^{\ell}=0^{\ell}\right), a_{\ell+1}, a_{n-\kappa+1} \neq 0$, and no $\ell$ consecutive zeros appear as a subsequence in $\mathbf{a}_{\ell+2}^{n-\kappa}$. With every $\mathbf{a} \in X$, we associate two sets $\mathcal{X}(\mathbf{a}, d-1)$ and $\mathcal{Y}(\mathbf{a}, d-1)$ : The set $\mathcal{X}(\mathbf{a}, d-1)$ includes sequences $\mathbf{b} \in \mathbb{F}_{q}^{n}$ that satisfy the following three conditions:

- Sequence $\mathbf{b}$ starts with $\ell$ consecutive zeros, i.e., $\mathbf{b}_{1}^{\ell}=0^{\ell}$.
- One has $b_{\ell+1} \neq 0$.
- Sequence $\mathbf{b}$ satisfies $d_{H}\left(\mathbf{a}_{\ell+1}^{n}, \mathbf{b}_{\ell+1}^{n}\right) \leq d-1$.

The set $\mathcal{Y}(\mathbf{a}, d-1) \subseteq \mathcal{X}(\mathbf{a}, d-1)$ is the collection of sequences $\mathbf{b}$ that contain $0^{\ell}$ as a subsequence in $\mathbf{b}_{\ell+2}^{n-\kappa}$, or that satisfy $b_{n-\kappa+1}=0$. Therefore,

$$
\mathcal{B}_{X}(\mathbf{a}, d-1)=\mathcal{X}(\mathbf{a}, d-1) / \mathcal{Y}(\mathbf{a}, d-1),
$$

and

$$
\left|\mathcal{B}_{X}(\mathbf{a}, d-1)\right|=|\mathcal{X}(\mathbf{a}, d-1)|-|\mathcal{Y}(\mathbf{a}, d-1)| .
$$

Let $\mathcal{L}_{0}=|\mathcal{X}(\mathbf{a}, d-1)|$. Thus,

$$
\begin{equation*}
\mathcal{L}_{0}=\mathcal{V}_{q}(n-\ell-1, d-1)+(q-2) \mathcal{V}_{q}(n-\ell-1, d-2) . \tag{2.6}
\end{equation*}
$$

The result holds as the first term on the right-hand side of the equation counts the number of sequences $\mathbf{b} \in \mathcal{X}(\mathbf{a}, d-1)$ that satisfy $b_{\ell+1}=a_{\ell+1}$, while the second term counts those sequences for which $b_{\ell+1} \neq a_{\ell+1}$.

We determine next $|\mathcal{Y}(\mathbf{a}, d-1)|$. For this purpose, we look into two disjoints subsets $\mathcal{Y}^{\mathrm{I}}$ and $\mathcal{Y}^{\mathrm{II}}$ in $\mathcal{Y}(\mathbf{a}, d-1)$ which allow us to use $|\mathcal{Y}(\mathbf{a}, d-1)| \geq$ $\left|\mathcal{Y}^{\mathrm{I}}\right|+\left|\mathcal{Y}^{\mathrm{II}}\right|$ and establish a lower bound on the cardinality sought.

The set $\mathcal{Y}^{\mathrm{I}}$ is defined according to

$$
\mathcal{Y}^{\mathrm{I}}=\bigcup_{i=\ell+2}^{n-\kappa-\ell+1} \mathcal{Y}^{\mathrm{I}}(i),
$$

where $\mathcal{Y}^{\mathrm{I}}(i)$ is the set of sequences $\mathbf{b} \in \mathcal{X}(\mathbf{a}, d-1)$ that satisfy the following constraints:

- The sequence $\mathbf{b}$ contains the substring $0^{\ell}$ starting at position $i$, i.e., $\mathbf{b}_{i}^{i+\ell-1}=0^{\ell}$.
- It holds that $b_{i-1} \neq 0$.
- The sequence $0^{\ell}$ does not appears as a substring in $\mathbf{b}_{\ell+2}^{i-2}$.
- One has $d_{\mathrm{H}}\left(\mathbf{a}_{\ell+1}^{i-1}, \mathbf{b}_{\ell+1}^{i-1}\right)+d_{\mathrm{H}}\left(\mathbf{a}_{i+\ell}^{n}, \mathbf{b}_{i+\ell}^{n}\right) \leq d-\ell-1$.

The cardinality of $\mathcal{Y}^{\mathrm{I}}(\ell+2)$ can be found according to

$$
\begin{aligned}
\left|\mathcal{Y}^{\mathrm{I}}(\ell+2)\right|= & \mathcal{V}_{q}(n-2 \ell-1, d-\ell-1) \\
& +(q-2) \mathcal{V}_{q}(n-2 \ell-1, d-\ell-2) \\
\geq & (q-1) \mathcal{V}_{q}(n-2 \ell-1, d-\ell-2)
\end{aligned}
$$

The first term on the right-hand side of the above equality counts the sequences $\mathbf{b} \in \mathcal{Y}^{\mathrm{I}}(\ell+2)$ for which $b_{\ell+1}=a_{\ell+1}$, while the second term counts sequences for which $b_{\ell+1} \neq a_{\ell+1}$. The inequality follows from the fact that $\mathcal{V}_{q}(n-2 \ell-1, d-\ell-1) \geq \mathcal{V}_{q}(n-2 \ell-1, d-\ell-2)$.

To evaluate the remaining terms $\mathcal{Y}^{\mathrm{I}}(i)$ for $\ell+3 \leq i \leq n-\kappa-\ell+1$, assume that $d_{\mathrm{H}}\left(\mathbf{a}_{\ell+1}^{i-2}, \mathbf{b}_{\ell+1}^{i-2}\right)=j$. In this case, there are at least

$$
\binom{i-\ell-2}{j}(q-2)^{j}
$$

possible choices for $\mathbf{b}_{\ell+1}^{i-2}$. This result easily follows from counting the number of ways to select the $j$ positions in $\mathbf{a}_{\ell+1}^{i-2}$ on which the sequences agree and the number of choices for the remaining symbols which do not include the corresponding values in a and 0 . As no additional symbol 0 is introduced in $\mathbf{b}_{\ell+1}^{i-2}, \mathbf{b}_{\ell+2}^{i-2}$ does not contain the substring $0^{\ell}$ as $\mathbf{a}_{\ell+2}^{i-2}$ avoids that string; similarly, $b_{\ell+1} \neq 0$.

On the other hand, there are $q-1$ possibilities for $b_{i-1} \in \mathbb{F}_{q} \backslash\{0\}$, and to satisfy the distance property we have to have

$$
d_{\mathrm{H}}\left(\mathbf{a}_{i+\ell}^{n}, \mathbf{b}_{i+\ell}^{n}\right) \leq d-\ell-j-2
$$

Therefore,

$$
\begin{aligned}
\left|\mathcal{Y}^{\mathrm{I}}(i)\right| \geq & (q-1)\left[\sum_{j=0}^{i-\ell-2}\binom{i-\ell-2}{j}(q-2)^{j}\right. \\
& \left.\times \mathcal{V}_{q}(n-i-\ell+1, d-\ell-j-2)\right]
\end{aligned}
$$

Hence, the cardinality of $\mathcal{Y}^{\mathrm{I}}$ may be bounded from below as

$$
\left|\mathcal{Y}^{\mathrm{I}}\right| \geq \mathcal{L}_{\mathrm{I}}
$$

where

$$
\begin{align*}
\mathcal{L}_{\mathrm{I}}= & (q-1) \sum_{i=\ell+2}^{n-\kappa-\ell+1}\left[\sum _ { j = 0 } ^ { i - \ell - 2 } \left(\binom{i-\ell-2}{j}(q-2)^{j}\right.\right. \\
& \left.\left.\times \mathcal{V}_{q}(n-i-\ell+1, d-\ell-j-2)\right)\right] . \tag{2.7}
\end{align*}
$$

The set $\mathcal{Y}^{\text {II }}$ comprises the set of sequences in $\mathbf{b} \in \mathcal{X}(\mathbf{a}, d-1)$ that have the following properties:

- The sequence $0^{\ell}$ does not appear as a substring in $\mathbf{b}_{\ell+2}^{n-\kappa}$.
- It holds $b_{n-\kappa+1}=0$.
- One has

$$
d_{\mathrm{H}}\left(\mathbf{a}_{\ell+1}^{n-\kappa}, \mathbf{b}_{\ell+1}^{n-\kappa}\right)+d_{\mathrm{H}}\left(\mathbf{a}_{n-\kappa+2}^{n}, \mathbf{b}_{n-\kappa+2}^{n}\right) \leq d-2
$$

It is easy to verify that

$$
\mathcal{Y}^{\mathrm{II}} \subseteq \mathcal{X}(\mathbf{a}, d-1) \backslash\left[\mathcal{B}_{X}(\mathbf{a}, d-1) \cup \mathcal{Y}_{\mathrm{I}}\right] .
$$

Following the same arguments used in establishing the bound on the cardi-
nality of $\mathcal{Y}^{\text {II }}$, one can show that

$$
\left|\mathcal{Y}^{\mathrm{II}}\right| \geq \mathcal{L}_{2},
$$

where

$$
\begin{equation*}
\mathcal{L}_{2}=\sum_{i=0}^{n-\kappa-\ell}\binom{n-\kappa-\ell}{i}(q-2)^{i} \mathcal{V}_{q}(\kappa-1, d-i-2) \tag{2.8}
\end{equation*}
$$

As a result, for each $\mathbf{a} \in X$, we have

$$
\begin{aligned}
\left|\mathcal{B}_{X}(\mathbf{a}, d-1)\right| & =|\mathcal{X}(\mathbf{a}, d-1)|-|\mathcal{Y}(\mathbf{a}, d-1)| \\
& \leq \mathcal{L}_{0}-\mathcal{L}_{1}-\mathcal{L}_{2}
\end{aligned}
$$

Note that $\mathcal{L}_{0}, \mathcal{L}_{1}, \mathcal{L}_{2}$ are independent from $\mathbf{a}$. Therefore,

$$
\mathcal{V}_{X, \text { max }}(d-1) \leq \mathcal{L}_{0}-\mathcal{L}_{1}-\mathcal{L}_{2}
$$

This inequality, along with the constrained form of the GV bound, establishes the validity of the claimed result.

Figure 2.1 plots the above derived lower bound on the maximum achievable rate for error-correcting $\kappa$-WMU codes (2.5), and for comparison, the best known error-correcting linear codes for binary alphabets. The parameters used are $n=50, \kappa=1, q=2$, corresponding to MU codes. To construct $q=$ 4 -ary error-correcting $\kappa$-WMU codes via the decoupled construction, we need to have at our disposition an error-correcting $\kappa$-WMU binary code. In what follows, we use ideas similar to Tavares' synchronization technique [36] to construct such codes. We start with a simple lemma and a short justification for its validity.

Lemma 3. Let $\mathcal{C}$ be a cyclic code of dimension $\kappa$. Then the run of zeros in any nonzero codesequence is at most $\kappa-1$.

Proof. Assume that there exists a non-zero codesequence $c(x)$, represented in polynomial form, with a run of zeroes of length $\kappa$. Since the code is cyclic, one may write $c(x)=a(x) g(x)$, where $a(x)$ is the information sequence corresponding to $c(x)$ and $g(x)$ is the generator polynomial of the code. Without loss of generality, one may assume that the run of zeros appears at positions


Figure 2.1: Comparison of two different lower bounds for binary codes: Error-correcting MU codes (inequality (2.5) of Theorem 4) and the best known linear error-correcting codes; $n=50, \kappa=1$, as $\kappa=1$-WMU codes are MU codes.
$0, \ldots, \kappa-1$, so that $\sum_{i+j=s} a_{i} g_{j}=0$, for $s \in\{0, \ldots, \kappa-1\}$. The solution of the previous system of equations gives $a_{0}=a_{1}=\ldots=a_{\kappa-1}=0$, contradicting the assumption that $c(x)$ is non-zero.

Construction 5. ( $\mathrm{d}-\mathrm{HD} \_\kappa-\mathrm{WMU}$ _q_n Codes) Construct a code $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ according to

$$
\mathcal{C}=\left\{\mathbf{a}+\mathbf{e} \mid \mathbf{a} \in \mathcal{C}_{1}, \mathbf{e}=(1,0, \ldots, 0)\right\}
$$

where $\mathcal{C}_{1}$ is a $[n, \kappa-1, d]$ cyclic code.
We argue that $\mathcal{C}$ is a $\kappa$-WMU code, with minimum Hamming distance $d$. To justify the result, we first demonstrate the property of weakly mutually uncorrelatedness. Suppose that on the contrary the code is $\mathcal{C}$ is not $\kappa$-WMU. Then there exists a proper prefix $\mathbf{p}$ of length at least $\kappa$ such that both pa and bp belong to $\mathcal{C}$. In other words, the sequences ( $\mathbf{p a}$ ) - $\mathbf{e}$ and (bp) - e belong to $\mathcal{C}_{1}$. Consequently, $(\mathbf{p b})-\mathbf{e}^{\prime}$ belongs to $\mathcal{C}_{1}$, where $\mathbf{e}^{\prime}$ is a cyclic shift of $\mathbf{e}$. Hence, by linearity of $\mathcal{C}_{1}, \mathbf{z} \triangleq \mathbf{0}(\mathbf{a}-\mathbf{b})+\mathbf{e}^{\prime}-\mathbf{e}$ belongs to $\mathcal{C}_{1}$. Now, observe that the first coordinate of $\mathbf{z}$ is one, and hence nonzero. But $\mathbf{z}$ has a run of zeros of length at least $\kappa-1$, which is a contradiction. Therefore, $\mathcal{C}$ is indeed a $\kappa$-WMU code. Since $\mathcal{C}$ is a coset of $\mathcal{C}_{1}$, the minimum Hamming distance property follows immediately.

As an example, consider the family of primitive binary $t$-error-correcting BCH codes with parameters $\left[n=2^{m}-1, \geq n-m t, \geq 2 t+1\right]$. The family is cyclic, and when used in Construction 5, it results in an error-correcting $(n-m t+1)$-WMU code of minimum distance $2 t+1$. The rate of such a code is $\frac{n-m t}{n} \geq 1-\frac{m t}{2^{m}-1}$, while according to the Theorem 3, the cardinality of the optimal-order corresponding $\kappa$-WMU code is at least $\frac{0.04688 \times 2^{n}}{m t}$, corresponding to an information rate of at least

$$
\begin{equation*}
\frac{\log \left(\frac{0.04688 \times 2^{n}}{m t}\right)}{n}>1-\frac{5+\log (m t)}{2^{m}-1} . \tag{2.9}
\end{equation*}
$$

As an illustration, we compare the rates of the BCH-based $\kappa$-WMU and the optimal $\kappa$-WMU codes for different values of $m=10, t=1,3,5$ :
(i) $m=10, t=1$ : In this case our BCH code has length 1023 , dimension 1013, and minimum Hamming distance 3. This choice of a code results in a binary 1014 -WMU code with minimum Hamming distance 3, and information rate 0.9902 , while the optimal binary 1014-WMU code has information rate greater than 0.9919.
(ii) $m=10, t=3$ : In this case our BCH code has length 1023 , dimension 993, and minimum Hamming distance 7. This choice of a code results in a binary 994 -WMU code with minimum Hamming distance 7, and information rate 0.9707 , while the optimal binary 994 -WMU code has information rate greater than 0.9903.
(iii) $m=10, t=5$ : In this case our BCH code has length 1023, dimension 973, and minimum Hamming distance 11. This choice of a code results in a binary 974 -WMU code with minimum Hamming distance 11, and information rate 0.9511 , while the optimal binary 974 -WMU code has information rate greater than 0.9896.

Next, we present a construction for MU ECC codes of length $n$, minimum Hamming distance $2 t+1$ and of size roughly $(t+1) \log n$. This construction outperforms the previous approach for codes of large rate, whenever $t$ is a small constant.

Assume that one is given a linear code of length $n^{\prime}$ and minimum Hamming distance $d_{H}=2 t+1$, equipped with a systematic encoder $\mathcal{E}_{H}\left(n^{\prime}, t\right):\{0,1\}^{\kappa} \rightarrow$ $\{0,1\}^{n^{\prime}-\kappa}$ which inputs $\kappa$ information bits and outputs $n^{\prime}-\kappa$ parity bits. We
feed into the encoder $\mathcal{E}_{H}$ sequences $\mathbf{u} \in\{\mathbf{0}, \mathbf{1}\}^{\kappa}$ that do not contain runs of zeros of length $\ell-1$ or more, where $\ell=\log \left(4 n^{\prime}\right)$. Let $p=\left\lceil\frac{\kappa}{n^{\prime}-\kappa}\right\rceil>\ell$. The MU ECC codesequences are of length $n=n^{\prime}+\ell+2$, and obtained according to the constrained information sequence $\mathbf{u}$ as:

$$
\begin{aligned}
& \left(0,0 \ldots, 0,1, \mathbf{u}_{\mathbf{1}}^{\mathbf{p}}, \mathcal{E}_{H}(\mathbf{u})_{\mathbf{1}}^{1}, \mathbf{u}_{\mathbf{p}+\mathbf{1}}^{2 \mathbf{p}}, \mathcal{E}_{\mathbf{H}}(\mathbf{u})_{\mathbf{2}}^{2}\right. \\
& \left.\quad \mathbf{u}_{\mathbf{2 p}+\mathbf{1}}^{3 \mathrm{p}}, \mathcal{E}_{H}(\mathbf{u})_{3}^{3} \ldots, \mathbf{u}_{\left(\mathbf{n}^{\prime}-\kappa-1\right) \mathbf{p}+\mathbf{1}}^{\mathbf{n}^{\prime}}, \mathcal{E}_{\mathbf{H}}(\mathbf{u})_{\mathbf{n}^{\prime}-\kappa}^{\mathbf{n}^{\prime}-\kappa}, \mathbf{1}\right)
\end{aligned}
$$

The codesequences start with the $0^{\ell} 1$ substring, and are followed by sequences $\mathbf{u}$ interleaved with parity bits, which are inserted every $p>\ell$ positions. Notice that the effect of the inserted bits is that they can extend the lengths of existing runs of zeros in $\mathbf{u}$ by at most one. Since $\mathbf{u}$ has no runs of length $\ell-1$ or more, this means that we do not see any runs of zeros of length $\geq \ell$ in the last $n-\ell-1$ bits of $\mathbf{x}$. This implies that the underlying code is MU, while the ECC properties are inherited from the initial linear code.

### 2.5 Balanced $\kappa$-WMU Codes

In what follows, we focus on the analysis of balanced $\kappa$-WMU codes, and start with a review of known bounds on the number of balanced binary sequences.

Let $A_{\text {d-HD_2_n }}$ denote the maximum size of a binary code of length $n$ and minimum Hamming distance $d$, and let $A_{\text {w-CST_d-HD_2_n }}$ denote the maximum cardinality of a binary code with constant weight $w$, length $n$ and even minimum Hamming distance $d_{H}$. Clearly,

$$
A_{\frac{\mathrm{n}}{2}-\mathrm{CST}_{-} 2-\mathrm{HD} \__{-}^{2}-\mathrm{n}}=\binom{n}{\frac{n}{2}} .
$$

Gyorfi et al. [37] derived several bounds for the more general function $A_{\text {w-CST_d-HD_2_n }}$ based on $A_{d-\mathrm{HD} \_2 \_\mathrm{n}}$.

Theorem 5. For even integer $d, 0 \leq d \leq n$, and every $w, 0 \leq w \leq n$,

$$
\frac{\binom{n}{w}}{2^{n-1}} A_{\mathrm{d}-\mathrm{HD} \_\__{-} \mathrm{n}} \leq A_{\mathrm{w}-\text { CST }_{-} \mathrm{d}-\mathrm{HD} \_^{2} \_\mathrm{n}} .
$$

We present next our first construction of balanced $\kappa$-WMU codes.

Construction 6. (BAL_ $\kappa-W M U \_4 \_n$ Codes) Form a code $\mathcal{C} \in\{A, T, C, G\}^{n}$ using the decoupled construction with component codes $\mathcal{C}_{1}$ and $\mathcal{C}_{2}$ chosen according to the following rules:

- Let $\mathcal{C}_{1} \subseteq\{0,1\}^{n}$ be a balanced code of size equal to $A_{\frac{\mathrm{n}}{2}-\text { CST_2-HD_2_n }}$.
- Let $\mathcal{C}_{2} \subseteq\{0,1\}^{n}$ be a $\kappa$-WMU code; one may use Construction 4 to generate $\mathcal{C}_{2}$.

Lemma 4. Let $\mathcal{C} \in\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}^{n}$ denote the code generated by Construction 6. Then,
(i) $\mathcal{C}$ is a $\kappa$-WMU code.
(ii) $\mathcal{C}$ is balanced.

Proof. (i) Since $\mathcal{C}_{2}$ is a $\kappa$-WMU code, property ii) of Lemma 2 ensures that $\mathcal{C}$ is also a $\kappa$-WMU code.
(ii) Since $\mathcal{C}_{1}$ is balanced, property i) of Lemma 2 ensures that $\mathcal{C}$ is a balanced binary code.
This completes the proof.
We discuss next the cardinality of the code $\mathcal{C}$ generated by Construction 6. According to Theorem 3, one has $\left|\mathcal{C}_{2}\right|=c_{2} \frac{2^{n}}{n-\kappa+1}$. In addition, $\left|\mathcal{C}_{1}\right|=\binom{n}{\frac{n}{2}}$. Hence, the size of $\mathcal{C}$ is bounded from below by:

$$
c_{2} \frac{\binom{n}{\frac{n}{2}} 2^{n}}{n-\kappa+1} .
$$

Theorem 6 proves that both Construction 1 and 6 are order optimal, in the sense that they produce codes with cardinality within a constant factor away from the maximal achievable value.

Theorem 6. Let $A_{\text {BAL_к-wMU_q_n }}$ denote the maximum size of a balanced $\kappa$-WMU code over $\mathbb{F}_{q}^{n}$, for $n \geq 2$ and $q \in\{2,4\}$. Then,

$$
\begin{equation*}
c_{2} \frac{\binom{n}{\frac{n}{2}} 2^{n}}{n-\kappa+1} \leq A_{\text {BAL_}_{-} \kappa-\text { WMU_4_n }} \leq \frac{\binom{n}{\frac{n}{2}} 2^{n}}{n-\kappa+1} . \tag{i}
\end{equation*}
$$

(ii)

$$
A_{\text {BAL_ }^{\kappa-\text { WMU_2_n }}} \leq \frac{\binom{n}{\frac{n}{2}}}{n-\kappa+1} .
$$

(iii)

$$
\frac{\binom{n}{\frac{n}{2}}}{2(n-1)} \leq A_{\text {BAL_MU_2_n }^{\mathrm{n}}} \leq \frac{\binom{n}{\frac{n}{2}}}{n} .
$$

Proof. To prove the upper bounds, we use the same technique as that described in Theorem 3. Assume that $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ is a balanced $\kappa$-WMU code, for $q \in\{2,4\}$, and consider the set $X$ of pairs (a,i) where $\mathbf{a} \in \mathbb{F}_{q}^{n}, i \in\{1, \ldots, n\}$, and the cyclic shift of the sequence a starting at position $i$ belongs to $\mathcal{C}$. One may easily verify that $|X|=n|\mathcal{C}|$. On the other hand, if $(\mathbf{a}, i) \in X$, then $\mathbf{a}$ is balanced itself and there are $\binom{n}{\frac{n}{2}}\left(\frac{q}{2}\right)^{n}$ balanced sequences to select from. Moreover, $(\mathbf{a}, j) \notin X$, for $j \notin\{i \pm 1, \ldots, i \pm(n-\kappa)\}_{\bmod n}$ due to the $\kappa$-WMU property. Hence, for a fixed balanced sequence $\mathbf{a} \in \mathbb{F}_{q}^{n}$, there are at most $\left\lfloor\frac{n}{n-\kappa+1}\right\rfloor$ pairs $\left(\mathbf{a}, i_{1}\right), \ldots,\left(\mathbf{a}, i_{\left\lfloor\frac{n}{n-\kappa+1}\right\rfloor}\right) \in X$. This implies that

$$
|X| \leq \frac{n\binom{n}{\frac{n}{2}}\left(\frac{q}{2}\right)^{n}}{n-\kappa+1}
$$

Therefore, $|\mathcal{C}| \leq \frac{\binom{n}{\frac{n}{2}}\left(\frac{q}{2}\right)^{n}}{n-\kappa+1}$.
The lower bound in (i) can be achieved through Construction 6, while the lower bound in (iii) can be met using Construction 1.

We complete our discussion by briefly pointing out how to use the balanced MU code Construction 1 to derive a balanced $\kappa$-WMU code $\mathcal{C} \in\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}^{n}$ that has the prefix balancing property with parameter $D$. For this purpose, we generate $\mathcal{C}$ according to the balanced WMU Construction 6. We set $\mathcal{C}_{2}=\{0,1\}^{n}$ and construct $\mathcal{C}_{1}$ by concatenating $\mathcal{C}_{1}^{\prime} \subseteq\{0,1\}^{\kappa-1}$ and $\mathcal{C}_{1}^{\prime \prime} \subseteq$ $\{0,1\}^{n-\kappa+1}$. Here, $\mathcal{C}_{1}^{\prime}$ is balanced and $\mathcal{C}_{1}^{\prime \prime}$ is a balanced WMU code with parameter $D$. It is easy to verify that $\mathcal{C}$ is a balanced $\kappa$-WMU DNA code with prefix-balancing parameter $D$ and of cardinality

$$
\begin{aligned}
|\mathcal{C}| & =\left|\mathcal{C}_{1}^{\prime}\right|\left|\mathcal{C}_{1}^{\prime \prime}\right|\left|\mathcal{C}_{2}\right|=A\left(\kappa-1,2, \frac{\kappa-1}{2}\right) \operatorname{Dyck}\left(\frac{n-\kappa}{2}, D\right) 2^{n} \\
& \sim \frac{4^{n} \tan ^{2}\left(\frac{\pi}{D+1}\right) \cos ^{n-\kappa}\left(\frac{\pi}{D+1}\right)}{\sqrt{2 \pi}(D+1)(\kappa-1)^{\frac{1}{2}}} .
\end{aligned}
$$

### 2.6 APD-MU Codes

Our next goal is to provide constructions for $\kappa$-WMU codes that do not form primer dimer byproducts.

We first discuss a construction of binary MU codes with the APD property.
Construction 7. (f - APD_MU_2_n) Let $n, f, \ell, p$ be positive integers such that $n=p f$ and $\ell+3 \leq \frac{f}{2}$. Let

$$
\mathcal{C}=\left\{\mathbf{a}_{1} \mathbf{a}_{2} \ldots \mathbf{a}_{2 p} \mid \mathbf{a} \in \mathcal{C}_{1}, \mathbf{a}_{2}, \ldots, \mathbf{a}_{2 p} \in \mathcal{C}_{2}\right\}
$$

where $\mathcal{C}_{1} \subseteq \mathbb{F}_{2}^{\frac{f}{2}}$ is the set of binary sequences $\mathbf{a}=\left(a_{1}, \ldots, a_{\frac{f}{2}}\right)$ such that:

- The sequence a starts with $0^{\ell} 1$ and ends with 1 ;
- The substring $\mathbf{a}_{\ell+1}^{\frac{f}{2}}$ does not contain $0^{\ell}$ as a substring;
and where $\mathcal{C}_{2} \subseteq \mathbb{F}_{2}^{\frac{f}{2}}$ is the set of binary sequences $\mathbf{a}=\left(a_{1}, \ldots, a_{\frac{f}{2}}\right)$ such that:
- The sequence a ends with 1 ;
- The sequence a contains $01^{\ell} 0$ as a substring.
- The sequence a does not contain $0^{\ell}$ as a substring.

Lemma 5. Let $\mathcal{C} \in\{0,1\}^{n}$ denote the code generated by Construction 7 . Then,
(i) $\mathcal{C}$ is an MU code.
(ii) $\mathcal{C}$ is an $f$-APD code.

Proof. The proof follows from two observations, namely
(i) The code $\mathcal{C}$ satisfies the constraints described in Construction 2, and is hence an MU code.
(ii) Any substring of length $f$ of any sequence in $\mathcal{C}$ contains an element from $\mathcal{C}_{2}$ as a substring. Hence, any substring of length $f$ in $\mathcal{C}$ contains $01^{\ell} 0$ as a substring, and so the reverse and forward complement sequence contains $10^{\ell} 1$. Furthermore, no proper substring of length $f$ in $\mathcal{C}$ contains $10^{\ell} 1$ as a substring. Hence, $\mathcal{C}$ is also an $f$-APD code.

Next, we use Lemma 1 to derive a lower bound on the size of the codes $\mathcal{C}_{1}$ and $\mathcal{C}_{2}$ in Construction 7, and a lower bound on the size of the code $\mathcal{C}$. First, notice that

$$
\left|\mathcal{C}_{1}\right| \geq \frac{2^{\frac{f}{2}}}{2^{\ell+2}}\left(1-\frac{\frac{f}{2}-\ell-2}{2^{\ell}}\right) \geq \frac{2^{\frac{f}{2}}}{2^{\ell+2}}\left(1-\frac{f}{2^{\ell+1}}\right)
$$

which follows from Lemma 1 , with $n=\frac{f}{2}-\ell-2, n_{s}=\ell, t=1$. To bound the cardinality of $\mathcal{C}_{2}$ we define an auxiliary code $\mathcal{C}_{3} \subseteq\{0,1\}^{\frac{f}{2}-\ell-3}$ such that sequences in $\mathcal{C}_{3}$ avoid $0^{\ell-1}, 1^{\ell-1}$ as a substring. One can once more apply Lemma 1 with $n=\frac{f}{2}-\ell-3, n_{s}=\ell-1, t=2$, to obtain

$$
\left|\mathcal{C}_{3}\right| \geq \frac{2^{\frac{f}{2}}}{2^{\ell+3}}\left(1-\frac{4\left(\frac{f}{2}-\ell-3\right)}{2^{\ell}}\right) \geq \frac{2^{\frac{f}{2}}}{2^{\ell+3}}\left(1-\frac{f}{2^{\ell-1}}\right)
$$

Notice that by inserting $01^{\ell} 0$ into sequences in $\mathcal{C}_{3}$ at any of the $\frac{f}{2}-\ell-2$ allowed positions, and then appending 1 to the newly obtained sequence, we obtain a subset of $\mathcal{C}_{2}$ of size $\left(\frac{f}{2}-\ell-2\right)\left|\mathcal{C}_{3}\right|$. Therefore,

$$
\left|\mathcal{C}_{2}\right| \geq\left(\frac{f}{2}-\ell-2\right) \frac{2^{\frac{f}{2}}}{2^{\ell+3}}\left(1-\frac{f}{2^{\ell-1}}\right)
$$

For $\ell=\left\lceil\log _{2}(3 f)\right\rceil$, one can verify that the size of the code $\mathcal{C}_{1}$ is within a constant factor of $\frac{2^{\frac{f}{T}}}{f}$, and the size of $\mathcal{C}_{2}$ is within a constant factor of $2^{\frac{f}{2}}$. In the last step, we use the fact that $|\mathcal{C}|=\left|\mathcal{C}_{1}\right|\left|\mathcal{C}_{2}\right|^{2 p-1}$ to show that $|\mathcal{C}|$ is within a constant factor of $\frac{2^{n}}{n}$. Therefore, Construction 7 produces an order-optimal $f$-APD MU binary codes. The result is summarized in the following theorem.

Theorem 7. Let $A_{\text {f-APD_MU_2_n }}$ denote the maximum size of an f-APD_MU_2_n code, for positive integers $n=p f$ such that $p$ is a constant factor. Then, there exist constants $c_{3}>0$ such that

$$
c_{3} \frac{2^{n}}{n} \leq A_{\text {f-APD_MU_2_n }} \leq \frac{2^{n}}{n} .
$$

Proof. The lower bound is a direct consequence of Construction 7, while the upper bound follows from Theorem 1, and the fact that any $f-$ APD_MU_2_n code is also an MU_2_n code.

### 2.7 APD, Balanced, Error-Correcting and WMU Codes

In what follows, we describe the main results of our work: Constructions of APD, balanced, error-correcting $\kappa$-WMU codes. The gist of our approach is to use the decoupling principle along with a pair of binary codes that satisfy one or two of the desired binary primer constraints in order to obtain a large set of proper address/primer sequences. In addition to constructions based on the decoupling procedure, we introduce a number of other constructions that directly produce the desired $q$-ary codes with large codebooks, or allow for simple encoding and decoding.

Recall Construction 5, in which we showed results in an error-correcting $\kappa$-WMU DNA code. Map the elements in $\mathbb{F}_{4}$ to $\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$ according to:

$$
0 \mapsto \mathrm{~A}, 1 \mapsto \mathrm{C}, \omega \mapsto \mathrm{~T}, \omega+1 \mapsto \mathrm{G},
$$

where $\omega$ is a primitive element of the field $\mathbb{F}_{4}$.
Let a be a sequence of length $n$. Then it is straightforward to see that the sequence $\left(\mathbf{a}, \mathbf{a}+1^{n}\right)$ is balanced, for $\mathbf{1}^{n}=(1, \ldots, 1)$. These observations lead to the simple primer construction described next.

Construction 8. (V1: BAL_2d $-\mathrm{HD} \_\kappa-\mathrm{WMU} \_4 \_$n Codes) Let $\mathcal{C}$ be an $\left[\frac{n}{2}, \kappa-1, d\right]$ cyclic code over $\mathbb{F}_{4}$ that contains the all ones vector 1 . Then

$$
\left\{\left(\mathbf{c}+\mathbf{e}, \mathbf{c}+\mathbf{1}^{\frac{n}{2}}+\mathbf{e}\right): \mathbf{c} \in \mathcal{C}\right\}
$$

is a GC balanced, $\kappa$-WMU code with minimum Hamming distance $2 d$.
The next construction follows by invoking the decoupling principle with binary error-correcting WMU codes constructed in Section 2.4 and codes meeting the bound of Theorem 5 .

Construction 9. (V2 : BAL_d $-H D \_\kappa-W M U \_4 \_n$ Codes) Construct a code $\mathcal{C} \in\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}^{n}$ via the decoupled construction of Lemma 2 involving two codes:
(i) A balanced code $\mathcal{C}_{1}$ of length $n$, with minimum Hamming distance $d$ and of size $A_{\frac{\mathrm{n}}{2}-\mathrm{CST} \mathrm{D}^{\mathrm{d}-\mathrm{HD}} \mathrm{Z}^{2} \mathrm{n}^{\mathrm{n}}}$.


Figure 2.2: Comparison of three different lower bounds for quaternary codes: Balanced error-correcting $\kappa$-WMU codes (inequality (2.10) in Example 1), error-correcting $\kappa$-WMU codes (inequality (2.5) in Theorem 4) and the best known linear error-correcting codes; $n=50, \kappa=25$.
(ii) A $\kappa$-WMU code $\mathcal{C}_{2} \subseteq\{0,1\}^{n}$ of length $n$ and minimum Hamming distance $d$, described in Section 2.4.

Lemma 6. Let $\mathcal{C} \in\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}^{n}$ denote the code generated by Construction 9 . Then,
(i) $\mathcal{C}$ is a $\kappa$-WMU code.
(ii) $\mathcal{C}$ is balanced.
(iii) The minimum Hamming distance of $\mathcal{C}$ is at least $d$.

Example 1. The size of the code $\mathcal{C}$ obtained from Construction 9 equals

$$
\begin{align*}
|\mathcal{C}| & =\left|\mathcal{C}_{1}\right|\left|\mathcal{C}_{2}\right| \\
& \geq c_{2} \frac{2^{n} A_{\frac{\mathrm{n}}{2}-\mathrm{CST}_{-} \mathrm{d}-\mathrm{HD} 2^{2} \_^{\mathrm{n}}}}{(n-\kappa+1)\left(\mathcal{L}_{0}-\mathcal{L}_{1}-\mathcal{L}_{2}\right)} \\
& \geq 0.09376 \frac{\binom{n}{\frac{n}{2}} A_{\mathrm{d}-\mathrm{HD} \_^{2} \_^{\mathrm{n}}}}{(n-\kappa+1)\left(\mathcal{L}_{0}-\mathcal{L}_{1}-\mathcal{L}_{2}\right)} . \tag{2.10}
\end{align*}
$$

The last two inequalities follow from the lower bounds of Lemma 4 and Theorem 5, respectively.

Figure 2.2 plots the lower bound on the maximum achievable rate for errorcorrecting $\kappa$-WMU codes (2.5), balanced error-correcting $\kappa$-WMU codes (2.10), and for comparison, the best known linear error-correcting codes over quaternary alphabets. The parameters used are $n=50, \kappa=25$.

The next result shows that Construction 7 may be used to devise sequences that are balanced, MU and do not form any PDs.

Construction 10. (f - APD_BAL_MU_4_n Codes) Using the decoupled code construction in Lemma 2, a balanced, MU code $\mathcal{C} \in \mathbb{F}_{4}^{n}$ that avoids PDs may be obtained by choosing $\mathcal{C}_{1} \subseteq \mathbb{F}_{2}^{n}$ to be an BAL_2_n code, and $\mathcal{C}_{2} \subseteq \mathbb{F}_{2}^{n}$ to be an $f-$ APD_MU_2_n code.

It is straightforward to see that $\left|\mathcal{C}_{1}\right|=\binom{n}{\frac{n}{2}}$ and that $\left|\mathcal{C}_{2}\right| \geq c_{3} \frac{2^{n}}{n}$. Therefore, the size of $\mathcal{C}$ is at least $c_{3} \frac{\left(\begin{array}{c}n \\ \frac{n}{2}\end{array} 2^{n}\right.}{n}$.

Theorem 8. Let $A_{\text {f-APD_BAL_MU_4_n }}$ denote the maximum size of a f - APD_BAL_MU_4_n code. Then

$$
c_{3} \frac{\binom{n}{\frac{n}{2}} 2^{n}}{n} \leq A_{\text {f-APD_BAL_MU_4_n}} \leq \frac{\binom{n}{\frac{n}{2}} 2^{n}}{n} .
$$

Proof. The lower bound is the direct consequence of Construction 10. To prove the upper bound, observe that any $f-\operatorname{APD}$ _BAL_MU_4_n code is also a valid BAL_MU_4_n code. The upper bound on the cardinality of an BAL_MU_4_n code may be obtained from the upper bound of Theorem 6, pertaining to a BAL_WMU_4_n code, by setting $\kappa=1$.

Next, we discuss an iterative construction based on an APD, balanced, error-correcting and $\kappa$-WMU seed code.

Construction 11. For a given integer $s \geq 1$, let $\mathcal{C}_{0}$ be a set of sequences in $\mathbb{F}_{q}^{s}$. Let

$$
\mathcal{C}=\left\{\mathbf{a}_{1} \ldots \mathbf{a}_{m} \mid \mathbf{a}_{i} \in \mathcal{C}_{i} \text { for } 1 \leq i \leq m\right\}
$$

where the subset codes $\mathcal{C}_{1}, \ldots, \mathcal{C}_{m} \subseteq \mathcal{C}_{0}$ are chosen according to:

$$
\begin{aligned}
& \quad \mathcal{C}_{1} \cap \mathcal{C}_{m}=\emptyset \\
& \text { and }\left(\mathcal{C}_{1} \cap \mathcal{C}_{m-1}=\emptyset\right) \text { or }\left(\mathcal{C}_{2} \cap \mathcal{C}_{m}=\emptyset\right) \\
& \quad \vdots \\
& \text { and }\left(\mathcal{C}_{1} \cap \mathcal{C}_{2}=\emptyset\right) \text { or } \ldots \text { or }\left(\mathcal{C}_{m-1} \cap \mathcal{C}_{m}=\emptyset\right) .
\end{aligned}
$$

Lemma 7. Let $\mathcal{C} \subseteq \mathbb{F}_{q}^{n}$ be a code generated according to the Construction 11. Then
(i) $\mathcal{C}$ is $2 f$ - APD if $\mathcal{C}_{0}$ is $f$ - APD .
(ii) $\mathcal{C}$ is balanced if $\mathcal{C}_{0}$ is balanced.
(iii) $\mathcal{C}$ and $\mathcal{C}_{0}$ have the same minimum Hamming distance.
(iv) $\mathcal{C}$ is $\kappa$-WMU if $\mathcal{C}_{0}$ is $\kappa$-WMU.

Proof. (i) Any proper substring of length $2 f$ of any codesequence in $\mathcal{C}$ contains a proper substring of length $f$ of a codesequence in $\mathcal{C}_{0}$. Then, $\mathcal{C}$ is $2 f$ - APD if $\mathcal{C}_{0}$ is $f$ - APD .
(ii) Codesequences in $\mathcal{C}$ form by concatenating codesequences in $\mathcal{C}_{0}$. If $\mathcal{C}_{0}$ is balanced then each codesequences in $\mathcal{C}$ is also balanced.
(iii) Again, any two distinct codesequences in $\mathcal{C}$ differ in at least one of the concatenated codesequences from $\mathcal{C}_{0}$. Therefore, $\mathcal{C}$ and $\mathcal{C}_{0}$ have identical minimum Hamming distance.
(iv) For any pair of not necessarily distinct $\mathbf{a}, \mathbf{b} \in \mathcal{C}$ and for $\kappa \leq l<n$, we show that $\mathbf{a}_{1}^{l}$ and $\mathbf{b}_{n-l+1}^{n}$ cannot be identical. This establishes that the constructed concatenated code is WMU. Let $l=i s+j$, where $i=\left\lfloor\frac{l}{s}\right\rfloor$ and $0 \leq j<s$. We consider three different scenarios for the index $j$ :

- $j=0$; In this case, $1 \leq i<m$. Therefore, $\left(\mathcal{C}_{1} \cap \mathcal{C}_{m-i+1}=\right.$ $\emptyset)$ or $\ldots$ or $\left(\mathcal{C}_{i} \cap \mathcal{C}_{1}=\emptyset\right)$ implies that $\mathbf{a}_{1}^{l} \neq \mathbf{b}_{n-l+1}^{n}$.
- $0<j<\kappa$; Again, one can verify that $1 \leq i<m$. It is easy to show that $\mathbf{a}_{l-s+1}^{l-j}$ is a suffix of length $s-j$ of a sequence in $\mathcal{C}_{0}$ and $\mathbf{b}_{n-s+1}^{n-j}$ is a prefix of length $s-j$ of an element in $\mathcal{C}_{0}$. Since $\kappa<s-j<s$, one has $\mathbf{a}_{l-s+1}^{l-j} \neq \mathbf{b}_{n-s+1}^{n-j}$. Hence, $\mathbf{a}_{1}^{l} \neq \mathbf{b}_{n-l+1}^{n}$.
- $\kappa \leq j<s$; In this case, $\mathbf{a}_{l-j+1}^{l}$ is a proper prefix of length $j$ of a sequence in $\mathcal{C}_{0}$, and $\mathbf{b}_{n-j+1}^{n}$ is a proper suffix of length $j$ of an element in $\mathcal{C}_{0}$. Since $\kappa \leq j<s$, one has $\mathbf{a}_{l-j+1}^{l} \neq \mathbf{b}_{n-j+1}^{n}$ and $\mathbf{a}_{1}^{l} \neq \mathbf{b}_{n-l+1}^{n}$.


Figure 2.3: Concatenation construction for information blocks avoiding $\kappa$-WMU primer sequences. The gist of the approach is to use a subset of address sequences for actual addressing, and the remaining sequences as blocks to be concatenated.

### 2.8 Information Encoding with WMU Addresses

In order to store user data in DNA, one needs to encode the binary information into relatively short sequences of nucleotide, each of which is equipped with a unique address sequence that satisfies the constraints outlined in the previous section. As already described, in order to enable accurate random access via PCR, the information bearing content of the sequences has to avoid the set of address sequences. This leads to the following problem formulation.

Given a set of sequences $\mathcal{A} \subseteq \mathbb{F}_{q}^{n}$, let $\mathcal{C}_{\mathcal{A}}(N) \subseteq \mathbb{F}_{q}^{N}$ denote another collection of sequences that avoid members of $\mathcal{A}$ as substrings, i.e., $\mathbf{a} \neq \mathbf{b}_{i}^{n+i-1}$ for every $\mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{C}_{\mathcal{A}}(N), 1 \leq i \leq N-n+1$. We refer to $\mathcal{A}$ as the set of addresses and $C_{\mathcal{A}}(N)$ as the set of address-avoiding information blocks.

We discuss next three different schemes for constructing an information codebook $\mathcal{C}_{\mathcal{A}}(N)$ of sequences of length $N$ for particular sets of address sequences $\mathcal{A}$.

Let $\mathcal{C}$ be an $\kappa$-WMU code over $\mathbb{F}_{q}^{n}$, for $\kappa \leq \frac{n}{2}$, and let $\mathcal{A} \subset \mathcal{C}$. For a given integer $s \geq 1$, let $N=s n$ and define $\mathcal{C}_{\mathcal{A}}(N) \subseteq \mathbb{F}_{q}^{N}$ as the collection of all sequences $\mathbf{b} \in \mathbb{F}_{q}^{N}$ of the form

$$
\mathbf{b}=\mathbf{b}_{1} \ldots \mathbf{b}_{s}
$$

where $\mathbf{b}_{i} \in \mathcal{C}-\mathcal{A}$, for $1 \leq i \leq s$. This construction is illustrated in Figure 2.3. We show next that no $\mathbf{b} \in \mathcal{C}_{\mathcal{A}}(N)$ contains any $\mathbf{a} \in \mathcal{A}$ as a substring.

The proof follows by contradiction. Assume that a appears as substring in $\mathbf{b}_{i} \mathbf{b}_{i+1}$, for some $1 \leq i<s$. Since $\mathbf{a} \in \mathcal{A}$ and $\mathbf{b}_{i}, \mathbf{b}_{i+1} \in \mathcal{C}-\mathcal{A}$, a may be written as $\mathbf{a}=\mathbf{s p}$, where $\mathbf{s}$ is a proper suffix of $\mathbf{b}_{i}$ and $\mathbf{p}$ is a proper prefix of $\mathbf{b}_{i+1}$. Then one of the two strings $\mathbf{p}$ or $\mathbf{s}$ has length greater than or equal to $\frac{n}{2} \geq \kappa$, which contradicts the fact that $\mathcal{C}$ is an $\kappa$-WMU code.

The previously described construction may be easily extended to obtain error-correcting information blocks $\mathcal{C}_{\mathcal{A}}(N)$. We start by identifying a bijection $\mathcal{M}$ from $\mathcal{C}-\mathcal{A}$ to a finite field $\mathbb{F}_{p^{t}}$ with appropriate prime parameter $p$ and $t \geq 1$; for this purpose, we expurgate codesequences from the set $\mathcal{C}$ so that $|\mathcal{C}|-|\mathcal{A}|=p^{t}$.

The bijection $\mathcal{M}$ is used to convert every sequence $\mathbf{b}=\mathbf{b}_{1} \ldots \mathbf{b}_{s}$ in $\mathcal{C}_{\mathcal{A}}(N)$ to a sequence $\mathbf{v}=\mathbf{v}_{1} \ldots \mathbf{v}_{s} \in \mathbb{F}_{p^{t}}^{s}$, where $\mathbf{v}_{i}=\mathcal{M}\left(\mathbf{b}_{i}\right)$, for $1 \leq i \leq s$. The sequence $\mathbf{v}$ is encoded using an $[r, s, r-s+1]_{p^{t}}$ Reed-Solomon (RS) error correcting code to arrive at a codesequence $\mathbf{w}=\mathbf{w}_{1} \ldots \mathbf{w}_{r} \in \mathbb{F}_{p^{t}}^{r}$, where $\mathbf{w}_{i} \in \mathbb{F}_{p^{t}}$ for $1 \leq i \leq r$. Since $\mathcal{M}$ is a bijection, one can apply $\mathcal{M}^{-1}$ to $\mathbf{w}$ to reconstruct $\mathbf{c}=\mathbf{c}_{1} \ldots \mathbf{c}_{r} \in \mathbb{F}_{q}^{s r}$, where $\mathbf{c}_{i}=\mathcal{M}^{-1}\left(\mathbf{w}_{i}\right)$, for $1 \leq i \leq r$. Since $\mathbf{c}$ is obtained by concatenating elements from $\mathcal{C}-\mathcal{A}$, it is easy to verify that c does not contain any element of $\mathcal{A}$ as a substring. Moreover, the RS code guarantees that given a sequence $\mathbf{c}$ with at most $\left\lfloor\frac{r-s}{2}\right\rfloor$ errors, one can still fully recover b.

For the second scheme, assume that $\mathcal{C}_{1}, \mathcal{C}_{2} \subseteq \mathbb{F}_{2}^{n}$ are two disjoint collections of binary sequences of length $n$ such that for all $\mathbf{a} \in \mathcal{C}_{1}$, the cyclic shifts of $\mathbf{a}$ do not belong to $\mathcal{C}_{2}$, i.e., for all $\mathbf{a} \in \mathcal{C}_{1}, \mathbf{a}_{i}^{n} \mathbf{a}_{1}^{i-1} \notin \mathcal{C}_{2}$ for all $1 \leq i \leq n$.

Now given $\mathcal{C}_{1}$ and $\mathcal{C}_{2}$, define the set of addresses $\mathcal{A} \subseteq \mathbb{F}_{4}^{2 n}$ as

$$
\mathcal{A}=\left\{\Psi(\mathbf{c}, \mathbf{a a}) \mid \mathbf{a} \in \mathcal{C}_{1}, \mathbf{c} \in \mathbb{F}_{2}^{2 n}\right\},
$$

where $\Psi$ was introduced in (2.4). To construct $\mathcal{C}_{\mathcal{A}}(N)$, let $s \geq 1$ be an integer such that $N=s n$. We define $\mathcal{C}_{\mathcal{A}}(N) \subseteq \mathbb{F}_{4}^{N}$ as the collection of all sequences $\mathbf{b}=\mathbf{b}_{1} \ldots \mathbf{b}_{s} \in \mathbb{F}_{4}^{N}$ where $\mathbf{b}_{i} \in \mathbb{F}_{4}^{n}$ that can be written as $\mathbf{b}_{i}=\Psi\left(\mathbf{f}_{i}, \mathbf{g}_{i}\right)$, for some $\mathbf{g}_{i} \in \mathcal{C}_{2}, \mathbf{f}_{i} \in \mathbb{F}_{2}^{n}$ and $1 \leq i \leq s$. We claim that $\mathcal{C}_{\mathcal{A}}(N)$ does not contain any element of $\mathcal{A}$ as a substring.

If $\Psi(\mathbf{c}, \mathbf{a a}) \in \mathcal{A}$ appears as a substring in a sequence $\mathbf{b} \in \mathcal{C}_{\mathcal{A}}(N)$, then there exists an index $1 \leq i \leq s-2$ such that $\Psi(\mathbf{c}, \mathbf{a a})$ is a substring of $\mathbf{b}_{i} \mathbf{b}_{i+2} \mathbf{b}_{i+3}$. Since $\Psi$ is a bijection one can verify that aa appears as a substring in $\mathbf{g}_{i} \mathbf{g}_{i+1} \mathbf{g}_{i+2}$, for $\mathbf{a} \in \mathcal{C}_{1}$ and $\mathbf{g}_{i}, \mathbf{g}_{i+1}, \mathbf{g}_{i+2} \in \mathcal{C}_{2}$. In addition, $\mathcal{C}_{1} \cap \mathcal{C}_{2}=\varnothing$
implies that aa can be written as $\mathbf{a a}=\mathbf{~}_{\mathbf{g}_{i+1}} \mathbf{p}$, where $\mathbf{s}$ is a proper suffix of $\mathbf{g}_{i}$ and $\mathbf{p}$ is a proper prefix of $\mathbf{g}_{i+2}$. It is clear that $\mathbf{a}=\mathbf{s p}$ and $\mathbf{g}_{i+1}=\mathbf{p s}$; hence, $\mathbf{g}_{i+1} \in \mathcal{C}_{2}$ is a cyclic shift of $\mathbf{a} \in \mathcal{C}_{1}$, which contradicts the fact that $\mathcal{C}_{2}$ contains no cyclic shifts of elements in $\mathcal{C}_{1}$.

The last information block design scheme we present is endowed with a simple encoding and decoding procedure. Let $\mathcal{A} \subseteq \mathbb{F}_{q}^{n}$ be a collection of sequences of length $n$ such that $\mathbf{e} \notin \mathcal{A}$, where $\mathbf{e}=(0, \ldots, 0,1)$. For the purpose of information block encoding, we may assume that $\mathcal{A}$ is a $\kappa$-WMU code with desired primer properties, constructed using cyclic error-correcting codes of minimum distance at least three, as described in Constructions 8 and 9. Let $N>n$ and define $\mathcal{I} \triangleq\left\{\mathbf{a} \mid \mathbf{a}_{1}^{n-1} \in \mathbb{F}_{q}^{n-1}, \mathbf{a}_{n}^{N} \in \mathbb{F}_{q-1}^{N-n+1}\right\}$, so that $|\mathcal{I}|=q^{n-1}(q-1)^{N-n+1}$. There is an efficient encoding scheme that maps elements of $\mathcal{I}$ to a set of sequences $\mathcal{C}_{\mathcal{A}}(N)$ of length $N$ that avoid all codesequences in $\mathcal{A}$.

Let $\mathbf{H}$ be a parity-check matrix of $\mathcal{A}$. Hence, a sequence $\mathbf{c}$ belongs to $\mathcal{A}$ if and only if $\mathbf{c H}=0$. Also, since $\mathbf{e} \notin \mathcal{A}$, one has $\mathbf{e H} \neq 0$.

To describe the encoding procedure, we define the following function $\phi$ : $\mathbb{F}_{q}^{n-1} \times\{0,1, \ldots, q-2\} \rightarrow \mathbb{F}_{q}$. Given $\mathbf{a} \in \mathbb{F}_{q}^{n-1}$ and $0 \leq i \leq q-2$, let $a_{n}=\phi(\mathbf{a}, i)$ be the $i$ smallest element in $\mathbb{F}_{q}$ such that $\left(\mathbf{a}, a_{n}\right) \mathbf{H} \neq 0$. For this function to be well-defined, it suffices to demonstrate that there are at least $q-1$ elements in $\mathbb{F}_{q}$ such that appending one of them to a yields a decoding syndrome not equal to 0 . Suppose otherwise. Then there exist distinct $u, u^{\prime}$ such that $(\mathbf{a}, u) \mathbf{H}=\left(\mathbf{a}, u^{\prime}\right) \mathbf{H}=0$. The last equality may be rewritten as $\left(\mathbf{0}, u-u^{\prime}\right) \mathbf{H}=\left(u-u^{\prime}\right) \mathbf{e H}=0$, contradicting the starting assumption.

Encoding a sequence a results in a sequence bobtained by concatenating the following sequences:

$$
\mathbf{b}_{i}= \begin{cases}a_{i} & \text { if } 1 \leq i \leq n-1 \\ \phi\left(\mathbf{b}_{i-n+1}^{i-1}, a_{i}\right), & \text { otherwise }\end{cases}
$$

It is straightforward to see that sequences obtained via this encoding method avoid all elements of the codebook $\mathcal{A}$.

### 2.9 Conclusions

Motivated by emerging code design problems for DNA-based data storage, we introduced the problem of address/primer sequence design. The address design problem reduces to constructing sequences that satisfy a new form of mutual uncorrelatedness and in addition, are balanced, at sufficient Hamming distance from each other and such that they avoid primer dimers which arise if a substring of an address sequence is also a substring of the reverse complement sequence of the same or another address sequence. Our main results are listed in Table 2.1. Given the constructed address sequences, we also described information encoding methods for sequences endowed with addresses, such that they avoid any address as a proper substring. This address avoidance property allows one to randomly access desired DNA sequences via simple and inexpensive PCR reactions.

Table 2.1: Summary of the optimal code constructions for various constrained WMU codes.

| Cons. No. | Name | Rate | Features | Comment |
| :--- | :---: | :---: | :---: | :---: |
| 1 | BAL_MU_2_n | $\frac{1}{2(n-1)}\binom{n}{n}$ | Binary, Balanced, MU |  |
| 2 | MU_q_n | $c_{q} \frac{q^{n}}{n}$ | $q$-ary, MU | $q \in\{2,4\}, c_{2}=0.04688, c_{4}=0.06152$ |
| 4 | WMU_q_n | $c_{q} \frac{q^{n}}{n-k+1}$ | $q$-ary, $\kappa$-WMU | $q \in\{2,4\}, c_{2}=0.04688, c_{4}=0.06152$ |
| 6 | BAL_ $\kappa-$ WMU_4_n | $c_{2} \frac{\left(\frac{n}{n} \frac{n}{n} 2^{n}\right.}{n-\kappa^{n+1}}$ | 4 -ary, Balanced, $\kappa$-WMU | $c_{2}=0.04688$ |
| 7 | f-APD_MU_2_n | $c_{3} \frac{2^{n}}{n}$ | Binary, $f$-APD, MU | For some constant $c_{3}>0$ |
| 10 | f-APD_BAL_MU_4_n | $c_{3} \frac{\left(\frac{n}{2} n_{2} 2^{n}\right.}{n}$ | 4 -ary, $f$-APD, Balanced, MU | For some constant $c_{3}>0$ |

## Chapter 3

# A REWRITABLE, RANDOM-ACCESS DNA-BASED STORAGE SYSTEM 

### 3.1 Introduction

Addressing the emerging demands for massive data repositories, and building upon the rapid development of technologies for DNA synthesis and sequencing, a number of laboratories have recently outlined architectures for archival DNA-based storage [4-6, 38, 39]. The architecture in [4] achieved a storage density of $87.5 \mathrm{~TB} /$ gram, while the system described in [5] raised the density to $2.2 \mathrm{~PB} /$ gram. The success of the latter method may be largely attributed to three classical coding schemes: Huffman coding, differential coding, and single parity-check coding [5]. Huffman coding was used for data compression, while differential coding was used for eliminating homopolymers (i.e., repeated consecutive bases) in the DNA strings. Parity-checks were used to add controlled redundancy, which in conjunction with four-fold coverage allows for mitigating assembly errors.

Due to dynamic changes in biotechnological systems, none of the three coding schemes represents a suitable solution from the perspective of current DNA sequencer designs: Huffman codes are fixed-to-variable length compressors that can lead to catastrophic error propagation in the presence of sequencing noise; the same is true of differential codes. Homopolymers do not represent a significant source of errors in Illumina sequencing platforms [40], while single parity redundancy or RS codes and differential encoding are inadequate for combating error-inducing sequence patterns such as long substrings with high GC content [40]. As a result, assembly errors are likely, and were observed during the readout process described in [5].

An even more important issue that prohibits the practical widespread use of the schemes described in $[4,5]$ is that accurate partial and random access to data is impossible, as one has to reconstruct the whole text in order to read
or retrieve the information encoded even in a few bases. Furthermore, all current designs support read-only storage. The first limitation represents a significant drawback, as one usually needs to accommodate access to specific data sections; the second limitation prevents the use of current DNA storage methods in architectures that call for moderate data editing, for storing frequently updated information and memorizing the history of edits. Moving from a read-only to a rewritable DNA storage system requires a major implementation paradigm shift, as:

1. Editing in the compressive domain may require rewriting almost the whole information content.
2. Rewriting is complicated by the current data DNA storage format that involves reads of length 100 bps shifted by 25 bps so as to ensure four-fold coverage of the sequence (See Figure 3.1 (a) for an illustration and description of the data format used in [5]). In order to rewrite one base, one needs to selectively access and modify four "consecutive" reads.
3. Addressing methods used in $[4,5]$ only allow for determining the position of a read in a file, but cannot ensure precise selection of reads of interest, as undesired cross-hybridization between the primers and parts of the information blocks may occur.

To overcome the aforementioned issues, we developed a new, randomaccess and rewritable DNA-based storage architecture based on DNA sequences endowed with specialized address strings that may be used for selective information access and encoding with inherent error-correction capabilities. The addresses are designed to be mutually uncorrelated and to satisfy the error-control running digital sum constraint [11,12]. Given the address sequences, encoding is performed by stringing together properly terminated prefixes of the addresses as dictated by the information sequence. This encoding method represents a special form of prefix-synchronized coding [8]. Given that the addresses are chosen to be uncorrelated and at large Hamming distance from each other, it is highly unlikely for one address to be confused with another address or with another section of the encoded blocks. Furthermore, selection of the blocks to be rewritten is made possible by the prefix encoding format, while rewriting is performed via two DNA editing techniques, the gBlock and OE-PCR (Overlap Extension PCR) methods [41, 42]. With the latter method, rewriting is done in several steps by using short and cheap primers. The first method is more efficient, but re-


Figure 3.1: (a) The scheme of [5] uses a storage format consisting of DNA strings that cover the encoded compressed text in fragments of length of 100 bps . The fragments overlap in 75 bps , thereby providing 4 -fold coverage for all except the flanking end bases. This particular fragmenting procedure prevents efficient file editing: If one were to rewrite the "shaded" block, all four fragments containing this block would need to be selected and rewritten at different positions to record the new "shaded" block. (b) The address sequence construction process we propose uses the notions of autocorrelation and cross-correlation of sequences [30]. A sequence is uncorrelated with itself if no proper prefix of the sequence is also a suffix of the same sequence. Alternatively, no shift of the sequence overlaps with the sequence itself. Similarly, two different sequences are uncorrelated if no prefix of one sequence matches a suffix of the other. Addresses are chosen to be mutually uncorrelated, and each 1000 bps block is flanked by an address of length 20 on the left and by another address of length 20 on the right (colored ends). (c) Content rewriting via DNA editing: the gBlock method [41] for short rewrites, and the cost efficient OE-PCR (Overlap Extension PCR) method [42] for sequential rewriting of longer blocks.
quires synthesizing longer and hence more expensive primers. Both methods were tested on DNA encoded Wikipedia entries of size 17 kB , corresponding to six universities, where information in one, two and three blocks was rewritten in the DNA encoded domain. The rewritten blocks were selected, amplified and Sanger sequenced [43] to verify that selection and rewriting are performed with $100 \%$ accuracy.

### 3.2 Results

The main feature of our storage architecture that enables highly sensitive random access and accurate rewriting is addressing. The rationale behind the proposed approach is that each block in a random access system must be equipped with an address that will allow for unique selection and amplification via DNA sequence primers.

Instead of storing blocks mimicking the structure and length of reads generated during high-throughput sequencing, we synthesized blocks of length 1000 bps tagged at both ends by specially designed address sequences. Adding addresses to short blocks of length 100 bps would incur a large storage overhead, while synthesizing blocks longer than 1000 bps using current technologies is prohibitively costly.

More precisely, each data block of length 1000 bps is flanked at both ends by two unique address blocks of length 20 bps each. These addresses are used to provide specificity of access, see Figure 3.1 (b). Note that different flanking addresses simplify the process of sequence synthesis. The remaining 960 bases in a block are divided into 12 sub-blocks of length 80 bps , with each block encoding six words of the text. The "word-encoding" process may be seen as a specialized compaction scheme suitable for rewriting, and it operates as follows. First, different words in the text are counted and tabulated in a dictionary. Each word in the dictionary is converted into a binary sequence of sufficient length to allow for encoding of the dictionary. For our current implementation and texts of choice this length was set to 21. Encodings of six consecutive words are subsequently grouped into binary sequences of length 126. The binary sequences are then translated into DNA blocks of length 80 bps using a new family of DNA prefix-synchronized codes described in the Methods section. Our choice for the number of jointly encoded words is
governed by the goal to make rewrites as straightforward as possible and to avoid error propagation due to variable code lengths. Furthermore, as most rewrites include words, rather than individual symbols, the word encoding method represents an efficient means for content update. Details regarding the counting and grouping procedure may be found in [17].

For three selected access queries, the 1000 bps blocks containing the desired information were "identified" (i.e., amplified) via primers corresponding to their unique addresses, PCR amplified, Sanger sequenced, and subsequently decoded.

Two methods were used for content rewriting. If the region to be rewritten had length exceeding several hundreds, new sequences with unique primers were synthesized as this solution represents a less costly alternative to rewriting. For the case that a relatively short substring of the encoded string had to be modified, the corresponding 1000 bps block hosting the string was identified via its address, amplified and the changes were generated via DNA editing.

Both the random access and rewriting protocols were tested experimentally on two jointly stored text files. One text file, of size 4 kB , contained the history of the University of Illinois at Urbana-Champaign (UIUC) based on its Wikipedia entry retrieved on $12 / 15 / 2013$. The other text file, of size 13 kB , contained the introductory Wikipedia entries of Berkeley, Harvard, MIT, Princeton, and Stanford, retrieved on 04/27/2014.

Encoded information was converted into DNA blocks of length 1000 bps synthesized by IDT (Integrated DNA Technologies), at a cost of $\$ 149$ per 1000 bps (see http://www.idtdna.com/pages/products/genes/gblocks-genefragments). The rewriting experiments encompassed:

1. PCR selection and amplification of one 1000 bps sequence and simultaneous selection and amplification of three 1000 bps sequences in the pool. All 32 linear 1000 bps fragments were stored in a mixed form, and the mixture was used as a template for PCR amplification and selection. The results of amplification were verified by confirming sequence lengths of 1000 bps via gel electrophoresis (Figure 3.2 (a)) and by randomly sampling $3-5$ sequences from the pools and Sanger sequencing them (Figure 3.2 (b)).
2. Experimental content rewriting via synthesis of edits located at various positions in the 1000 bps blocks. For simplicity of notation, we refer to the blocks in the pool on which we performed selection and editing as B1, B2,

b


Figure 3.2: (a) Gel electrophoresis results for three blocks, indicating that the length of the three selected and amplified sequences is tightly concentrated around 1000 bps , and hence correct. (b) Output of the Sanger sequencer, where all bases shaded in yellow correspond to correct readouts. The sequencing results confirmed that the desired sequences were selected, amplified, and rewritten with $100 \%$ accuracy.
and B3. Two primers were synthesized for each rewrite in the blocks, for the forward and reverse direction. In addition, two different editing/mutation techniques were used, gBlock and OE-PCR. gBlocks are double-stranded genomic fragments used as primers or for the purpose of genome editing, while OE-PCR is a variant of PCR used for specific DNA sequence editing via point editing/mutations or splicing. To demonstrate the plausibility of a cost-efficient method for editing, OE-PCR was implemented with general primers ( $\leq 60 \mathrm{bps}$ ) only. Note that for edits shorter than 40 bps , the mutation sequences were designed as overhangs in primers. Then, the three PCR products were used as templates for the final PCR reaction involving the entire 1000 bps rewrite. Figure 3.1 (c) illustrates the described rewriting process. In addition, a summary of the experiments performed is provided in Table 3.1.

Given that each basepair has weight roughly equal to 650 dalton ( $650 \times$ $1.67 \times 10^{-24}$ gram), and given that $27,000+5000=32,000$ bps were needed to encode a file of size $13+4=17 \mathrm{kB}$ in ASCII format, we estimate a potential storage density of $4.9 \times 10^{20} \mathrm{~B} / \mathrm{g}$ for our scheme. This density significantly surpasses the current state-of-the-art storage density of $2.2 \times 10^{15} \mathrm{~B} / \mathrm{g}$, as we avoid costly multiple coverage, use larger blocklengths and specialized word encoding schemes of large rate. A performance comparison of the three currently known DNA-based storage media is given in Table 3.2. We observe
that the cost of sequence synthesis in our storage model is clearly significantly higher than the corresponding cost of the prototype in [5], as blocks of length 1000 bps are still difficult to synthesize. This trend is likely to change dramatically in the near future, as within the last seven months, the cost of synthesizing 1000 bps blocks reduced almost 7 -fold. Despite its high cost, our system offers exceptionally large storage density and enables, for the first time, random access and content rewriting features. Furthermore, although we used Sanger sequencing methods for our small scale experiment, for large scale storage projects Next Generation Sequencing (NGS) technologies will enable significant reductions in readout costs.

### 3.3 Methods

### 3.3.1 Address Design and Encoding

To encode information on DNA media, we employed a two-step procedure. First, we designed address sequences of short length which satisfy a number of constraints that makes them suitable for highly selective random access [30]. Constrained coding ensures that DNA patterns prone to sequencing errors are avoided and that DNA blocks are accurately accessed, amplified and selected without perturbing or accidentally selecting other blocks in the DNA pool. The coding constraints apply to address primer design, but also indirectly govern the properties of the fully encoded DNA information blocks. The design procedure used is semi-analytical insofar as it combines combinatorial

Table 3.1: Selection, rewriting and sequencing results. Each rewritten 1000 bps sequence was ligated to a linearized pCRTM-Blunt vector using the Zero Blunt PCR Cloning Kit and was transformed into E. coli. The E. coli strains with correct plasmids were sequenced at ACGT, Inc. Sequencing was performed using two universal primers: M13F _20 (in the reverse direction) and M13R (in the forward direction) to ensure that the entire block of 1000 bps is covered.

| Sequence identifier - Editing Method | \# of sequence samples | Length of edits (bps) | Selection accuracy/error percentage |
| :--- | :---: | :---: | :---: |
| B1-M-gBlock | 5 | 20 | $(5 / 5) / 0 \%$ |
| B1-M-PCR | 5 | 20 | $(5 / 5) / 0 \%$ |
| B2-M-gBlock | 5 | 28 | $(5 / 5) / 0 \%$ |
| B2-M-PCR | 5 | 28 | $(5 / 5) / 0 \%$ |
| B3-M-gBlock | 5 | $41+29$ | $(5 / 5) / 0 \%$ |
| B3-M-PCR | 5 | $41+29$ | $(5 / 5) / 0 \%$ |

methods with limited computer search techniques. A unifying and highly technically charged coding approach will be reported elsewhere.

We required the address sequences to satisfy the following constraints:

- (C1) Constant GC content (close to $50 \%$ ) of all their prefixes of sufficiently long length. DNA strands with $50 \%$ GC content are more stable than DNA strands with lower or higher GC content and have better coverage during sequencing. Since encoding user information is accomplished via prefix-synchronization, it is important to impose the GC content constraint on the addresses as well as their prefixes, as the latter requirement also ensures that all fragments of encoded data blocks have balanced GC content.
- (C2) Large mutual Hamming distance, as it reduces the probability of erroneous address selection. Recall that the Hamming distance between two strings of equal length equals the number of positions at which the corresponding symbols disagree. An appropriate choice for the minimum Hamming distance is equal to half of the address sequence length ( 10 bps in our current implementation which uses length 20 address primers). It is worth pointing out that rather than using the Hamming distance, one could use the Levenshtein (edit) distance instead, capturing the smallest number of deletions, insertions and substitutions needed to convert one string into another. Unfortunately, many address design problems become hard to analyze under this distance measure, and are hence not addressed in this manuscript.
- (C3) Uncorrelatedness of the addresses, which imposes the restriction

Table 3.2: Comparison of storage densities for the DNA encoded information expressed in B/g (bytes per gram), file size, synthesis cost, and random access features of three known DNA storage technologies. Note that the density does not reflect the entropy of the information source, as the text files are encoded in ASCII format, which is a redundant representation system.

| Work | Church et al. [4] | Goldman et al. [5] | Our scheme [17] |
| :--- | :---: | :---: | :---: |
| Density | $0.7 \times 10^{15} \mathrm{~B} / \mathrm{g}$ | $2.2 \times 10^{15} \mathrm{~B} / \mathrm{g}$ | $4.9 \times 10^{20} \mathrm{~B} / \mathrm{g}$ |
| File size | 5.27 Mb | 739 kB | 17 kB |
| Cost | Not available | $\$ 12,600$ | $\$ 4,023$ |

that prefixes of one address do not appear as suffixes of the same or another address and vice versa. The motivation for this new constraint comes from the fact that addresses are used to provide unique identities for the blocks, and that their substrings should therefore not appear in "similar form" within other addresses. Here, "similarity" is assessed in terms of hybridization affinity. Furthermore, long undesired prefixsuffix matches may lead to read assembly errors in blocks during joint informational retrieval and sequencing.

- (C4) Absence of secondary (folding) structures, as such structures may cause errors in the process of PCR amplification and fragment rewriting.

Addresses satisfying constraints C1-C2 may be constructed via error-correcting codes with small running digital sum [11] adapted for the new storage system. Properties of these codes are discussed in Section 3.3.2. The notion of mutually uncorrelated sequences is described in 3.3.3; it was studied in an unrelated context under the name cross-bifix-free coding. We also introduce a new and more suitable version of cross-bifix-free codes termed weakly mutually uncorrelated sequences. Constructing addresses that simultaneously satisfy the constraints C1-C4 and determining bounds on the largest number of such sequences is prohibitively complex [10,14]. To mitigate this problem, we resort to a semi-constructive address design approach, in which balanced error-correcting codes are designed independently, and subsequently expurgated so as to identify a large set of mutually uncorrelated sequences. The resulting sequences are subsequently tested for secondary structure using mfold and Vienna [15].

Given two uncorrelated sequences as flanking addresses of one block, one of the sequences is selected to encode user information via a new implementation of prefix-synchronized encoding [13,15], described in 3.3.4. The asymptotic rate of an optimal single sequence prefix-free code is one. Hence, there is no asymptotic coding loss for avoiding prefixes of one sequence; we only observe a minor coding loss for each finite-length block. For multiple sequences of arbitrary structure, the problem of determining the optimal code rate is significantly more complicated and the rates have to be evaluated numerically, by solving systems of linear equations [13] as described in 3.3.4 and [17]. This system of equations leads to a particularly simple form for the
generating function of mutually uncorrelated sequences.

### 3.3.2 Balanced Codes and Running Digital Sums

One criterion for selecting block addresses is to ensure that the corresponding DNA primer sequences have prefixes with a GC content approximately equal to $50 \%$, and that the sequences are at large pairwise Hamming distance. Due to their applications in optical storage, codes that address related issues have been studied in a different form under the name of bounded running digital sum (BRDS) codes [11, 12]. A detailed overview of this coding technique may be found in [11].

Consider a sequence $a=a_{0}, a_{1}, a_{2}, \ldots, a_{l}, \ldots, a_{n}$ over the alphabet $\{-1,1\}$. We refer to $S_{l}(a)=\sum_{i=0}^{l-1} a_{i}$ as the running digital sum (RDS) of the sequence $a$ up to length $l, l \geq 0$. Let $D_{a}=\max \left\{\left|S_{l}(a)\right|: l \geq 0\right\}$ denote the largest value of the running digital sum of the sequence $a$. For some predetermined value $D>0$, a set of sequences $\{a(i)\}_{i=1}^{M}$ is termed a BRDS code with parameter $D$ if $D_{a(i)} \leq D$ for all $i=1, \ldots, M$. Note that one can define non-binary BRDS codes in an equivalent manner, with the alphabet usually assumed to be symmetric, $\{-q,-q+1, \ldots,-1,1, \ldots, q-1, q\}$, and where $q \geq 1$. A set of DNA sequences over $\{\mathrm{A}, \mathrm{T}, \mathrm{G}, \mathrm{C}\}$ may be constructed in a straightforward manner by mapping each +1 symbol into one of the bases $\{A, T\}$, and -1 into one of the bases $\{G, C\}$, or vice versa. Alternatively, one can use BRDS over an alphabet of size four directly.

To address the constraints C1-C2, one needs to construct a large set of BRDS codewords at sufficiently large Hamming distance from each other. Via the mapping described above, these codewords may be subsequently translated to DNA sequences with a GC content approximately equal to $50 \%$ for all sequence prefixes, and at the same Hamming distance as the original sequences.

Let $(n, C, d ; D)$ be the parameters of a BRDS error-correcting code, where $C$ denotes the number of codewords of length $n, d$ denotes the minimum distance of the code, while $\frac{\log C}{n}$ equals the code rate. For $D=1$ and $d=2$, the best known BRDS-code has parameters $\left(n, 2^{\frac{n}{2}}, 2 ; 1\right)$, while for $D=2$ and $d=1$, codes with parameters $\left(n, 3^{\frac{n}{2}}, 1 ; 2\right)$ exist. For $D=2$ and $d=2$, the best known BRDS code has parameters $\left(n, 2 \cdot 3^{\left(\frac{n}{2}\right)-1}, 2 ; 2\right)$ [12]. Note
that each of these codes has an exponentially large number of codewords, among which an exponentially large number of sequences satisfy the required correlation property C3, discussed next. Codewords satisfying constraint C4 were found by expurgating the BRDS codes via computer search.

### 3.3.3 Sequence Correlation

We describe next the notion of the autocorrelation of a sequence and describe mutually uncorrelated sequences (i.e., cross-bifix-free codes) and the new class of weakly mutually uncorrelated sequences. Mutually uncorrelated sequences, cross-bifix-free and non-overlapping codes were introduced and rediscovered many times, as witnessed by the publications [23, 44-46].

It was shown in [13] that the autocorrelation function is the crucial mathematical concept for studying sequences avoiding forbidden strings and substrings. In the storage context, forbidden strings correspond to the addresses of the blocks in the pool. In order to accommodate the need for selective retrieval of a DNA block without accidentally selecting any undesirable blocks, we find it necessary to also introduce the notion of mutually uncorrelated sequences.

Let $X$ and $Y$ be two words, possibly of different lengths, over some alphabet of size $q>1$. The correlation of $X$ and $Y$, denoted by $X \circ Y$, is a binary string of the same length as $X$. The $i$-th bit (from the left) of $X \circ Y$ is determined by placing $Y$ under $X$ so that the leftmost character of $Y$ is under the $i$-th character (from the left) of $X$, and checking whether the characters in the overlapping segments of $X$ and $Y$ are identical. If they are identical, the $i$-th bit of $X \circ Y$ is set to 1 , otherwise, it is set to 0 . For example, for $X=$ CATCATC and $Y=$ ATCATCGG, $X \circ Y=0100100$, as depicted below.

Note that in general, $X \circ Y \neq Y \circ X$, and that the two correlation vectors may be of different lengths. In the example above, we have $Y \circ X=00000000$. The autocorrelation of a word $X$ equals $X \circ X$.

In the example below, $X \circ X=1001001$.

$$
\begin{array}{llllllllllllllll}
X= & \text { C } & \text { A } & \text { T } & \text { C } & \text { A } & \text { T } & \text { C } & & & & & & & & \\
Y= & \text { A } & \text { T } & \text { C } & \text { A } & \text { T } & \text { C } & \text { G } & \text { G } & & & & & & & 0 \\
& & \text { A } & \text { T } & \text { C } & \text { A } & \text { T } & \text { C } & \text { G } & \text { G } & & & & & & 1 \\
& & \text { A } & \text { T } & \text { C } & \text { A } & \text { T } & \text { C } & \text { G } & \text { G } & & & & & 0 \\
& & & A & \text { T } & \text { C } & \text { A } & \text { T } & \text { C } & \text { G } & \text { G } & & & & 0  \tag{3.1}\\
& & & & & \text { A } & \text { T } & \text { C } & \text { A } & \text { T } & \text { C } & \text { G } & \text { G } & & & 1 \\
& & & & & \text { A } & \text { T } & \text { C } & \text { C } & \text { A } & \text { T } & \text { C } & \text { G } & \text { G } & 0
\end{array}
$$

Definition 3. $A$ sequence $X$ is self-uncorrelated if $X \circ X=10 \ldots 0$. A set of sequences $\left\{X_{1}, X_{2}, \ldots, X_{M}\right\}$ is mutually uncorrelated (cross-bifix-free) if each sequence is self-uncorrelated and if all pairs of distinct sequences satisfy $X_{i} \circ X_{j}=0 \ldots 0$ and $X_{j} \circ X_{i}=0 \ldots 0$.

Intuitively, correlation captures the extent to which prefixes of sequences overlap with suffixes of the same or other sequences. Furthermore, the notion of mutual uncorrelatedness may be relaxed by requiring that only sufficiently long prefixes do not match sufficiently long suffixes of other sequences. Sequences with this property, and at sufficiently large Hamming distance, eliminate undesired address cross-hybridization during selection and crosssequence assembly errors.

We provide the following extremely simple and easy-to-prove bound on the size of the largest mutually uncorrelated set of sequences of length $n$ over an alphabet of size $q=4$. The bounds show that there exist exponentially many mutually uncorrelated sequences for any choice of $n$, and the lower bound is constructive. Furthermore, the construction used in the bound "preserves" the Hamming distance and GC content, which distinguishes it from any known results in classical coding theory.

Theorem 9. Suppose that $\left\{X_{1}, \ldots, X_{M}\right\}$ is a set of $M$ pairwise mutually uncorrelated sequences of length $n$. Let $u(n)$ denote the largest possible value of $M$ for a given $n$. Then

$$
\begin{equation*}
4 \cdot 3^{\frac{n}{4}} \leq u(n) \leq 9 \cdot 4^{n-2} \tag{3.2}
\end{equation*}
$$

As an illustration, for $n=20$, the lower bound equals 972 . The proof of
the theorem is given in [17].
It remains an open problem to determine the largest number of address sequences that jointly satisfy the constraints C1-C4. We conjecture that the number of such sequences is exponential in $n$, as the numbers of words that satisfy C1-C3 and C4 [14] are exponential. Exponentially large families of address sequences are important indicators of the scalability of the system and they also influence the rate of information encoding in DNA.

Using a casting of the address sequence design problem in terms of a simple and efficient greedy search procedure, we were able to identify 1149 sequences for length $n=20$ that satisfy constraints C1-C4, out of which 32 pairs were used for block addressing. Another means to generate large sets of sequences satisfying the constraints is via approximate solvers for the largest independent set problem [47]. Examples of sequences constructed in the aforementioned manner and used in our experiments are listed in [17].

### 3.3.4 Prefix-Synchronized DNA Codes

In the previous sections, we described how to construct address sequences that can serve as unique identifiers of the blocks they are associated with. We also pointed out that once such address sequences are identified, user information has to be encoded in order to avoid the appearance of any of the addresses, sufficiently long substrings of the addresses, or substrings similar to the addresses in the resulting DNA codeword blocks. For this purpose, we developed new prefix-synchronized encoding schemes based on ideas presented in [10], but generalized to accommodate multiple sequence avoidance.

To address the problem at hand, we start by introducing comma-free and prefix-synchronized codes which allow for constructing codewords that avoid address patterns. A block code $\mathcal{C}$ comprising a set of codewords of length $N$ over an alphabet of size $q$ is called comma-free if and only if for any pair of not necessarily distinct codewords $a_{1} a_{2} \ldots a_{N}$ and $b_{1} b_{2} \ldots b_{N}$ in $\mathcal{C}$, the $N$ concatenations $a_{2} a_{3} \ldots a_{N} b_{1}, a_{3} a_{4} \ldots b_{1} b_{2}, \ldots, a_{N} a_{1} \ldots b_{N-2} b_{N-1}$ are not in $\mathcal{C}$ [13]. Comma-free codes enable efficient synchronization protocols, as one is able to determine the starting positions of codewords without ambiguity. A major drawback of comma-free codes is the need to implement an exhaustive search procedure over sequence sets to decide whether or not a given string of
length $n$ should be used as a codeword or not. This difficulty can be overcome by using a special family of comma-free codes, introduced by Gilbert [8] under the name prefix-synchronized codes. Prefix-synchronized codes have the property that every codeword starts with a prefix $\mathbf{p}=p_{1} p_{2} \ldots p_{n}$, which is followed by a constrained sequence $c_{1} c_{2} \ldots c_{\ell}$. Moreover, for any codeword $p_{1} p_{2} \ldots p_{n} c_{1} c_{2} \ldots c_{\ell}$ of length $n+\ell$, the prefix $\mathbf{p}$ does not appear as a substring of $p_{2} \ldots p_{n} c_{1} c_{2} \ldots c_{\ell} p_{1} p_{2} \ldots p_{n-1}$. More precisely, the constrained sequences of prefix-synchronized codes avoid the pattern $\mathbf{p}$ which is used as the address. First, we point out that in our work, no consideration is given to concatenations of codewords as DNA blocks are stored unattached. Furthermore, due to the choice of mutually uncorrelated addresses at large Hamming distance, we can encode each information block by avoiding only one of the address sequences, used for that particular block. Avoidance of all other address sequences is automatically guaranteed by the lack of correlation between the sequences, as demonstrated in the proof of our encoding method.

Specifically, for a fixed set $\mathcal{A}$ of address sequences of length $n$, we define the set $\mathcal{C}_{\mathcal{A}}(\ell)$ to be the set of sequences of length $\ell$ such that each sequence in $\mathcal{C}_{\mathcal{A}}(\ell)$ does not contain any string belonging to $\mathcal{A}$. Therefore, by definition, when $\ell<n$, the $\operatorname{set} \mathcal{C}_{\mathcal{A}}(\ell)$ is simply the set of strings of length $\ell$. Our objective is then to design an efficient encoding algorithm (one-to-one mapping) to encode a set $\mathcal{I}$ of messages into $\mathcal{C}_{\mathcal{A}}(\ell)$. For the sake of simplicity, we let $\mathcal{I}=\{0,1,2, \ldots,|\mathcal{I}|-1\}$.

In this scheme, we assume that $\mathcal{A}$ is mutually uncorrelated and all sequences in $\mathcal{A}$ end with the same base, which we assume without loss of generality to be G . We then pick an address $\mathbf{a}=a_{1} a_{2} \ldots a_{n} \in \mathcal{A}$ and define the following entities for $1 \leq i \leq n$,

$$
\begin{align*}
\bar{A}_{i} & =\{\mathrm{A}, \mathrm{~T}, \mathrm{C}\} \backslash\left\{a_{i}\right\}  \tag{3.3}\\
\mathbf{a}^{(i)} & =a_{1} \ldots a_{i} . \tag{3.4}
\end{align*}
$$

In addition, assume that the elements of $\bar{A}_{i}$ are arranged in increasing order, say using the lexicographical ordering $\mathrm{A} \prec \mathrm{C} \prec \mathrm{T}$. We subsequently use $\bar{a}_{i, j}$ to denote the $j$-th smallest element in $\bar{A}_{i}$, for $1 \leq j \leq\left|\bar{A}_{i}\right|$. For example, if $\bar{A}_{i}=\{\mathrm{C}, \mathrm{T}\}$, then $\bar{a}_{i, 1}=\mathrm{C}$ and $\bar{a}_{i, 2}=\mathrm{T}$.

Next, we define a sequence of integers $S_{n, 1}, S_{n, 2}, \ldots$ that satisfies the fol-
lowing recursive formula:

$$
S_{n, \ell}= \begin{cases}3^{\ell}, & 1 \leq \ell<n  \tag{3.5}\\ \sum_{i=1}^{n-1}\left|\bar{A}_{i}\right| S_{n, \ell-i}, & \ell \geq n\end{cases}
$$

For an integer $\ell \geq 0$ and $y<3^{\ell}$, let $\theta_{\ell}(y)=\{\mathrm{A}, \mathrm{T}, \mathrm{C}\}^{\ell}$ be a length- $\ell$ ternary representation of $y$. Conversely, for each $W \in\{\mathrm{~A}, \mathrm{~T}, \mathrm{C}\}^{\ell}$, let $\theta^{-1}(W)$ be the integer $y$ such that $\theta_{\ell}(y)=W$. We proceed to describe how to map every integer $\left\{0,1,2, \ldots, S_{n, l}-1\right\}$ into a sequence of length $\ell$ in $\mathcal{C}_{\mathcal{A}}(\ell)$ and vice versa. We denote these functions as $\operatorname{EnCODE}_{\mathbf{a}, \ell}$ and $\mathrm{DECODE}_{\mathbf{a}}$, respectively.

The steps of the encoding and decoding procedures are listed in Algorithm 1.

The following theorem is proved in [17].
Theorem 10. Let $\mathcal{A}$ be a set of mutually uncorrelated sequences that ends with the same base. Then for $\mathbf{a} \in \mathcal{A}$, Encode $\mathbf{a}_{\mathbf{a}, \ell}$ is an one-to-one mapping from $\left\{0,1,2, \ldots, S_{n, l}-1\right\}$ to $\mathcal{C}_{\mathcal{A}}(\ell)$. Moreover, for all $x \in\left\{0,1,2, \ldots, S_{n, l}-1\right\}$, $\operatorname{DECODE}_{\mathbf{a}}\left(\operatorname{ENCODE}_{\mathbf{a}, \ell}(x)\right)=x$.

A simple example describing the encoding and decoding procedure for the short address string $\mathbf{a}=$ AGCTG, which can easily be verified to be selfuncorrelated, is provided in [17].

The previously described $\operatorname{EnCODE}_{\mathbf{a}, \ell}(x)$ algorithm imposes no limitations on the length of a prefix used for encoding. This feature may lead to unwanted cross hybridization between address primers used for selection and the prefixes of addresses encoding the information. One approach to mitigate this problem is to "perturb" long prefixes in the encoded information in a controlled manner. For small-scale random access/rewriting experiments, the recommended approach is to first select all prefixes of length greater than some predefined threshold. Afterwards, the first and last quarter of the bases of these long prefixes are used unchanged while the central portion of the prefix string is cyclically shifted by half of its length. For example, for the address $\mathbf{a}=$ AGTAAGTCTCGCAGTCATCG, if the prefix $\mathbf{a}^{(16)}=$ AGTAAGTCTCGCAGTC appears as a subword, say $V$, in $X=\operatorname{EnCoDE}_{\mathbf{a}, \ell}(x)$ then $X$ is modified to $X^{\prime}$ by mapping $V$ to $V^{\prime}=$ AGTAATCGGTCCAGTC. This process of shifting is

```
Algorithm 1 Encoding and decoding
\(X=\operatorname{ENCODE}_{\mathbf{a}, \ell}(x)\)
begin
\(1 \quad n \leftarrow\) length (a);
2 if \((\ell \geq n)\)
\(3 \quad t \leftarrow 1\);
\(4 \quad y \leftarrow x\);
\(5 \quad\) while \(\left(y \geq\left|\bar{A}_{t}\right| S_{n, \ell-t}\right)\)
\(6 \quad y \leftarrow y-\left|\bar{A}_{t}\right| S_{n, \ell-t}\);
\(7 \quad t \leftarrow t+1\);
8 end;
\(9 \quad c \leftarrow\left\lfloor y / S_{n, \ell-t}\right\rfloor\);
\(10 \quad d \leftarrow y \bmod S_{n, \ell-t}\);
11 return \(\mathbf{a}^{(t-1)} \bar{a}_{t, c+1} \operatorname{ENCODE}_{\mathbf{a}, \ell-t}(d)\);
12 else
13 return \(\theta_{\ell}(x)\);
14 end;
end;
\(x=\operatorname{DECODE}_{\mathbf{a}}(X)\)
begin
\(n \leftarrow\) length (a);
\(2 \quad \ell \leftarrow\) length \((X)\);
3 assume that the input is \(X=X_{1} X_{2} \ldots X_{\ell}\);
4 if \((\ell<n)\)
5 return \(\theta^{-1}(X)\);
6 else
7 find \((u, v)\) such that \(\mathbf{a}^{(u-1)} \bar{a}_{u, v}=X_{1} \ldots X_{u}\);
8 return \(\left(\sum_{i=1}^{u-1}\left|\bar{A}_{i}\right| S_{n, \ell-i}\right)+(v-1) S_{n, \ell-u}+\operatorname{DECODE}_{\mathbf{a}}\left(X_{u+1} \ldots X_{\ell}\right)\);
9 end;
end;
```

illustrated below:


For an arbitrary choice of the addresses, this scheme may not allow for unique decoding Encode $\mathbf{E}_{\mathbf{a}, \ell}$. However, there exist simple conditions that can be checked to eliminate primers that do not allow this transform to be "unique".

Given the address primers created for our random access/rewriting experiments, we were able to uniquely map each modified prefix to its original prefix and therefore uniquely decode the readouts.

As a final remark, we would like to point out that prefix-synchronized coding also supports error-detection and limited error-correction. Errorcorrection is achieved by checking if each substring of the sequence represents a prefix or "shifted" prefix of the given address sequence and making proper changes when needed.

## Chapter 4

# PORTABLE AND ERROR-FREE DNA-BASED DATA STORAGE 

### 4.1 Introduction

Modern data storage systems primarily rely on optical and magnetic media to record massive volumes of data that may be efficiently accessed, retrieved, and copied [48]. Key features of existing recorders include random access and highly accurate data retrieval, supported by low-cost, real-time operations. Recently, these systems were challenged by the emergence of the first DNA- and polymer-based data storage platforms [4-6,17,49-51]. These new platforms have the potential to overcome existing bottlenecks of classical recorders as they offer ultrahigh storage densities on the order of $10^{15}-10^{20}$ bytes per gram of DNA $[4,5,17,49]$.

Experiments have shown that using DNA-based data storage one can record files as large as 200 MB [49], and ensure long-term data integrity through encapsulation [6] and coding [17, 49, 52, 53]. Data retrieval has exclusively been performed via high-throughput, high-cost sequencers, such as Illumina HiSeq $[4,5]$ and MiSeq $[6,49]$, because inexpensive portable sequencers such as MinION may introduce a prohibitively large number of deletion, insertion, and substitution errors. Some highly conservative estimates [54] for first-generation MinION sequencers suggested error rates as high as $30 \%$, which by far exceed those of optical recorders equal to 1 bit/10 TBs [55].

In order to make DNA-based data storage competitive with existing flash technologies, it is hence imperative to reduce synthesis cost by avoiding undesirable DNA sequence patterns; provide for random access, as otherwise selective reading becomes impossible; reduce sequencing cost by enabling portable readout systems; and offer extremely low error rates, comparable to those of classical recorders.

Our implementation addresses these challenges by introducing several unique, new concepts in bioinformatics, coding theory, and synthetic biology. In particular, it entails:

- Reducing the cost of synthesizing DNA containing user information via compression and subsequent constrained coding. Constrained coding eliminates substrings that may cause problems during synthesis, such as short repetitive substrings near the 3 ' and 5 ' ends of the string; it also limits the length of homopolymers (homopolymers are "runs" of consecutive symbols of the same kind, for example, AAAA) that cause both synthesis and sequencing problems, and balances out the GC content within short substrings of the encoded data.
- Providing random access by storing data in gBlock codewords (long DNA strings) equipped with addresses that allow for accurate selection via polymerase chain reactions (PCRs). The addresses have specialized properties, such as GC balanced content, large mutual Hamming distance, and weak mutual correlation. Controlled mutual correlation allows for avoiding matches of substrings of the address sequences in encoded data, and consequent erroneous codeword selection. The addresses are constructed mathematically using two binary component codes, without resorting to computer search.
- Ensuring portability of the system by using nanopore sequencers, such as MinION, while error-tolerance, which is challenging to accomplish with such architectures, is built-in via a new set of consensus sequence construction algorithms and asymmetric deletion-correcting codes tailormade for the nanopore channel. The new consensus method combines classical multiple sequence alignment methods with side information provided by the address sequences, and improves upon the state-of-the-art nanopolish platform, as it exploits the algebraic structure of the gBlock codewords. Furthermore, the deletion correcting codes are designed for errors that occur in consensus sequences, such as bounded magnitude errors in the homopolymer length sequences.

All these techniques are seamlessly combined into an integrated pipeline for data encoding (compression and constrained encoding) and postprocessing (address sequence identification, iterative sequence alignment and error
correction). On a broader scale, our work also presents experimental results regarding a new DNA-based data storage architecture that has many features of modern storage devices and paves the way for practical employment of macromolecular storage systems (see Table 4.1).

### 4.2 The Encoding Step

When compressed, data is stripped of its redundancy and errors in the compressed domain introduced either during synthesis or sequencing may cause catastrophic error propagation in the decompressed file. Even one single substitution error in the compressed domain may render the file unrecognizable. Hence, it may appear undesirable to perform data compression. Unfortunately, uncompressed files are significantly larger than their compressed counterparts, which implies significantly higher costs for synthesizing the information into DNA codewords. Our analysis detailed in Section 4.5 shows that the cost of adding redundancy for eliminating errors in the compressive domain is negligible compared to the cost of synthesizing uncompressed files. As a result, to accommodate large file sizes at low synthesis cost, the data is first compressed. To introduce the redundancy needed for different stages of error correction and to minimize the addressing overhead, we chose the DNA codeword length to be 1,000 base pairs (bp). This codeword length also offers good assembly quality of long files without additional coverage redundancy or word identifiers, and the overall smallest commercial synthesis cost. The prevalent method for encoding information into DNA relies on the use of oligos of length close to 100 nucleotides. Such a length introduces high loss in coding efficiency when addressing is performed, and

Table 4.1: Comparison of features/properties of current DNA-based storage platforms.

| Work | Random <br> access | Portability | Sequencing <br> technology | Sequencer <br> error rate | Error <br> correction/detection | Net density <br> $(\mathrm{bits} / \mathrm{bp})$ |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
| Church [4] | No | No | HiSeq | $0.1-0.3 \%$ | None | 0.83 |
| Goldman [5] | No | No | HiSeq | $0.1 \%$ | Detection | 0.33 |
| Yazdi [17] | Yes | No | Sanger | $0.05 \%$ | Correction | 1.575 |
| Grass [6] | No | No | MiSeq | $0.1 \%$ | Correction | 1.14 |
| Bornholt [49] | Yes | No | MiSeq | $0.1 \%$ | None | 0.88 |
| Erlich [52] | No | No | MiSeq | $0.1 \%$ | None | 1.55 |
| Yazdi [18] | Yes | Yes | MinION | $\mathbf{1 2 \%}$ | Correction | $\mathbf{1 . 7 2}$ |

underutilizes nanopore sequencing platforms. Some work has reported lower synthesis cost for oligo sequences, but this may be due to special arrangements made with the companies performing synthesis. To accommodate this choice of codeword length, as well as the inclusion of codeword address sequences, we grouped $123 \times 14=1,722$ consecutive bits in the compressed file and translated them into DNA blocks comprising $123 \times 8=984$ bases. We then balanced the GC-content of each substring of 8 bases via specialized constrained coding techniques that extend our previous results in terms of mitigating the need for computer search and providing mathematical characterizations of the addresses [16], outlined in Section 4.5. Balancing eliminates certain secondary structures, reduces synthesis errors, and helps to correct sequencing deletion errors. Each of the remaining 16 bases in a DNA codeword is used as a codeword address. As already pointed out, the purpose of the addressing method is to enable random access to codewords via highly selective PCR reactions. Selectivity is achieved by prohibiting the appearance of the address sequence anywhere in the encoded DNA blocks [16, 17]. Additional protection against deletion errors is provided via a new coding method we term homopolymer check codes. When coupled with balancing and subsequent read alignment steps, homopolymer checks lead to error-free readouts. A detailed description of the balancing and addressing schemes may be found in Section 4.5. Homopolymer checks are also discussed in the postprocessing step. All the encoding techniques are universal and therefore transparent to the type of data to be stored. The encoding pipeline is illustrated in Figure 4.1.

### 4.3 The Postprocessing Step

Postprocessing follows the physical process of sequencing via nanopores, as outlined in Section 4.5. The reads obtained using the MinION MkI sequencers have sequence-dependent substitution, deletion, and insertion errors, described in detail in Section 4.5. In practice, arbitrary combinations of deletions, insertions and substitution are harder to correct than deletions alone. Hence, we performed a consensus alignment procedure that "transforms" almost all insertion and substitution errors into deletion errors confined to homopolymers of certain lengths, and generates an estimate of the


Figure 4.1: The encoding stage. This stage involves compression, representation conversion, encoding into DNA, and subsequent synthesis. Each synthesized DNA codeword is equipped with one or two addresses. The encoding phase entails constrained coding, which limits the occurrence of the address block to one predefined position in the codeword only, and GC-content balancing of each substring of eight bases. Additional homopolymer checks are added directly into the string or stored on classical media; they correspond to only $0.02 \%$ of the data content.

DNA codeword based on the noisy reads.
In the first phase of postprocessing, we constructed a rough estimate of the DNA codewords. For this purpose, we used the address sequences to identify high-quality reads, i.e., those reads that contain an exact match with the given address. Aligning all reads instead of only high quality reads results in a large number of errors, and the quality of the reads is highly nonuniform. Next, we ran different multiple sequence alignment (MSA) algorithms on the identified high-quality reads and obtained different consensus sequences. For that purpose, we used Kalign, Clustal Omega, Coffee, and MUSCLE [56,57]. As multiple sequence alignment algorithms are traditionally designed for phylogenetic analysis, their parameters are inappropriate for modeling "mutations" introduced by nanopore sequencers. Hence, for each alignment method, new parameters were chosen by trial and error (see Section 4.5). The choice of the parameters was governed by the edit distance between the MSA consensus sequence and the corresponding DNA codeword.

As each alignment method produced a different consensus sequence, we formed an aggregate consensus. The aggregate consensus contains the "majority homopolymer" of the different MSA algorithms. As an example, if three MSA algorithms produced three consensus sequences, AAATTGCC, AATTTGCA, and AAATTGC, the majority homopolymer consensus would equal AAATTGCA, as two sequences contain a homopolymer of three As at the first position; two sequences contain a homopolymer of two Ts in the positions to follow; and all three sequences contain $G$ and $C$. Observe that A is included in the last position of the consensus.

In the second phase of postprocessing, we performed iterative alignment. By this stage, consensus sequences that estimate the original DNA blocks were identified, with errors mostly confined to deletions in homopolymers of length at least two. (See Section 4.5 for a detailed analysis.) To further improve the reconstruction quality of the blocks and thereby correct more errors, we performed one more round of BWA [58] alignment to match more reads with the corresponding estimates of their DNA codewords. Once this alignment was generated, two sequential checks were performed simultaneously on the bases. The checks included computing the majority consensus for each homopolymer length and determining whether the GC-balancing constraint for all substrings of length 8 was satisfied. More precisely, in the majority count, only homopolymer lengths that resulted in a correct balance


Figure 4.2: Postprocessing via sequence alignment and homopolymer correction. In the first phase, estimates of the DNA codewords are obtained by running several MSA algorithms on high-quality reads that contain an exact match with the address sequence. The second phase improves the estimate by employing an iterative method that includes BWA alignment and an error-correcting scheme.
were considered. This procedure is illustrated by an example in Section 4.5. Note that alignment does not require any coding redundancy, while balancing uses typical sequences and, as a result of this, has a high coding rate of 0.88. The alignment procedure is depicted in Figure 4.2.

In the final stage of postprocessing, we corrected deletion errors in homopolymers of length exceeding one. For this purpose, we used an errorcorrection scheme that parses the consensus sequence into homopolymers. As an example, the parsing of the sequence AATCCCGA into homopolymers AA, T, CCC, G, A gives rise to a homopolymer length sequence of $2,1,3,1,1$. Special redundancy that protects against asymmetric substitution errors is incorporated into the homopolymer length sequence. If two deletions were to occur in the example consensus, resulting in ATCCGA,
the homopolymer lengths would equal $1,1,2,1,1$. Here, we can recover the original length sequence $2,1,3,1,1$ from $1,1,2,1,1$ by correcting two bounded magnitude substitution errors. Note that the sequence of the homopolymer symbols is known from the consensus.

### 4.4 System Implementation

Because we tested address-based DNA data storage methods for ordinary text files [17], for practical implementation we focused on image data. Two images were used as test samples: a poster for the movie Citizen Kane (released in 1941), and a color Smiley Face emoji. The total size of the images was 10,894 bytes. The two images were compressed into a JPEG [59] format and then converted into a binary string using Base64 [60] (Base64 allows one to embed images into HTML files). The resulting size for the two compressed images was 3,633 bytes.

Through the previously described data encoding methods, the images were converted into 17 DNA blocks, out of which 16 blocks were of length 1,000 bp and one single block was of length 880 bp . Before the sequences were submitted for synthesis, they were tested by the IDT (Integrated DNA Technologies) gBlocks Gene Fragments Entry online software; they were then synthesized. The total cost of the testing and synthesis was $\$ 2,540$. IDT failed to synthesize one of the blocks because of a high GC-content in one substring of the address sequence, which was subsequently corrected through the addition of adapters at the two ends of the sequences. Based on information about this type of synthesis error, the sequence encoding procedure was modified to accommodate balancing of all short substrings of the DNA blocks, including the addresses, as previously described. This reduced the synthesis error rate and synthesis time.

The gBlocks representing our DNA codewords synthesized by IDT were mixed in equimolar concentration. One microgram of pooled gBlocks was used to construct the Oxford Nanopore libraries with the Nanopore Sequencing kit SQK-MAP006. The gBlock libraries were pooled and sequenced for 24 hours in a portable size MinION MkI using R7 chemistry and flowcell Mk 1 FLO-MAP103.12 with sequencing speed $75 \mathrm{bp} / \mathrm{s}$. All of the reads used in our subsequent testing were generated within the first 12 hours of sequencing.

Base-calling was performed in real time with the cloud service of Metrichor (Oxford, UK); the run produced a total of 6,660 reads that passed the filter. Table 4.2 provides a summary of the alignment results for all obtained reads, with respect to the reference genomes, along with the types of errors observed. It also illustrates how our new consensus formation algorithm significantly outperforms nanopolish. After the consensus formation stage, the error rate reduced to a mere $0.02 \%$ without any error-correction redundancy. It is important to observe that there are two levels of errors we are dealing with: per read and per consensus errors. Sequencing coverage clearly allows for the consensus error to be significantly smaller than the average per read error.

The three residual errors in the 17 consensus codewords were of the following type: in one block, two homopolymers AAAAAAA were erroneously decoded to AAAAA, while in one block, the homopolymer AAAAA was converted into AAAA. Error patterns where long homopolymer lengths are being reduced by one or two were also observed in the raw reads, as well as in other experiments that we will report on elsewhere. These asymmetric homopolymer errors were subsequently corrected using homopolymer checks, thereby producing error-free reconstructed images. The images reconstructed with and without homopolymer checks are shown in Figure 4.3 (e,f) and Figure 4.3 ( $\mathrm{c}, \mathrm{d}$ ), respectively.

The described implementation represents the only known random access DNA storage system that operates in conjunction with a MinION sequencer. Despite the fact that MinION has significantly higher error rates than Illumina sequencers and that random-access DNA systems typically require additional data redundancy, our DNA storage system has the highest reported information rate of 0.85 , storage density of $1.1 \times 10^{23}$ bytes/gram, and it offers error-free reconstruction.

### 4.5 Supplementary Information

Before data is encoded into DNA, it is compressed to reduce the file size and hence lower the cost of DNA synthesis. Any compression method is compatible with our encoding procedures and is left as a choice for the user. Given that the focus of this study is image storage, the compression method used


Figure 4.3: Image files used in our experiment. (a, b) show the raw images which were compressed, encoded and synthesized into DNA blocks. The Citizen Kane poster (photographed by Kahle, A., date of access: 17/11/2016) RKO Radio Pictures, not copyrighted per claim of Wikipedia repository) and Smiley Face emoji were of size 9,592 and 130.2 bytes, and had dimensions of $88 \times 109$ and $56 \times 56$ pixels, respectively. (c, d) show the recovered images after sequencing of the DNA blocks and the postprocessing phase without homopolymer error correction. Despite having only two errors in the Citizen Kane file, we were not able to recover any detail in the image. On the other hand, one error in the Smiley Face emoji did not cause any visible distortion. (e,f) show the image files obtained after homopolymer error correction, leading to an error-free reconstruction of the original file.
throughout our analysis and implementation is JPEG. The actual process of encoding refers to converting the compressed data into DNA codewords of length 1,000 bp (gBlocks). Note that we alternatively refer to the codewords as DNA blocks or gBlocks.

Despite a large amount of work on oligo-based data storage, we used gBlocks as they have many advantages over oligo sequences, including:

1. Long DNA sequences may be easily assembled with minimal or no coverage redundancy even when no addressing schemes are used. When using addresses, one effectively mitigates the need for assembly if the address sequences are carefully designed. Furthermore, for long codeword lengths, the overhead introduced by address sequences is negligible, as the length of the address is typically logarithmic in the length of the codeword.
2. Long DNA sequences offer largest coding efficiency as the address overhead for oligos may be as large as $20 \%$, while for gBlocks it amounts to a mere $2 \%$. Furthermore, all known coding schemes offer best performance for long block lengths.
3. Most third-generation sequencing technologies are being developed to accommodate long reads, and hence long-read sequencers will become the dominant market product in the near future; at the same time, a number of companies (e.g., IDT) offer smallest costs per base pair for long DNA sequences (for IDT, this cost equals 14 cents). Note that currently, a significantly lower cost for gBlocks may be obtained through Gen9 (https://www.gen9bio.com).

All modern storage systems offer random access, as otherwise one would have to retrieve the complete data content to read out even one bit of desired information. As every random-access data storage system is built around addressing schemes, we start with a description of address sequence construction and encoding. The results presented in the next section build upon our previous random-access architecture [17], with the important difference that this work presents the first explicit mathematical construction of exponentially large sets of addresses.

### 4.5.1 Data Encoding

We represent a DNA codeword (block) by a row vector over the four symbols alphabet $\mathcal{D}=\{A, C, G, T\}$. Moreover, we make frequent use of the following notation:

- If $\mathbf{a}=\left(a_{1}, \cdots, a_{n}\right)$ and $\mathbf{b}=\left(b_{1}, \cdots, b_{m}\right)$ are two vectors over the same alphabet $\mathcal{D}$, then ab is the vector obtained by appending $\mathbf{b}$ to $\mathbf{a}$, i.e., $\mathbf{a b}=\left(a_{1}, \cdots, a_{n}, b_{1}, \cdots, b_{m}\right)$.
- If $\mathbf{a}=\left(a_{1}, \cdots, a_{n}\right)$, then $\mathbf{a}_{i}^{j}=\left(a_{i}, \cdots, a_{j}\right)$ is used to denote the substring of a starting at position $i$ and ending at position $j, 1 \leq i \leq j \leq n$.
- If $\mathbf{a}=\left(a_{1}, \cdots, a_{n}\right)$, then $\mathbf{a}^{(i)}=\mathbf{a}_{i}^{n} \mathbf{a}_{1}^{i-1}$ is the cyclic shift of the vector $\mathbf{a}$ starting at position $1 \leq i \leq n$. Note that for the shift to be well defined we impose the initial condition $\mathbf{a}^{(1)}=\mathbf{a}$.

Next, we describe the encoding scheme used for constructing DNA codewords that may contain arbitrary user information. Each DNA block starts with a unique address of length $p$ base pairs; the remaining symbols in the block are referred to as the encoded information part of the block. Using the previously introduced notation we may write each DNA block as ab, where $\mathbf{a}$ denotes the address and $\mathbf{b}$ denotes the encoded information part. We also denote the set of all the allowed address sequences of length $p$ and all valid encoded information sequences of length $N-p$ by $\mathcal{A}$ and $\mathcal{B}$, respectively. Hence, $\mathbf{a} \in \mathcal{A}$ and $\mathbf{b} \in \mathcal{B}$ and the total codelength equals $N$.

The encoding goals are three-fold. The first goal is to design a set of addresses and encoded information sequences such that for any two not necessarily distinct addresses $\mathbf{a}, \mathbf{a}^{\prime} \in \mathcal{A}$, and for any encoded information sequence $\mathbf{b} \in \mathcal{B}, \mathbf{a}^{\prime}$ does not appear as a substring anywhere in $\mathbf{a}_{2}^{p} \mathbf{b}$. In other words, if $\mathbf{c}=\mathrm{ab}$ then $\mathbf{a}^{\prime} \neq \mathbf{c}_{i}^{p+i-1}$, for $i \geq 2$. The second goal is to make the addresses as distinguishable as possible, and ideally the solution for this problem would be to use sequences at large edit distance. (Recall that the edit distance between two strings equals the smallest number of deletions, insertions and substitutions needed to convert one string into the other.) Working with the edit distance causes a number of mathematical challenges, and we instead adopt an approach that requires the Hamming distance between pairs of different
addresses to be as large as possible (the Hamming distance between two sequences $\mathbf{a}, \mathbf{a}^{\prime} \in \mathcal{A}$ is defined as $\left.d_{H}\left(\mathbf{a}, \mathbf{a}^{\prime}\right)=\left|\left\{i: a_{i} \neq b_{i}\right\}\right|\right)$. The third goal is to balance out the GC-content of sequences in $\mathcal{B}$ locally. Local balancing refers to balancing the GC-content in each relatively short substring of the DNA codewords. Local balancing simplifies the process of synthesis, enables error correction and makes rewriting easier to accommodate with short primers. These properties will be discussed further in the experimental study section.

More formally, with respect to the first design goal, our address sequences are required to satisfy the following property.

Property 1. For any two not necessarily distinct addresses $\mathbf{a}, \mathbf{a}^{\prime} \in \mathcal{A}$, and for any encoded information sequence $\mathbf{b} \in \mathcal{B}, \mathbf{a}^{\prime}$ is not allowed to appear as a substring in $\mathbf{a}_{2}^{p} \mathbf{b}$, i.e., $\mathbf{a}^{\prime} \neq(\mathrm{ab})_{i}^{p+i-1}$, for $i \geq 2$.

We now describe a simple new construction for sequences that satisfy Property 1. Another sequence family with related properties was introduced in Chapter 2. Given a binary set (code) $\mathcal{C} \subseteq\{0,1\}^{n}$, let $\mathcal{C}^{\text {cyclic }}$ denote the set of all cyclic shifts of elements in $\mathcal{C}$, that is if $\mathbf{a} \in \mathcal{C}$ then $\mathbf{a}^{(i)} \in \mathcal{C}^{\text {cyclic }}$, for $1 \leq i \leq n$. Consider two binary codes $\mathcal{C}_{1}, \mathcal{C}_{2} \subseteq\{0,1\}^{n}$ such that

- $\mathcal{C}_{1}^{\text {cyclic }} \cap \mathcal{C}_{2}=\varnothing$.
- If $\mathbf{a} \in \mathcal{C}_{1}$, then $\mathbf{a}^{(i)} \notin \mathcal{C}_{1}$ for $2 \leq i \leq n$.

Given these two binary codes, we first construct the set of addresses $\mathcal{A} \subseteq\{A, C, G, T\}^{2 n}$ according to:

$$
\begin{equation*}
\mathcal{A}=\left\{\psi(\mathrm{ff}, \mathbf{g}) \mid \mathbf{f} \in \mathcal{C}_{1}, \mathbf{g} \in \mathcal{C}_{3}\right\} . \tag{4.1}
\end{equation*}
$$

Note that the address length is $p=2 n, \mathcal{C}_{3}$ is a binary code of length $p=2 n$ whose properties may be chosen so as to enforce a minimum Hamming distance on the addresses, and $\psi(\bullet, \bullet)$ is a bijection that maps two binary strings into one DNA codeword. More precisely, if $\mathbf{f}=\left(f_{1}, \cdots, f_{m}\right) \in\{0,1\}^{m}$ and $\mathbf{g}=\left(g_{1}, \cdots, g_{m}\right) \in\{0,1\}^{m}$, then $\psi(\mathbf{f}, \mathbf{g})=\mathbf{h} \in\{A, C, G, T\}^{m}$, where $\mathbf{h}=$ $\left(h_{1}, \cdots, h_{m}\right)$ is obtained according to the rules:

$$
h_{i}=\left\{\begin{array}{ll}
A, & \text { if }\left(f_{i}, g_{i}\right)=(0,0)  \tag{4.2}\\
T, & \text { if }\left(f_{i}, g_{i}\right)=(0,1) \\
C, & \text { if }\left(f_{i}, g_{i}\right)=(1,0) \\
G, & \text { if }\left(f_{i}, g_{i}\right)=(1,1)
\end{array}, \text { for } 1 \leq i \leq m\right.
$$

The set of encoded information sequences $\mathcal{B}$ is defined as a collection of DNA strings of the form

$$
\begin{equation*}
\mathcal{B}=\left\{\mathbf{s}_{1} \cdots \mathbf{s}_{m} \mid \text { for } m \geq 1, \mathbf{s}_{i} \in \mathcal{S}\right\} . \tag{4.3}
\end{equation*}
$$

where the set $\mathcal{S}$ equals

$$
\begin{equation*}
\mathcal{S}=\left\{\psi(\mathbf{f}, \mathbf{g}) \mid \mathbf{f} \in \mathcal{C}_{2}, \mathbf{g} \in \mathcal{C}_{4}\right\} \tag{4.4}
\end{equation*}
$$

Here $\mathcal{C}_{2}, \mathcal{C}_{4}$ are binary code of length $n$ whose properties may be tuned to accommodate balancing and Hamming distance constraints.

The next theorem shows that the sequences in $\mathcal{A}$ and $\mathcal{B}$ satisfy Property 1.

Theorem 1. $\mathcal{A}$ and $\mathcal{B}$ defined in 1 and 3 , satisfy Property 1 .
Proof. Consider two arbitrary address sequences $\mathbf{a}, \mathbf{a}^{\prime} \in \mathcal{A}$ and an encoded information sequence $\mathbf{b} \in \mathcal{B}$. For simplicity of notation assume that $\mathbf{a}=\psi(\mathbf{f f}, \mathbf{g})$, $\mathbf{a}^{\prime}=\psi\left(\mathbf{f}^{\prime} \mathbf{f}^{\prime}, \mathbf{g}^{\prime}\right)$ and that $\mathbf{b}=\mathbf{s}_{1} \cdots \cdot \mathbf{s}_{m}$, where $\mathbf{s}_{i}=\psi\left(\mathbf{f}_{i}, \mathbf{g}_{i}\right)$ for $1 \leq i \leq m$. Since the mapping $\psi(\bullet, \bullet)$ defined above is one-to-one, the claimed result will follow if we can prove that $\mathbf{f}^{\prime} \mathbf{f}^{\prime}$ does not appear as a substring anywhere in $\mathbf{f}_{2}^{n} \mathrm{ff}_{1} \cdots \mathbf{f}_{m}$. This can be done by checking two conditions:

- $\mathbf{f}^{\prime}$ does not appear as a substring anywhere in $\mathbf{f}_{2}^{n} \mathbf{f}$. Otherwise, $\mathbf{f}^{\prime}$ would have to be a proper cyclic shift of $\mathbf{f}$, i.e., there would exist an index $2 \leq i \leq n$ such that $\mathbf{f}^{\prime}=\mathbf{f}^{(i)}$. But then $\mathbf{f}^{\prime}=\mathbf{f}^{(i)} \notin \mathcal{C}_{1}$, which contradicts the assumption that $\mathbf{f}, \mathbf{f}^{\prime} \in \mathcal{C}_{1}$.
- $\mathbf{f}^{\prime} \mathbf{f}^{\prime}$ does not appear as a substring in $\mathrm{ff}_{1} \cdots \mathbf{f}_{m}$. Otherwise, there would exist a sequence $\mathbf{f}_{i}$ for $1 \leq i \leq m$ that appears as a substring in $\mathbf{f}^{\prime} \mathbf{f}^{\prime}$. This would in turn imply that $\mathbf{f}_{i}$ is a cyclic shift of $\mathbf{f}^{\prime}$, i.e., $\mathbf{f}_{i} \in \mathcal{C}_{1}^{\text {cyclic }}$. This contradicts our initial assumptions that $\mathbf{f}_{i} \in \mathcal{C}_{2}$ and $\mathcal{C}_{1}^{\text {cyclic }} \cap \mathcal{C}_{2}=$ $\varnothing$.

We now turn our attention to the practical implementation of this scheme and the choice of various code components and parameters. In our experiments, we set $n=8$ and designed addresses of length $p=16$. We also selected $\mathcal{C}_{2}$ and $\mathcal{C}_{4}$ to be the set of all balanced binary words (i.e., words with half 0 s and half 1 s ) and all binary words of length 8 , respectively. Note that $\left|\mathcal{C}_{2}\right|=\left(\frac{8}{4}\right),\left|\mathcal{C}_{4}\right|=2^{8}$. In addition, according to the defining relation (2), $\psi(\mathbf{f}, \mathbf{g})$ is a GC-balanced DNA string if and only if $\mathbf{f}$ is a balanced binary word. So, since $\mathcal{C}_{2}$ is a balanced set, $\mathcal{S}$ is just the set of all GCbalanced DNA strings of length 8 bp . The cardinality of the latter set equals $|\mathcal{S}|=\left(\frac{8}{4}\right) \times 2^{8}$, and $\mathcal{B}$ is formed by stringing together elements from $\mathcal{S}$. An important observation which will subsequently be used in our postprocessing step is that balancing the GC-content of each substring of a given length limits the longest homopolymer length to the same value - in our case, 8 .

To construct a large set of addresses, one may select $\mathcal{C}_{1}=\{10000000,01111111\}$ and let $\mathcal{C}_{3}$ be the set of all binary strings of length 16 . In this case, the number of addresses equals $|\mathcal{A}|=2^{17}$. Alternatively, one may select the code $\mathcal{C}_{3}$ to have large Hamming distance which would result in the same Hamming distance for the constructed addresses (in our experiments, we used an extended $[16,11,4] \mathrm{BCH}$ code for $\left.\mathcal{C}_{3}\right)$. It is easy to verify that in this case $\mathcal{A}$ and $\mathcal{B}$ satisfy the condition of Property 1. Also of importance are the numerically computed Hamming distances between the chosen addresses of the blocks and all the substrings of the encoded DNA blocks of the same length. For each address of length 16 we hence recorded the distance between the address and all the substrings in the codewords. We then identified the "most similar" substrings for the address sequences in terms of the Hamming distance and replaced the later if needed to achieve larger discriminative power during PCR reactions.

Using the described method, we designed 17 DNA blocks, each of length $1,000 \mathrm{bp}$, containing 984 bp of encoded information involving a black and white movie poster (Citizen Kane) and a color image (Smiley Face). Here, $\mathcal{B}$ was formed by grouping together 123 balanced strings from $\mathcal{S}$. The block addresses are listed in Table 4.3, along with the average and minimum Hamming distances between our chosen addresses and encoded substrings. Note that the choice of the BCH code $\mathcal{C}_{3}$ is justified by the fact that the minimum Hamming distance between a DNA address and any encoded information substring equals $d_{H}=4$.

Table 4.2: Summary of the readout data, along with the number and type of errors encountered in the reads.

| Block (length) | Number of reads | Sequencing coverage depth |  | Number of errors: (substitution, insertion, deletion) |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Average | Maximum | Per read (average) | Consensus |  |
|  |  |  |  |  | Nanopolish | Our method |
| $1(1,000)$ | 201 | 176.145 | 192 | (107, 14, 63) | $(14,32,5)$ | $(0,0,2)$ |
| $2(1,000)$ | 407 | 315.521 | 349 | (123, 12, 70) | (75, 99, 40) | (0, 0, 0) |
| $3(1,000)$ | 490 | 460.375 | 482 | $(80,23,42)$ | (10, 45, 0) | (0, 0, 0) |
| $4(1,000)$ | 100 | 81.763 | 87 | $(69,18,37)$ | $(1,54,1)$ | $(0,0,0)$ |
| $5(1,000)$ | 728 | 688.663 | 716 | $(88,20,48)$ | $(4,45,3)$ | (0, 0, 0) |
| $6(1,000)$ | 136 | 120.907 | 129 | (79, 21, 42) | (390, 190, 61) | (0, 0, 0) |
| $7(1,000)$ | 577 | 542.78 | 566 | $(83,26,41)$ | $(3,31,3)$ | $(0,0,0)$ |
| $8(1,000)$ | 217 | 199.018 | 207 | (83, 20, 46) | $(18,51,1)$ | (0, 0, 0) |
| $9(1,000)$ | 86 | 56.828 | 75 | $(60,16,30)$ | (404, 92, 54) | (0, 0, 0) |
| $10(1,000)$ | 442 | 396.742 | 427 | (91, 18, 52) | $(388,100,59)$ | $(0,0,0)$ |
| $11(1,000)$ | 114 | 101.826 | 110 | (79, 23, 42) | $(16,23,18)$ | $(0,0,0)$ |
| $12(1,000)$ | 174 | 162.559 | 169 | (94, 23, 50) | $(14,59,1)$ | (0, 0, 0) |
| $13(1,060)$ | 378 | 352.35 | 366 | $(88,26,44)$ | $(7,55,4)$ | (0, 0, 0) |
| $14(1,000)$ | 222 | 189.918 | 203 | (69, 22, 34) | $(15,34,3)$ | (0, 0, 0) |
| $15(1,000)$ | 236 | 222.967 | 232 | $(92,24,45)$ | $(15,46,2)$ | (0, 0, 0) |
| $16(1,000)$ | 198 | 182.99 | 195 | $(103,16,61)$ | $(15,62,4)$ | $(0,0,1)$ |
| 17 (880) | 254 | 240.273 | 250 | (77, 19, 42) | $(359,95,44)$ | (0, 0, 0) |

Table 4.3: Block addresses and Hamming distance profiles of the addresses vs DNA blocks. Only one address is used.

| Block <br> (length) | Average Hamming <br> distance | Minimum Hamming <br> distance | Forward address |
| :--- | :---: | :---: | :---: |
| $1(1,000 \mathrm{bp})$ | 8.75 | 4 | TATGCGCGACCCCCCT |
| $2(1,000 \mathrm{bp})$ | 8.62 | 4 | CCGAATATCAAAAATC |
| $3(1,000 \mathrm{bp})$ | 9.27 | 5 | AATCCGCGACCCCCGA |
| $4(1,000 \mathrm{bp})$ | 9.28 | 5 | CCCAATATCAAAATAG |
| $5(1,000 \mathrm{bp})$ | 9.28 | 5 | AAACCGCGACCCCGCT |
| $6(1,000 \mathrm{bp})$ | 9.34 | 5 | GCCTATATCAAAATTC |
| $7(1,000 \mathrm{bp})$ | 9.30 | 6 | TAAGCGCGACCCCGGA |
| $8(1,000 \mathrm{bp})$ | 9.33 | 5 | CGCAATATCAAATAAC |
| $9(1,000 \mathrm{bp})$ | 9.33 | 5 | ATACCGCGACCCGCCA |
| $10(1,000 \mathrm{bp})$ | 9.32 | 5 | GGCTATATCAAATATG |
| $11(1,000 \mathrm{bp})$ | 9.27 | 5 | TTAGCGCGACCCGCGT |
| $12(1,000 \mathrm{bp})$ | 9.29 | 5 | GGGTATATCAAATTAC |
| $13(1,060 \mathrm{bp})$ | 9.35 | 4 | TTTGCGCGACCCGGCA |
| $14(1,000 \mathrm{bp})$ | 9.2 | 5 | CGGAATATCAAATTTG |
| $15(1,000 \mathrm{bp})$ | 9.2 | 5 | ATTCCGCGACCCGGGT |
| $16(1,000 \mathrm{bp})$ | 9.01 | 5 | AAAGCCCCTGCGCCGT |
| $17(880 \mathrm{bp})$ | 9.23 | 5 | TTACCGCCTCCCCCCA |



Figure 4.4: Thermodynamic profiles of sequences with undesirable GC-content. (a) The profile of the DNA block with large GC-content in an address substring. (b) The profile of the sequence upon addition of terminal adapters. It may be observed that the GC-content of the adapters is more balanced that that of the address sequence.

### 4.5.2 DNA Synthesis

The blocks constructed using the binary component codes were synthesized by Integrated DNA Technology (IDT). Before performing the actual writing process, the synthesis complexity of each DNA codeword was tested by a special purpose IDT software (http://www.idtdna.com/pages/products/genes/gblocks-gene-fragments). It appears that synthesis errors are highly correlated with the total repeat density (direct, inverse and palindromic repeat elements), extreme variations in GC-content, and secondary structures, especially if such properties hold near the 3' and 5' termini of the sequence. All these issues were accommodated to the largest extent possible via our balancing and address selection procedures.

Although all the 17 DNA blocks passed the initial complexity test, IDT was not able to synthesize one of the blocks due to high GC-content in one substring of the corresponding address sequence, see Figure 4.4 (a). To overcome the problem, we redesigned the address and in addition, requested terminal adapters to be added to the original DNA block in order to check if the hard-to-synthesize sequence had other undesirable properties that may arise only during sequencing. The sequences augmented by adapters passed all subsequent tests without issues. Note that IDT maintains a small subset of adapters which have been optimized to be compatible with the gBlock synthesis process, see Figure 4.4 (b). These adapters can be appended to the 5 ' and 3 ' ends of any sequence and may increase synthesis feasibility whenever complex secondary structures are encountered.

### 4.5.3 DNA Sequencing

Nanopore sequencers are built around nanopores, small holes with an internal diameter of the order of 1 nm . The idea behind nanopore sequencing is that when a nanopore is immersed in a conducting fluid and a voltage is applied across it, an electric current due to conduction of ions through the nanopore is induced. The current is very sensitive to the size and shape of the nanopore. If DNA strands or other molecules pass through the nanopore, they create a characteristic change in the magnitude of the current through the nanopore, and one can use the generated signal and sequence the DNA strands [61]. MinION is the first commercial sequencer using nanopore technology, and it was released by Oxford Nanopore Technologies in 2014. It is also very small compared to the other sequencing devices available, $10 \times 3 \times 2 \mathrm{~cm}$ in size and it weighs just 90 g . DNA sequencing on MinION is performed by first adding the sample to the flowcell that contains all the nanopores that are needed to perform sequencing. When DNA strings pass through nanopores, there will be a change in electrical current, and this current in the nanopore is measured and sampled by a sensor several thousand times per second. Base-calling is performed on 5 -mers or 6 -mers [62]. In our experiments, we used the R7 chemistry, with a throughput of at least 75 base pairs/s. The latest model, R9, may only improve the system performance compared to R7, as it produces single strand reads, deep-learning base calling, and simpler workflows. Still, none of these features were critical for demonstrating the power of our coding schemes. The readouts of R 9 are also created with smaller delay, but yet again, the delay parameter does not influence our findings.

### 4.5.4 Random Access (RA)

RA is performed using PCR reactions with primers corresponding to the addresses of the selected blocks. The protocols for RA experiments were described in detail in Chapter 3.

### 4.5.5 Reconstructing Sequences from Traces

Reconstructing a sequence from a number of reads that were generated by passing the same sequence multiple times through different "versions" of the nanopore requires using specialized tools from bioinformatics and theoretical computer science. Here, a version may refer to a nanopore used at a certain time or a different nanopore. As Oxford Nanopore MinION uses a biological pore, the more the pore is used, the more likely that the quality of the reads will differ. It is currently not known if the system relies on one or multiple pores to perform sequencing. The main challenge is to identify which pores gave unacceptable quality readouts and perform accurate sequence estimation based on high quality reads only.

Variants of the latter problem have been studied in the computer science literature under the name of sequence reconstruction via traces Figure 4.5. The general traces problem may be stated as follows: Reconstruct a string $\mathbf{x}$ of length $n$ from a set of $m$ subsequences generated by randomly editing symbols in $x$ with a given probability fixed for all subsequences [63]. In a separate work [64], the authors showed that by focusing on edits of the form of deletions, which occur with a small constant probability, $m=\operatorname{poly}(n)$ traces suffice for exact reconstruction. Here, poly $(n)$ stands for a polynomial expression in $n$. Later, this result was improved [65] to show that for certain alphabet sizes, even a sub-polynomial number of traces suffices. Both lines of work also described several algorithms for high probability, error-free reconstruction. Other works $[66,67]$ considered the reconstruction problem for the case that the traces represent nanopore DNA reads. In [66], the authors studied the noiseless setup where each read has some fixed length and a noisy setup in which the reads were subjected to substitution errors. In [67], the authors consider a different type of noise model that more realistically captures the properties of nanopore sequencing technologies. In particular, they gave bounds on the parameter $(m)$ necessary for reliable reconstruction for the case that edits change the homopolymer lengths. Another related line of work includes [24], where accurate sequence reconstructing via new profile coding techniques was proposed. And although the model studied in [67] is relevant to our approach, it cannot be applied directly due to the fact that the pores introduce a complex mixture of errors that are not necessarily confined to homopolymer changes. In addition, the noise parameters of the


Figure 4.5: Nanopore-based sequence estimation as a sequence reconstruction from traces problem. A trace refers to the readout of a "pore" at a certain time point, or the readout from one out of multiple pores. During each read time the pore may behave differently, introducing sequence-dependent substitution, deletion and insertion errors at different rates.
pore(s) observed during different readout times may be very different, and they depend on the particular sequence structure as well. Hence, we propose a two-stage procedure that will allow us to mitigate all the aforementioned problems.

### 4.5.6 Consensus Sequence Formation via Iterative Alignment

Given the different read qualities, it is important to first identify reads with small fractions of deletion and insertion errors which will be used in the first step of reconstruction. For this purpose, we use the idea of pilot sequences. A pilot sequence is used to assess the performance of a noisy channel (e.g., wireless channel) as it is known both to the transmitter and receiver. The transmitter sends the pilot sequence to the receiver which can, based on knowledge of the sequence, estimate the probability of error. If the error probability is small, the transmitter asks for the information sequence to be transmitted; otherwise, the transmitter delays transmission. Clearly, as the addresses of the sequences of the DNA codewords are known to the
user, they may serve the role of pilot sequences. More precisely, based on the quality of the address sequence one may decide to use a read in the reconstruction process or not. Once good-quality reads are identified, the obvious next step is to perform alignment of the reads and then form a consensus sequence through majority voting at each sequence coordinate or for each homopolymer. This procedure may be performed in multiple rounds, using more and more reads to correct consensus errors as needed, especially in areas where the consensus has low confidence values; see Figure 4.6 obtained using the Tablet Software [68]. Furthermore, as pointed out in the main text, the parameters of various MSA algorithms need to be carefully tuned to produce the best alignment results, and are shown in the Table 4.4.

To better explain our iterative scheme for finding the consensus sequence using majority rules and the balancing property, we provide a simple example. This example involves three reads and the rough estimate of the DNA block after running the first alignment phase described in the main text. The three reads (read1, read2 and read3) and the running consensus estimate (c_est) are listed below:

```
c_est TTCACCCCAAAACCCGAAAACCGCTTCACGA
read_1 TTCACCCAAAACCGAAAACCGCTTCACGA
read_2 TTCACCCCAAAACCCGAAAACCGCTTCAGCGA
read_3 TTCACCCAAAAACCCGAAAACCGCTTCAGCGA
```

By running the MATLAB built-in MSA algorithm [69] we obtain the fol-

Table 4.4: Parameter choices for MSA algorithms used to build a consensus of consensus sequences. The parameters are selected to achieve best empirical performance for nanopore read alignments.

| Software | Parameters |
| :--- | :---: |
| Kalign | Gap open penalty $=\{5,11\}$, |
|  | Gap extension penalty $=\{0.2,0.85\}$, |
| Terminal Gap Penalties $=\{0.1,0.45\}$, |  |
| Clustal Omega | Bonus Score $=\{5.2,5,1,0\}$ |
| Coffee | Number of combined iterations $=\{0,1,2,3,4,5\}$ |
| MUSCLE | Default |
| MAFFT | Default |
| BWA | Default |
| MATLAB | $\mathrm{K}=14, \mathrm{~W}=20, \mathrm{r}=10, \mathrm{~A}=1, \mathrm{~B}=1, \mathrm{O}=1, \mathrm{E}=1, \mathrm{~L}=0$ |



Figure 4.6: Screen shot from the Tablet software showing poor consensus quality substrings formed during the iterative alignment phase of data recovery.
lowing result:

| c_est | TT | C | A | $C C C C$ | $A A A A-$ | $C C C$ | $G$ | $A A A A$ | $C C$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| read_1 | TT | C | $A$ | $C C C-$ | $A A A A-$ | CC- | $G$ | AAAA | $C C$ |
| read_2 | TT | C | $A$ | $C C C C$ | $A A A A-$ | $C C C$ | $G$ | $A A A A$ | $C C$ |
| read_3 | TT | C | $A$ | CCC- | AAAAA | CCC | $G$ | $A A A A$ | $C C$ |
|  | $r_{1}$ | $r_{2}$ |  | $r_{4}$ | $r_{5}$ | $r_{6}$ | $r_{7}$ | $r_{8}$ | $r_{9}$ |
| c_est | $G$ | C | TT | $C \quad A$ | $A-C$ | $G$ | A |  |  |
| read_1 | $G$ | C | TT | $C \quad A$ | $A-C$ | $G$ | A |  |  |
| read_2 | $G$ | C | TT | $C \quad A$ | $G \quad C$ | $G$ | A |  |  |
| read_3 | $G$ | C | TT | $C \quad A$ | $G \quad C$ | $G$ | A |  |  |
|  | $r_{10}$ | $r_{11}$ | $r_{12}$ | $r_{13} r_{1}$ | $\begin{array}{lll}14 & r_{15} & r_{16}\end{array}$ | $r_{17}$ | $r_{18}$ |  |  |

We next divide the alignment into 18 nonoverlapping segments corresponding to different maximal length homopolymers. The segments may have different lengths, but due to the balancing constraint, no segment has length exceeding 8 . To improve the current estimate Cest, we start by forming the consensus from the left and by adding a new homopolymer at each step. Since for the first three segments all four sequences agree on the homopolymer lengths and do not violate the balancing property, we initiate the new consensus sequence to
cns TTCA.
Next, we add one more homopolymer corresponding to the forth segment $\left(r_{4}\right)$. According to the four sequences, this homopolymer should have length either 3 or 4 (the sequences in question are CCC or CCCC). Note that the majority rule suggests that the correct sequence is CCC; this sequence also satisfies the balancing property, since we know that the next symbol is from the fifth segment and equals A . The second option CCCC does not satisfy the balancing property. Hence, the only valid solution up until this point reads as
cns TTCACCC.
For the fifth segment, the majority rule suggests picking the sequence AAAA as the next homopolymer candidate. Also, it is apparent that we need to have at least four G or C symbols in both the segments $r_{6}$ and $r_{7}$, as otherwise the resulting block does not have a balanced GC-content. The only homopolymer choices that satisfy these constraints are CCC for $r_{6}$ and G for $r_{7}$. As all sequences agree on the homopolymer choices for segments $r_{8}$ to $r_{12}$, the 24 symbols of the consensus read as TTCACCCAAAACCC-

GAAAACCGCT. As may be seen, the last 8 symbols do not have a balanced GC composition. As the first encountered ambiguity in the reads came from segment $r_{5}$, we change our homopolymer to AAAAA instead of AAAA, although it violates the majority choice. Then, the consensus up until and including segment 14 equals:

## cns TTCACCCAAAAACCCGAAAACCGCTTCA.

(Note that violations of the majority rules as described above are rare, and included only for illustrative purposes.) In the final step, we select the homopolymer G to represent the segment $r_{15}$, although it again violates the majority rule - another choice would result in a consensus with 31 symbols, a length not divisible by 8 . Hence, the new consensus equals

## cns TTCACCCAAAAACCCGAAAACCGCTTCAGCGA

and satisfies the GC-balanced property. This new consensus is used in the BWA reference based alignment software to identify more good and acceptable quality reads that may resolve issues with poor alignment regions. The procedure is repeated until no sufficiently good reads are available, or until all poor alignment regions are resolved or declared unresolvable.

Clearly, one cannot guarantee that the above procedure produces errorfree read estimates. But given the choice of the alignment parameters, the majority rules and the balancing checks, one tends to observe only a very small number of errors which tend to be exclusively deletions. In our experiments, we encountered three deletion errors for the whole encoded file: All deletions were confined to homopolymers of length at least five, and exclusively included As. Hence, the information about homopolymer symbols was recovered correctly. Table 4.5 (Experiment 1), shows the summary of error events in the DNA blocks after one round of address-anchored alignment and an additional round of BWA alignment.

We conclude by observing that in all alignment procedures, we used less than 200 reads per DNA codeword. Such a small number of reads may be generated in a relatively short time, and it appears to be a rule of thumb that the best quality reads are generated first. Hence, the readout cost and delay of the MinION system are highly competitive with those of other technologies.

We also tested the Nanopolish software [70] (Version 0.6-dev) to obtain

Table 4.5: Summary of errors on the consensus sequences on each experiment. Experiment 1 represents our iterative alignment method that identifies the consensus sequences using multiple sequence alignment techniques, majority rules and the balancing property. Experiments 2,3 use Nanopolish software to find the consensus sequences.

| Block | Number of errors (substitution, insertion, deletion) |  |  |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: |
|  | Experiment 1 |  | Experiment 2 |  | Experiment 3 |
|  | Round 1 | Round 2 | Round 1 | Round 2 |  |
| 1 | $(0,0,9)$ | $(0,0,2)$ | $(14,36,8)$ | $(14,32,5)$ | $(0,4,1)$ |
| 2 | $(0,0,7)$ | $(0,0,0)$ | $(83,93,58)$ | $(75,99,40)$ | $(3,10,7)$ |
| 3 | $(0,0,3)$ | $(0,0,0)$ | $(11,48,1)$ | $(10,45,0)$ | $(0,10,0)$ |
| 4 | $(0,0,1)$ | $(0,0,0)$ | $(4,57,2)$ | $(1,54,1)$ | $(0,2,0)$ |
| 5 | $(0,0,0)$ | $(0,0,0)$ | $(8,52,10)$ | $(4,45,3)$ | $(0,1,0)$ |
| 6 | $(0,0,1)$ | $(0,0,0)$ | $(397,100,54)$ | $(390,102,61)$ | $(0,2,0)$ |
| 7 | $(0,0,2)$ | $(0,0,0)$ | $(8,34,4)$ | $(3,31,3)$ | $(0,0,0)$ |
| 8 | $(0,0,0)$ | $(0,0,0)$ | $(22,53,2)$ | $(18,51,1)$ | $(1,2,0)$ |
| 9 | $(0,0,2)$ | $(0,0,0)$ | $(388,107,55)$ | $(404,92,54)$ | $(0,0,0)$ |
| 10 | $(0,0,0)$ | $(0,0,0)$ | $(398,89,55)$ | $(388,100,59)$ | $(0,1,0)$ |
| 11 | $(0,0,0)$ | $(0,0,0)$ | $(21,36,20)$ | $(16,23,18)$ | $(0,1,0)$ |
| 12 | $(0,0,1)$ | $(0,0,0)$ | $(21,72,3)$ | $(14,59,1)$ | $(0,1,0)$ |
| 13 | $(0,0,1)$ | $(0,0,0)$ | $(19,60,10)$ | $(7,5,4)$ | $(0,1,0)$ |
| 14 | $(0,0,1)$ | $(0,0,0)$ | $(21,30,9)$ | $(15,34,3)$ | $(0,2,0)$ |
| 15 | $(0,0,0)$ | $(0,0,0)$ | $(17,50,3)$ | $(15,46,2)$ | $(0,2,2)$ |
| 16 | $(0,0,4)$ | $(0,0,1)$ | $(21,67,12)$ | $(15,62,4)$ | $(0,1,0)$ |
| 17 | $(0,0,0)$ | $(0,0,0)$ | $(362,96,42)$ | $(359,95,44)$ | $(0,1,0)$ |

the consensus sequences. Nanopolish is a software package for signal-level analysis of Oxford Nanopore sequencing data, and it is used to calculate an improved consensus sequence from an initial draft genome. The commands used to obtain the consensus are as follow:

1. Extract the QC-passed reads from a directory of FAST5 files:
\$ nanopolish extract --type 2d FAST5/ > reads.fa
2. Index the draft genome:
\$ bwa index draft.fa
3. Align the basecalled reads to the draft sequence:
\$ bwa mem -x ont2d -t 8 draft.fa reads.fa | samtools sort -o reads.sorted.bam -T reads.tmp - samtools index reads.sorted.bam
4. Use Nanopolish to compute the consensus sequence:
\$ nanopolish variants -consensus cns.fa -r reads.fa - b reads.sorted.bam -g draft.fa -t 4 -- min-candidate frequency 0.1

To test the performance of Nanopolish, we designed two separate experiments; see Table 4.5, Experiment 2 and Experiment 3.

- The first experiment includes two rounds, and follows the same procedure we used during our iterative alignment method. For the first round, we used the 17 known address sequences as pilot sequences and selected 17 DNA blocks from the reads file (reads.fa) to form the draft genomes (draft.fa). We used the draft genome and flowed the Nanopolish workflow to obtain 17 consensus sequences (cns.fa). Next, each consensus sequence was compared to its original DNA block, and the difference were recorded in terms of the number of insertion, deletion, and substitution; see Table 4.5, Experiment 2/Round 1.

The second round repeats the same procedure, with the draft genome being the consensus sequence from the previous round. The result on Table 4.5, Experiment 2/Round 2 suggests that running the Nanopolish software multiple times does not necessarily improve the consensus sequence accuracy.

- In the second experiment, we used the original 17 DNA blocks as the draft genomes and formed the draft.fa. The results on Table 4.5, Experiment 3 suggest that for our data set Nanopolish is unable to reconstruct the original DNA blocks even in this genie-aided case.


### 4.5.7 Deletion Correction

There are numerous known coding schemes that can be used to correct standard deletion errors. The principle drawback of all these schemes is their low information rate and inability to tackle one particular subclass of deletion errors that was encountered in our sequencing experiments.

In what follows, we briefly discuss three prominent techniques from the literature, and review their information rates.

- Watermark coding and variants thereof refer to the process of inserting or super-imposing a known sequence over an information sequence. Typically, the information sequence is then encoded with an outer probabilistic code such as an LDPC (low-density parity-check) code that enables recovery of the encoded message. The decoding proceeds in two stages whereby the first stage makes use of the watermark code to compute likelihoods that are used in the second stage to decode the LDPC code. The watermark codes are typically designed to correct low error rates and these codes have small information rates. For instance, the highest rate code from [71] was 0.71 and this code had a block error rate which exceeded 0.1 , even for a raw deletion rate as small as 0.005 . A code with a similar information rate was presented in [72], where again the code had a block error rate exceeding 0.1 when the raw deletion rate was below 0.1. Another potential drawback to these codes is their long block length (on the order to several thousand bits long) which may be unsuitable for DNA codewords that are typically of length of the order of $1,000 \mathrm{bp}$.
- A more recent scheme for correcting deletions was introduced in [73]. The idea is to use a series of error-control codes on patterns that appear in a large majority of the strings. To correct $t$ deletions, in a block of length ( $n$ ) the construction from [73] was shown to require
$O\left(t^{2}(\log t) \log n\right)$ redundant bits; however, for shorter block lengths, the construction requires many redundant bits. For instance, for the case where $t=2$, it can be shown that this construction requires at least $144(n)$ bits of redundancy which renders the approach impractical for shorter block lengths.
- In [74], a number-theoretic code construction was proposed, which extended the congruence for single deletion codes introduced by Levenshtein in [19]. This construction requires many redundant bits, especially for longer block lengths. For only $t=2$, the redundancy (or the number of redundant bits) is linear with respect to the block length $n$. More specifically, it can be shown that the redundancy (in bits) for the case where $t=2$ is at least $\left\lfloor\frac{1.5^{n}}{\sqrt{5}}\right\rfloor$ which implies that for $n=1024$, such a construction would require at least 597 bits of redundancy.


### 4.5.8 Homopolymer Parity-check Coding

Motivated by the findings of the iterative anchored alignment phase, we propose a simple deletion correcting code related to the method described in [65] which leverages the so-called "integer sequence" of an information sequence.

To correct $t$ deletions that preserve homopolymer symbols, our encoding scheme requires approximately $t N$ bits of redundancy, where $N$ denotes the length of the encoded information sequence. Furthermore, the code partitions the space $\mathbb{F}_{4}^{N}$ and hence lends itself to systematic encoding, which is desirable when constructing codes compatible with different nanopore sequencers. We also note that the proposed construction outperforms existing general multiple deletion correcting codes such as those described in [73, 74] as it is adapted to the nanopore channel at hand.

We begin by reviewing the notion of an integer sequence. For a vector $x \in \mathbb{F}_{4}^{n}$, the integer sequence of $x$ is the sequence of lengths of maximal runs in $x$. For example, the integer sequence of $x=(0,0,1,3,3,2,1,1)$ is

$$
I(x)=(2,1,2,1,2) .
$$

Similarly, the integer sequence of AATTTGCGAA equals ( $2,3,1,1,1,2$ ). In the context of DNA coding, we refer to such a sequence as the homopolymer
length sequence. We also make use of the string sequence of a codeword. The string sequence represents the symbols in the maximal runs of $x$. For example, the string sequence of $x$ equals

$$
S(x)=(0,1,3,2,1)
$$

since the first run in $x$ has value 0 , the next run has value 1 and so on. The string sequence of AATTTGCGAA equals (A,T,G,C,G,A).

It is straightforward to see that one can uniquely recover $x$ given its integer sequence and string sequence. For shorthand, we use $M(I(x), S(x))$ to denote the "reconstruction map", a map such that

$$
M(I(x), S(x))=x
$$

We introduce one more relevant piece of notation. Suppose that $z \in \mathbb{F}_{2}^{N}$, i.e., that $z$ is a binary vector of length $N$. Then, let

$$
\mathfrak{B}_{t}(z)=\{z+e\},
$$

where $e \in\{0,-1\}^{N}$ and $e$ has at most $t$ non-zero components. Given $I, S$, and $\mathfrak{B}_{t}$ we are now able to define the types of errors we wish to correct, which we refer to as sticky deletions, a special case of the general family of repetition errors [75].

Let $x \in \mathbb{F}_{4}^{N}$. In our model, we assume that $y \in \mathbb{F}_{4}^{N-s}$ (where $s \leq t$ ) is such that

1. $S(y)=S(x)$, and
2. $I(y) \in \mathfrak{B}_{t}(I(x))$.

Note that the first condition enforces that deletions occurring in $x$ leading to $y$ can never cause runs to be added or removed. In other words, the deletions are not allowed to change the string sequence. The second restriction enforces that deletions occurring in $x$ can cause each run length in $x$ to decrease by at most one.

We define next a code $\mathcal{C}(N, t)$ capable of correcting sticky deletions. For $x \in \mathbb{F}_{4}^{N}$, let $|x|$ denote the number of runs in $x$. We assume in what follows that for $|x|=r$, where $r<N$, the output of $I(x)$ has length $N$ with the last $N-r$ components of $I(x)$ set to zero.

Suppose that $\mathcal{C}_{H}(N, t)$ is a binary code of length $N$ capable of correcting up to $t$ substitution errors. Let $\mathcal{C}(N, t) \subseteq \mathbb{F}_{4}^{N}$ be defined so that
$\mathcal{C}(N, t)=\left\{x \in \mathbb{F}_{4}^{N}: I(x) \bmod 2 \in \mathcal{C}_{H}(N, t)\right\}$.
Let $\mathcal{D}_{t}$ denote a decoder for $\mathcal{C}_{H}(N, t)$ where if $y=z+e^{(2)} \in \mathbb{F}_{2}^{N}, z \in$ $\mathcal{C}_{H}(N, t)$, and $e^{(2)} \in \mathbb{F}_{2}^{N}$ has at most $t$ non-zero components, then $\mathcal{D}_{t}(y)=z$. We prove next that the code $\mathcal{C}(N, t)$ can correct up to $t$ sticky deletions.

Theorem 2. The code $\mathcal{C}(N, t) \subseteq \mathbb{F}_{4}^{N}$ can correct up to $t$ sticky deletions.
Proof: We prove the result by outlining the decoding algorithm. Suppose that the vector $x \in \mathcal{C}(N, t)$ is stored and that the vector $y \in \mathbb{F}_{4}^{N-s}$ is retrieved, where $s \leq t$ is the result of $s$ sticky deletions.

First, we compute $S=S(y)$. Since $y$ is the result of at most $t$ sticky deletions, we know that $S=S(x)$. We now show how to recover $I(x)$. Since $y$ is the result of at most $t$ sticky deletions occurring to $x, I(y) \in \mathfrak{B}_{t}(I(x))$, so that we may write $I(y)=I(x)+e$, where $e$ has at most $t$ non-zero components and each component has value -1 . Notice that

$$
I(y) \bmod 2 \equiv(I(x)+e) \bmod 2 \equiv I(x) \bmod 2+e \bmod 2,
$$

where $I(x) \bmod 2 \in \mathcal{C}_{H}(N, t)$ and $e$ is a binary vector with at most $t$ non-zero components. Therefore, applying $\mathcal{D}_{t}$ to $I(y) \bmod 2$ gives
$\mathcal{D}_{t}(I(y) \bmod 2)=\mathcal{D}_{t}(I(x) \bmod 2+e \bmod 2)=I(x) \bmod 2$.
From the previous equation, given $I(x) \bmod 2$, we can determine $e \bmod 2$. Notice that every non-zero component at position $i$ in $e \bmod 2$ is the result of a sticky deletion. Therefore, we increment the value of $I(y)$ at position $i$ by one to obtain $I(x)$. Using the map $M$ we may then reconstruct $x=M(I(x), S)$.

Note that the construction in Theorem 1 is not systematic. We describe next how to use the scheme to perform systematic encoding. We consider the case where $t=2$ and use a primitive BCH code over $\mathbb{F}_{2}$ of length 1023 and dimension 1003, shortened on the last 23 bits. The resulting code has length 1000, dimension 980, and can correct up to 2 random substitution errors. We denote this code by $C(1000,980,2)$. Since $C(1000,980,2)$ is a linear code, there exists a systematic encoder for $\mathcal{C}_{H}$, which we denote by Enc, that given an input $w \in \mathbb{F}_{2}^{980}$, outputs 20 parity bits such that $(w, \operatorname{Enc}(w)) \in C(1000,980,2)$.
We encode our information sequence, denoted by $M_{4} \in \mathbb{F}_{4}^{980}$ into a codeword $x \in \mathbb{F}_{4}^{1000}$ according to the following procedure:

1. We set the first 980 symbols of $x$ to be equal to $M_{4}$ and let $=\operatorname{Enc}\left(I\left(M_{4}\right)\right)$ $\in \mathbb{F}_{2}^{20}$.
2. We convert $z$ to a quaternary representation and denote the resulting sequence with $z^{(4)} \in \mathbb{F}_{4}^{10}$.
3. We set $\left(x_{981}, x_{982}, x_{983}, \ldots, x_{1000}\right)=\left(z_{1}^{(4)}, z_{1}^{(4)}, z_{2}^{(4)}, z_{2}^{(4)}, \ldots, z_{10}^{(4)}, z_{10}^{(4)}\right)$.

Note that since $z$ is binary, it follows that 10 symbols over $\mathbb{F}_{4}$ suffice to store $z$. Observe also that the last 20 symbols in $x$ arise by simply repeating every symbol in $z^{(4)}$ twice. Hence, it is straightforward to prove the following corollary.

Corollary 1. Suppose $x$ is encoded according to the previous procedure and $y$ is the result of up to 2 sticky deletions in $x$. Then, it is possible to recover $x$ from $y$.

Proof: Let $v$ equal the last 20 symbols in $y$ read in reverse order. In other words, the first symbol of $v$ equals the last symbol in $y$, the second symbol in $v$ equals the second to last symbol in $y$, and so on. Let $z_{R}$ be equal to the last 20 symbols in $x$ (which results from repeating every symbol in $z^{(4)}$, generated by our encoding procedure) read in reverse order. We show that it is possible to recover $z_{R}$ from $v$ given that at most 2 sticky deletions occurred in the string. In fact, it can be shown that $\left(z_{1}^{(4)}, z_{1}^{(4)}, z_{2}^{(4)}, z_{2}^{(4)}, \ldots, z_{10}^{(4)}, z_{10}^{(4)}\right)$ can be recovered given any number of sticky deletions. Note that if $z_{R}$ is known, one can easily recover the parity bits $z$ and combine the parity bits (which have no errors) with the information symbols in $x$ to construct a vector $\widehat{y} \in \mathfrak{B}_{2}(x)$, where $\widehat{y}$ has at most 2 sticky deletions in the information symbols and $x \in \mathcal{C}(N, t)$.

Consider the sequences $I(v)=\left(u_{1}, u_{2}, \ldots, u_{|v|}\right), \mathrm{I}\left(z_{R}\right)=\left(s_{1}, s_{2}, \ldots, s_{\left|z_{R}\right|}\right)$, and $S(v)$. As a consequence of the sticky channel definition, $S(v)=S\left(z_{R}\right)$. Note also that for every symbol $u_{i} \in I(v)$, we have $u_{i} \in\left\{s_{i}, s_{i}-1\right\}$. As a result of the encoding, $s_{i} \equiv 0 \bmod 2$. Therefore, we can recover $s_{i}$ from $u_{i}$ by setting $u_{i}=u_{i}+1$ if $u_{i} \equiv 1 \bmod 2$ and setting $u_{i}=u_{i}$ otherwise. In this manner, we can recover $I\left(z_{R}\right)$ and determine $z_{R}$ from $M\left(I\left(z_{R}\right), S\left(z_{R}\right)\right)$.

The rate of the homopolymer check codes was 0.98 , and it allowed for systematic encoding, which cannot be achieved via simple $(d=1, k=6)$ run-length-constrained code of slightly higher rate 0.998 . Furthermore, the homopolymer parity checks may be stored on classical media which have neg-
ligible small error probabilities to further increase the code rate to 0.98 . This comes at almost no additional cost, as the homopolymer checks constitute less than $0.02 \%$ of the data volume.

Remark: Note that by using the integer and symbol sequences one may correct a more general class of sticky errors, including both deletions and insertions. In this context, it suffices to use a classical substitution errorcorrecting code on the integer sequence. Any asymmetries observed in the deletion or insertion patterns can be handled by using modulo constraints akin to those described in the definition of the code $\mathcal{C}(N, t)$. As asymmetries provide additional side information, one may expect a higher coding rate for modulo constrained codes compared to standard substitution error-correcting codes.

### 4.5.9 Information Rate and Data Storage Density

The information rate of a code (also known as the code rate) is the proportion of the stored data that represents useful information (i.e., is non-redundant). That is, if the code rate equals $k / n$, for every $k$ bits of useful information, the code generates a total of $n$ bits of data, of which $n-k$ are redundant. In our data storage system, we have three layers of data encoding. We calculate next the information rate for each layer and multiply the results of the three layers to get the overall information rate.

- In the first stage of encoding, we used GC-balanced DNA fragments of length 8 to store data. Note that the total number of GC-balanced DNA string of length 8 is $\left(\frac{8}{4}\right) \times 2^{8}$. Hence, the information rate of the balancing code equals

$$
\begin{equation*}
R_{1}=\frac{\left[\left(\frac{8}{4}\right) \times 2^{8}\right]}{4^{8}}=0.8831 \tag{4.5}
\end{equation*}
$$

Note that the nominator is the logarithm of the number of bits required to represent each GC-balanced DNA string of length 8 .

- In the second stage of encoding, we formed each DNA block of length $1,000 \mathrm{bp}$ by concatenating an address sequence with the balanced blocks.

At this stage, only 984 bp are used to store useful data while the remaining 16 bp are reserved for addressing. Hence, the information rate of this stage equals

$$
\begin{equation*}
R_{2}=\frac{984}{1,000}=0.984 \tag{4.6}
\end{equation*}
$$

- The third stage of encoding is homopolymer parity-check coding. Here, the code rate equals

$$
\begin{equation*}
R_{3}=0.98 \tag{4.7}
\end{equation*}
$$

The total information rate is obtained by multiplying the three rates from 5,6 and 7 , and equals

$$
\begin{equation*}
R=\text { Information rate }=0.8831 \times 0.984 \times 0.98 \approx 0.85 \tag{4.8}
\end{equation*}
$$

To calculate the DNA information density of our scheme, we use the fact that average weight of a DNA base pair is 650 dalton ( 1 dalton equals the mass of a single hydrogen atom, or $1.67 \times 10^{-24}$ gram [76]). As we mapped 3,633 bytes of compressed data into 17 DNA blocks with total number of $16,880 \mathrm{bp}$, the DNA information density equals

$$
\begin{align*}
\text { DNA information density } & =\frac{3,633 \text { bytes } \times 0.85}{(16,880 \mathrm{bp}) \times\left(1.67 \times 10^{-24} \mathrm{gram} / \mathrm{bp}\right)} \\
& \approx 1.1 \times 10^{23} \frac{\text { bytes }}{\text { gram }} \tag{4.9}
\end{align*}
$$

These are the highest known achievable information rate and density of all DNA-based data storage systems, even when taking into account systems that do not use addressing or rely on highly accurate, but large volume Illumina sequencers.

Data availability. The sequencing data are available at Google Drive:
https://drive.google.com/open?id=0BwIM8p8qEKCaU1NIRzFWTjltZ2M
Software availability. The encoding, alignment and decoding algorithms are available at GitHub:
https://github.com/smhty/MATLAB_MinION

## BIBLIOGRAPHY

[1] N. C. Seeman, "An overview of structural dna nanotechnology," Molecular Biotechnology, vol. 37, no. 3, p. 246, 2007.
[2] L. Orlando, A. Ginolhac, G. Zhang, D. Froese, A. Albrechtsen, M. Stiller, M. Schubert, E. Cappellini, B. Petersen, I. Moltke et al., "Recalibrating equus evolution using the genome sequence of an early middle pleistocene horse," Nature, vol. 499, no. 7456, pp. 74-78, 2013.
[3] J. Shendure and E. L. Aiden, "The expanding scope of dna sequencing," Nature Biotechnology, vol. 30, no. 11, pp. 1084-1094, 2012.
[4] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in dna," Science, vol. 337, no. 6102, pp. 1628-1628, 2012.
[5] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, lowmaintenance information storage in synthesized dna," Nature, 2013.
[6] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on dna in silica with error-correcting codes," Angewandte Chemie International Edition, vol. 54, no. 8, pp. 2552-2555, 2015.
[7] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," Journal of the Society for Industrial and Applied Mathematics, vol. 8, no. 2, pp. 300-304, 1960.
[8] E. Gilbert, "Synchronization of binary messages," Information Theory, IRE Transactions on, vol. 6, no. 4, pp. 470-477, 1960.
[9] M. P. Schützenberger, "On the synchronizing properties of certain prefix codes," Information and Control, vol. 7, no. 1, pp. 23-36, 1964.
[10] H. Morita, A. J. van Wijngaarden, and A. Han Vinck, "On the construction of maximal prefix-synchronized codes," Information Theory, IEEE Transactions on, vol. 42, no. 6, pp. 2158-2166, 1996.
[11] G. D. Cohen and S. Litsyn, "Dc-constrained error-correcting codes with small running digital sum," Information Theory, IEEE Transactions on, vol. 37, no. 3, pp. 949-955, 1991.
[12] M. Blaum, S. Litsyn, V. Buskens, and H. C. van Tilborg, "Errorcorrecting codes with bounded running digital sum," IEEE Transactions on Information Theory, vol. 39, no. 1, pp. 216-227, 1993.
[13] L. J. Guibas and A. M. Odlyzko, "Maximal prefix-synchronized codes," SIAM Journal on Applied Mathematics, vol. 35, no. 2, pp. 401-418, 1978.
[14] O. Milenkovic and N. Kashyap, "On the design of codes for dna computing," in Coding and Cryptography. Springer, 2006, pp. 100-119.
[15] J.-M. Rouillard, M. Zuker, and E. Gulari, "Oligoarray 2.0: design of oligonucleotide probes for dna microarrays using a thermodynamic approach," Nucleic Acids Research, vol. 31, no. 12, pp. 3057-3062, 2003.
[16] S. Yazdi, H. M. Kiah, R. Gabrys, and O. Milenkovic, "Mutually uncorrelated primers for dna-based data storage," arXiv preprint arXiv:1709.05214, 2017.
[17] S. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access dna-based storage system," Scientific Reports, vol. 5, no. 14138, 2015.
[18] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free dna-based data storage," Scientific Reports, vol. 7, no. 1, p. 5011, 2017.
[19] V. Levenshtein, "Decoding automata, invariant with respect to the initial state," Problemy Kibernet, vol. 12, pp. 125-136, 1964.
[20] A. J. De Lind Van Wijngaarden and T. J. Willink, "Frame synchronization using distributed sequences," Communications, IEEE Transactions on, vol. 48, no. 12, pp. 2127-2138, 2000.
[21] D. Bajić and J. Stojanović, "Distributed sequences and search process," in Communications, 2004 IEEE International Conference on, vol. 1. IEEE, 2004, pp. 514-518.
[22] S. Bilotta, E. Pergola, and R. Pinzani, "A new approach to cross-bifixfree sets," IEEE Transactions on Information Theory, vol. 6, no. 58, pp. 4058-4063, 2012.
[23] S. R. Blackburn, "Non-overlapping codes," IEEE Transactions on Information Theory, vol. 61, no. 9, pp. 4890-4894, 2015.
[24] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for dna sequence profiles," IEEE Transactions on Information Theory, vol. 62, no. 6, pp. 3125-3146, 2016.
[25] R. Gabrys, E. Yaakobi, F. Farnoud, F. Sala, J. Bruck, and L. Dolecek, "Codes correcting erasures and deletions for rank modulation," IEEE Transactions on Information Theory, vol. 62, no. 1, pp. 136-150, 2016.
[26] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric lee distance codes for dna-based storage," IEEE Transactions on Information Theory, 2017.
[27] 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.
[28] P. Yakovchuk, E. Protozanova, and M. D. Frank-Kamenetskii, "Basestacking and base-pairing contributions into thermal stability of the dna double helix," Nucleic Acids Research, vol. 34, no. 2, pp. 564-574, 2006.
[29] P. M. Vallone, J. M. Butler et al., "Autodimer: a screening tool for primer-dimer and hairpin structures," Biotechniques, vol. 37, no. 2, pp. 226-231, 2004.
[30] K. A. S. Immink, Codes for Mass Data Storage Systems. Shannon Foundation Publisher, 2004.
[31] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for dna storage," in Information Theory (ISIT), 2017 IEEE International Symposium on. IEEE, 2017, pp. 3115-3119.
[32] O. Milenkovic and N. Kashyap, "Dna codes that avoid secondary structures," in Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on. IEEE, 2005, pp. 288-292.
[33] N. de Bruijn, D. Knuth, and S. Rice, "The average height of planted plane trees," Graph Theory and Computing/Ed. RC Read, p. 15, 1972.
[34] E. N. Gilbert, "A comparison of signalling alphabets," Bell System Technical Journal, vol. 31, no. 3, pp. 504-522, 1952.
[35] R. Varshamov, "Estimate of the number of signals in error correcting codes," in Dokl. Akad. Nauk SSSR, vol. 117, no. 5, 1957, pp. 739-741.
[36] S. Tavares, "A study of synchronization techniques for binary cyclic codes," Ph.D. dissertation, Thesis (Ph. D.)-McGill University, 1968.
[37] L. Gyorfi, J. Massey et al., "Constructions of binary constant-weight cyclic codes and cyclically permutable codes," IEEE Transactions on Information Theory, vol. 38, no. 3, pp. 940-949, 1992.
[38] C. Bancroft, T. Bowler, B. Bloom, and C. T. Clelland, "Long-term storage of information in dna." Science (New York, NY), vol. 293, no. 5536, pp. 1763-1765, 2001.
[39] J. Davis, "Microvenus," Art Journal, vol. 55, no. 1, pp. 70-74, 1996.
[40] M. G. Ross, C. Russ, M. Costello, A. Hollinger, N. J. Lennon, R. Hegarty, C. Nusbaum, and D. B. Jaffe, "Characterizing and measuring bias in sequence data," Genome Biol, vol. 14, no. 5, p. R51, 2013.
[41] Integrated DNA Technologies, "gblocks $®$ gene fragments-related decoded articles." [Online]. Available: https://www.idtdna.com/ pages/decoded/decoded-articles/synthetic-biology/decoded/2017/04/ 25/tips-for-working-with-gblocks-gene-fragments
[42] A. V. Bryksin and I. Matsumura, "Overlap extension per cloning: a simple and reliable way to create recombinant plasmids," Biotechniques, vol. 48, no. 6, p. 463, 2010.
[43] S. C. Schuster, "Next-generation sequencing transforms today's biology," Nature Methods, vol. 5, no. 1, pp. 16-18, 2008.
[44] J. L. Massey, "Optimum frame synchronization," Communications, IEEE Transactions on, vol. 20, no. 2, pp. 115-119, 1972.
[45] D. Bajic, "On construction of cross-bifix-free kernel sets," 2nd MCM COST, 2007.
[46] Y. M. Chee, H. M. Kiah, P. Purkayastha, and C. Wang, "Cross-bifixfree codes within a constant factor of optimality," Information Theory, IEEE Transactions on, vol. 59, no. 7, pp. 4668-4674, 2013.
[47] P. Berman and M. Fürer, "Approximating maximum independent set in bounded degree graphs." in SODA, vol. 94, 1994, pp. 365-371.
[48] K. Goda and M. Kitsuregawa, "The history of storage systems," Proceedings of the IEEE, vol. 100, no. Special Centennial Issue, pp. 1433-1440, 2012.
[49] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A dna-based archival storage system," ACM SIGOPS Operating Systems Review, vol. 50, no. 2, pp. 637-649, 2016.
[50] V. Zhirnov, R. M. Zadegan, G. S. Sandhu, G. M. Church, and W. L. Hughes, "Nucleic acid memory," Nature Materials, vol. 15, no. 4, pp. 366-370, 2016.
[51] C. Laure, D. Karamessini, O. Milenkovic, L. Charles, and J.-F. Lutz, "Coding in 2d: Using intentional dispersity to enhance the information capacity of sequence-coded polymer barcodes," Angewandte Chemie, vol. 128, no. 36, pp. $10880-10883,2016$.
[52] Y. Erlich and D. Zielinski, "Capacity-approaching dna storage," bioRxiv, p. 074237, 2016.
[53] M. Blawat, K. Gaedke, I. Huetter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for dna data storage," Procedia Computer Science, vol. 80, pp. 1011-1022, 2016.
[54] T. Laver, J. Harrison, P. O'neill, K. Moore, A. Farbos, K. Paszkiewicz, and D. J. Studholme, "Assessing the performance of the oxford nanopore technologies minion," Biomolecular Detection and Quantification, vol. 3, pp. 1-8, 2015.
[55] J. Gray and C. Van Ingen, "Empirical measurements of disk failure rates and error rates," arXiv preprint cs/0701166, 2007.
[56] R. C. Edgar, "Muscle: multiple sequence alignment with high accuracy and high throughput," Nucleic Acids Research, vol. 32, no. 5, pp. 17921797, 2004.
[57] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," Nucleic Acids Research, vol. 22, no. 22, pp. 4673-4680, 1994.
[58] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows-wheeler transform," Bioinformatics, vol. 25, no. 14, pp. 17541760, 2009.
[59] G. K. Wallace, "The jpeg still picture compression standard," IEEE Transactions on Consumer Electronics, vol. 38, no. 1, pp. xviii-xxxiv, 1992.
[60] S. Josefsson, "The base16, base32, and base64 data encodings," Network Working Group RFC 4648, 2006.
[61] "Nanopore sequencing," 2017. [Online]. Available: https://en.wikipedia. org/wiki/Nanopore_sequencing
[62] H. Lu, F. Giordano, and Z. Ning, "Oxford nanopore minion sequencing and genome assembly," Genomics, Proteomics \& Bioinformatics, vol. 14, no. 5, pp. 265-279, 2016.
[63] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2004, pp. 910-918.
[64] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder, "Trace reconstruction with constant deletion probability and related results," in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2008, pp. 389-398.
[65] A. McGregor, E. Price, and S. Vorotnikova, "Trace reconstruction revisited," in European Symposium on Algorithms. Springer, 2014, pp. 689-700.
[66] L. Sok, P. Solé, and A. Tchamkerten, "Lattice based codes for insertion and deletion channels," in Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. IEEE, 2013, pp. 684-688.
[67] J. Duda, W. Szpankowski, and A. Grama, "Fundamental bounds and approaches to sequence reconstruction from nanopore sequencers," arXiv preprint arXiv:1601.02420, 2016.
[68] I. Milne, G. Stephen, M. Bayer, P. J. Cock, L. Pritchard, L. Cardle, P. D. Shaw, and D. Marshall, "Using tablet for visual exploration of secondgeneration sequencing data," Briefings in Bioinformatics, vol. 14, no. 2, pp. 193-202, 2012.
[69] "Matlab multialign function," 2016. [Online]. Available: https: //www.mathworks.com/help/bioinfo/ref/multialign
[70] "Nanopolish," 2017. [Online]. Available: https://github.com/jts/ nanopolish
[71] M. C. Davey and D. J. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," IEEE Transactions on Information Theory, vol. 47, no. 2, pp. 687-698, 2001.
[72] E. A. Ratzer, "Marker codes for channels with insertions and deletions," Annals of Telecommunications, vol. 60, no. 1, pp. 29-44, 2005.
[73] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," IEEE Transactions on Information Theory, 2017.
[74] A. S. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," IEEE Transactions on Information Theory, vol. 48, no. 1, pp. 305-308, 2002.
[75] L. Dolecek and V. Anantharam, "Repetition error correcting sets: Explicit constructions and prefixing methods," SIAM Journal on Discrete Mathematics, vol. 23, no. 4, pp. 2120-2146, 2010.
[76] "Molecular facts and figures," 2016. [Online]. Available: https://www.idtdna.com/pages/docs/educational-resources/ molecular-facts-and-figures.pdf

