

Protecting the Future of Information: LOCO Coding With Error Detection for DNA Data Storage

Canberk İrimağzı, Yusuf Usulan, and Ahmed Hareedy, *Member, IEEE*

Abstract—From the information-theoretic perspective, DNA strands serve as a storage medium for 4-ary data over the alphabet $\{A, T, G, C\}$. DNA data storage promises formidable information density, long-term durability, and ease of replicability. However, information in this intriguing storage technology might be corrupted because of error-prone data sequences as well as insertion, deletion, and substitution errors. Experiments have revealed that DNA sequences with long homopolymers and/or with low GC -content are notably more subject to errors upon storage. In order to address this biochemical challenge, constrained codes are proposed for usage in DNA data storage systems, and they are studied in the literature accordingly.

This paper investigates the utilization of the recently-introduced method for designing lexicographically-ordered constrained (LOCO) codes in DNA data storage to improve performance. LOCO codes offer capacity-achievability, low complexity, and ease of reconfigurability. This paper introduces novel constrained codes, namely DNA LOCO (D-LOCO) codes, over the alphabet $\{A, T, G, C\}$ with limited runs of identical symbols. Due to their ordered structure, these codes come with an encoding-decoding rule we derive, which provides simple and affordable encoding-decoding algorithms. In terms of storage overhead, the proposed encoding-decoding algorithms outperform those in the existing literature. Our algorithms are based on small-size adders, and therefore they are readily reconfigurable. D-LOCO codes are intrinsically balanced, which allows us to achieve balanced AT - and GC -content over the entire DNA strand with minimal rate penalty. Moreover, we propose four schemes to bridge consecutive codewords, three of which guarantee single substitution error detection per codeword. We examine the probability of undetecting errors over a presumed symmetric DNA storage channel subject to substitution errors only. We also show that D-LOCO codes are capacity-achieving and that they offer remarkably high rates even at moderate lengths.

Index Terms—Constrained codes, low-complexity algorithms, reconfigurable coding, LOCO codes, homopolymer run, balancing, error-detection, DNA data storage.

I. INTRODUCTION

In the current information age, DNA data storage is the next-generation technology offered to accommodate the needs of storing mass cold data [1]. Due to its remarkable advantage in terms of information density, durability, and ease of replicability over current commercial storage technologies, a pool of synthetic DNA is proposed as a potential medium to store data for archival purposes. Coding and data processing are essential for such emerging technology in order to prevent and

correct errors resulting from biochemical effects. To achieve high reliability of DNA strands for a long period of time, sequences with (i) limited runs of identical symbols and (ii) balanced percentage of A - T and G - C nucleotides are required to be generated for synthesis, i.e., for writing [1], [2].

Constrained codes are a class of nonlinear codes that eliminate a chosen set of forbidden patterns from codewords. The use of constrained codes forbidding error-prone patterns in accordance with channel requirements improves system performance, and thus they have a profusion of applications. Historically, in his seminal work of 1948 [3], Shannon showed how to represent an infinite sequence that forbids certain data patterns through a finite-state transition diagram (FSTD) and defined the capacity, i.e., the highest achievable rate. In 1970, Tang and Bahl [4] as well as Franaszek [5] introduced an important class of constrained codes called run-length-limited (RLL) codes. A (d, k) sequence, or an RLL sequence, is a binary sequence with the constraint that at least d and at most k zeros must separate consecutive ones (see Remark 10 in the Appendix for the relation between the binary version of D-LOCO codes and $(0, k)$ RLL codes). Since the works [4] and [5], advances in graph theory, numerical matrix theory, and symbolic dynamics remarkably benefited the analysis and design of constrained codes, especially those that are based on finite-state machines [6].

The data storage evolution has always been associated with advances in coding and signal processing. Here, we focus on the value constrained coding has been bringing to various data storage systems. In one-dimensional magnetic recording (MR) systems, constrained codes enabled remarkable density increases in early generations of MR devices that are based on peak detection [7], and they are still used to improve performance in MR systems adopting sequence detection [8]. By forbidding detrimental two-dimensional isolation patterns, constrained codes mitigate interference in modern two-dimensional MR systems [9]–[11]. As for solid-state storage systems, constrained codes are used to protect Flash memories from the effect of inter-cell interference resulting from charge propagation [12]–[15]. Constrained codes find application in other storage systems such as optical recording devices [16] and DNA data storage systems [17], [18], which are the topic of this paper. The spectral analysis of constrained codes characterizes desirable properties, such as balancing, in the system of interest [16], [19].

The design of constrained codes by adopting lexicographic indexing, also called enumerative coding, has started with the work of Tang and Bahl [4] as well as that of Cover [20], and it has been intermittently revisited in history [21]–[23]. Recently, Hareedy, Dabak, and Calderbank [24], following

This work was supported in part by the TÜBİTAK 2232-B International Fellowship for Early Stage Researchers.

Canberk İrimağzı is with the Institute of Applied Mathematics, Middle East Technical University (METU), 06800 Ankara, Turkey (e-mail: canberk.irimagzi@metu.edu.tr).

Yusuf Usulan and Ahmed Hareedy are with the Department of Electrical and Electronics Engineering, Middle East Technical University, 06800 Ankara, Turkey (e-mail: yusuf.usulan@metu.edu.tr; ahareedy@metu.edu.tr).

the works [25], [26] by Hareedy and Calderbank, introduced a novel technique in the field of constrained coding based on lexicographic indexing. They presented a general method for designing what are now called lexicographically-ordered constrained (LOCO) codes in a systematic manner for any finite set of forbidden patterns. They provided simple formulae, namely encoding-decoding rules, for the index of a codeword in terms of the codeword symbols and the cardinalities of LOCO codes having smaller lengths. In this paper, we follow this step-by-step outline to design constrained codes for DNA data storage with affordable encoding-decoding algorithms in accordance with channel requirements.

LOCO codes are constrained codes having codebooks equipped with lexicographic ordering, and thus they naturally come with unique encoding-decoding algorithms that are based on a rule executed by small-size adders. Therefore, LOCO codes do not require any look-up tables. D-LOCO codes, which are LOCO codes defined over the alphabet $\{A, T, G, C\}$ that eliminate long runs of identical symbols and achieve *AT*- and *GC*-content balance, are proposed here as capacity-achieving LOCO codes for DNA storage systems. Note that the “D” in “D-LOCO” is simply for “DNA”.

A. Some Related Works

Immink and Cai [27], [28] studied constrained coding schemes that address *GC*-content balance and homopolymer run-length constraints based on look-up tables. Song et al. [29] provided a high-rate coding scheme that produces DNA sequences with homopolymers of length at most 3 and whose *GC*-content is probabilistically proved (and empirically verified) to be close to 0.5 as a fraction. Moreover, they provided a low-complexity encoding-decoding algorithm that is also based on look-up tables. Improving the result in [29], Wang et al. [30] offered an efficient coding scheme, a product of which is a constrained code with normalized rate 0.9585 that eliminates homopolymers of length at least 4 and that has a guaranteed 40% – 60% *GC*-content based on look-up tables.¹

Using a sequence replacement technique, Nguyen et al. [31] proposed low-complexity encoders that convert binary sequences into 4-ary sequences of limited run-length ℓ and *GC*-content in $[0.5 - \epsilon, 0.5 + \epsilon]$, where ϵ is adaptable, which enable correcting a single insertion, single deletion, or single substitution error. For example, for $\ell = 4$ and 40% – 60% *GC*-content, such DNA sequences of length $n = 100$ symbols offer normalized code rate 0.91 with no error-correction. In [32], Nguyen et al. presented non-binary Varshamov-Tenengolts (VT) codes that are capable of correcting a single deletion or a single insertion with linear time encoding-decoding algorithms. For more error-correction coding schemes with efficient decoding algorithms for DNA data storage, see [33]–[35].

Park et al. [36] proposed an iterative decoding algorithm based on a mapping table for a constrained code that addresses the *GC*-content and the maximum homopolymer length requirements. Their method achieves normalized code

rate 0.9165 and 45% – 55% *GC*-content range, and their iterative encoding algorithm has a mapping table with 48 3-tuple 4-ary entries as a building block. Liu et al. [37] proposed a constrained coding scheme that achieves a *GC*-content in $[0.5 - \epsilon, 0.5 + \epsilon]$ and eliminates homopolymers of length larger than ℓ , and whose encoding-decoding algorithms have polynomial execution time and storage overhead. They also offered a coding scheme that satisfies a local *GC*-content constraint in order to further improve immunity against errors.

B. Our Contribution and Organization of the Paper

Our main result is designing the new D-LOCO codes for DNA data storage with low-complexity encoding-decoding algorithms and with desirable properties such as

- capacity-achievability,
- reconfigurability,
- low error propagation,
- parallelism,
- substitution error detection, and
- local *GC*-content balance.

In particular, we devise the general D-LOCO encoding-decoding rule, which is given in Theorem 2 in Subsection III-B, after we discuss an interesting special case in Theorem 1 in Subsection III-A. Theorem 2 provides a one-to-one mapping from an index set to the D-LOCO code, which is the encoding, and a one-to-one demapping from the D-LOCO code to the index set, which is the decoding. The encoding-decoding rule we obtain provides us with simple, low-complexity encoding-decoding algorithms (see Section V for algorithms). In fact, the storage overhead of our encoding-decoding algorithms turns out to be drastically low compared with those given in the literature (see Subsection VII-B). With this idea of encoding-decoding, one just needs a simple adder that converts an index (binary message) to a codeword and vice versa. This provides the pivotal advantage of reconfigurability of the code, which is as easy as reprogramming an adder via a set of multiplexers. This ease of reconfigurability allows one to adopt different coding schemes in accordance with the requirement of the DNA data storage system at different stages of its lifetime (see Subsection VII-C).

We also offer systematic approaches for balancing and bridging. Our approach for balancing incurs a minimal rate penalty that vanishes as the code length increases, which means it is capacity-achievable. This balancing approach guarantees not only global *GC*-content balance, but also local *GC*-content balance since it picks each codeword from two possible options according to the running *AT* – *GC* disparity. Furthermore, we suggest various bridging schemes that enable error detection, which enhances system reliability.

In Section II, we introduce D-LOCO codes and study their cardinality. In Section III, we derive a simple encoding-decoding rule for D-LOCO codes forbidding runs of length higher than 3 (Theorem 1) and then generalize this result to D-LOCO codes forbidding runs of length higher than any fixed value ℓ (Theorem 2). In Section IV, we propose four bridging schemes, three of which guarantee single substitution error detection. Assuming a $(1 - p, p/3, p/3, p/3)$ -symmetric DNA storage channel with substitution error rate p , a careful analysis

¹We adopt normalized rates throughout this paper, where the actual DNA code rates are divided by 2, since they immediately give the fraction of non-redundant information.

of the probability of no-detection in the occurrence of multiple substitution errors is presented. In Section V, we provide the encoding-decoding algorithms and discuss how to balance the DNA sequence in order to achieve close percentage of A - T and G - C nucleotides overall. In Section VI, we compare code rates at finite lengths when different bridging schemes are applied, and we show that D-LOCO codes are capacity-achieving. We conclude the paper with the complexity of encoding-decoding algorithms and other desirable properties of our proposed coding scheme.

II. DEFINITION AND CARDINALITY

In this section, we introduce D-LOCO codes and study their cardinality.

Definition 1. ([24, Definition 1]) For integers $\ell \geq 1$, the D-LOCO code $\mathcal{D}_{m,\ell}$ is defined as the set of all codewords of length m defined over the alphabet $\{A, T, G, C\}$ that do not contain any pattern in $\mathcal{F} = \{\Lambda^{\ell+1} \mid \Lambda \in \{A, T, C, G\}\}$. Here, $\Lambda^{\ell+1}$ is the sequence of length $\ell+1$ all of whose symbols are Λ , and such a sequence $\Lambda^{\ell+1}$ is simply called a run of length $\ell+1$.

Elements of $\mathcal{D}_{m,\ell}$ are also called \mathcal{F} -constrained sequences of length m , and they are ordered lexicographically.

Codewords in $\mathcal{D}_{m,\ell}$ are ordered in an ascending manner by following the rule $A < T < G < C$ for any symbol, and the symbol significance reduces from left to right. For more illustration, consider the two arbitrary codewords \mathbf{c} and \mathbf{c}' in $\mathcal{D}_{m,\ell}$. We say $\mathbf{c} < \mathbf{c}'$ if and only if for the first symbol position the two codewords differ at, \mathbf{c} has a ‘‘less’’ symbol than that of \mathbf{c}' . This is how we define lexicographic ordering.

Notation 1. The cardinality, i.e., codebook size, of the D-LOCO code $\mathcal{D}_{m,\ell}$ is denoted by $N_{\mathcal{D}}(m, \ell)$. However and for the ease of notation, we denote this cardinality by $N(m)$ whenever the context clarifies ℓ .

Example 1. $\mathcal{D}_{m,m-1}$ is the set of all non-constant sequences of length m over the alphabet $\{A, T, G, C\}$, and thus $N_{\mathcal{D}}(m, m-1) = 4^m - 4$; whereas $\mathcal{D}_{m,1}$ consists of all sequences of length m whose consecutive terms are distinct, which means $N_{\mathcal{D}}(m, 1) = 4 \cdot 3^{m-1}$. Throughout the paper, $\mathcal{D}_{m,3}$ will be of special interest to us based on the literature on the topic (see Subsection III-A) [29], [36].

The D-LOCO code $\mathcal{D}_{m,\ell}$ is partitioned into 4 groups, consisting of total 4ℓ subgroups, based on the set \mathcal{F} of forbidden patterns. In particular, for $1 \leq k \leq \ell$, we have the following subgroups:

Subgroup A(k): Codewords starting with $\mathbf{A}^k \Lambda$, where $\Lambda \in \{T, G, C\}$ from the left,

Subgroup T(k): Codewords starting with $\mathbf{T}^k \Lambda$, where $\Lambda \in \{A, G, C\}$ from the left,

Subgroup G(k): Codewords starting with $\mathbf{G}^k \Lambda$, where $\Lambda \in \{A, T, C\}$ from the left, and

Subgroup C(k): Codewords starting with $\mathbf{C}^k \Lambda$, where $\Lambda \in \{A, T, G\}$ from the left.

This partition of the D-LOCO code $\mathcal{D}_{m,\ell}$ is essential in both deriving its cardinality and the encoding-decoding rule. Let us first derive the cardinality $N(m)$ of $\mathcal{D}_{m,\ell}$ as a linear

combination of cardinalities of D-LOCO codes with lengths smaller than m using the group structure above.

We now introduce three notations and illustrate their relations to each other below.

Notation 2. For $1 \leq k \leq \ell$ and $\ell < m$, let

1. $N_{\Lambda_1, \Lambda_2, k}(m)$ denote the cardinality of the set of codewords in $\mathcal{D}_{m,\ell}$ starting with $\Lambda_1^k \Lambda_2$, for $\Lambda_2 \in \{A, T, G, C\} \setminus \{\Lambda_1\}$, and let
2. $N_{\Lambda_1, k}(m)$ denote the number of codewords $\mathbf{c} = c_{m-1}c_{m-2} \dots c_1c_0$ in $\mathcal{D}_{m,\ell}$ whose first k symbols from the left are all Λ_1 and $c_{m-k-1} \neq \Lambda_1$.

Note that we have

$$N_{\Lambda_1, k}(m) = \sum_{\Lambda_2 \in \{A, T, G, C\} \setminus \{\Lambda_1\}} N_{\Lambda_1, \Lambda_2, k}(m), \quad (1)$$

for $1 \leq k \leq \ell$. Moreover, let

3. $N_{\Lambda_1}(m)$ denote the number of codewords in $\mathcal{D}_{m,\ell}$ with $c_{m-1} = \Lambda_1$. Then, we have

$$N_{\Lambda_1}(m) = \sum_{k=1}^{\ell} N_{\Lambda_1, k}(m). \quad (2)$$

Proposition 1. ([27, Equation 1]) The cardinality $N(m)$ of the D-LOCO code $\mathcal{D}_{m,\ell}$, where $\ell \geq 1$, satisfies the following recursive relation for $m \geq \ell$:

$$N(m) = 3N(m-1) + 3N(m-2) + \dots + 3N(m-\ell). \quad (3)$$

For $0 \leq m \leq \ell$,

$$N(0) \triangleq \frac{4}{3}, \text{ and } N(m) = 4^m \text{ for } 1 \leq m \leq \ell.$$

Proof: We first consider the case of $m > \ell$. Combining (1) and (2) with the observation that

$$N_{\Lambda_1, \Lambda_2, k}(m) = N_{\Lambda_2}(m-k), \quad (4)$$

we obtain

$$\begin{aligned} N_A(m) &\stackrel{(2)}{=} \sum_{k=1}^{\ell} N_{A, k}(m) \\ &\stackrel{(1)}{=} \sum_{k=1}^{\ell} [N_{A, T, k}(m) + N_{A, G, k}(m) + N_{A, C, k}(m)] \\ &= \sum_{k=1}^{\ell} [N_T(m-k) + N_G(m-k) \\ &\quad + N_C(m-k)]. \end{aligned} \quad (5)$$

Thus, for all $\Lambda_1 \in \{A, T, G, C\}$, we have

$$N_{\Lambda_1}(m) = \sum_{k=1}^{\ell} \sum_{\Lambda_2 \in \{A, T, G, C\} \setminus \{\Lambda_1\}} N_{\Lambda_2}(m-k). \quad (6)$$

Adding these up over the 4-ary alphabet, we obtain

$$\begin{aligned} N(m) &= N_A(m) + N_T(m) + N_G(m) + N_C(m) \\ &= 3 \sum_{k=1}^{\ell} [N_A(m-k) + N_T(m-k) \\ &\quad + N_G(m, k) + N_C(m-k)] \end{aligned}$$

$$= 3 \sum_{k=1}^{\ell} N(m-k), \quad (7)$$

proving the relation (3) for $m > \ell$. For $1 \leq m \leq \ell$, $\mathcal{D}_{m,\ell}$ is the set of all codewords in $\{A, T, G, C\}$ of length m , and thus $N(m) = 4^m$. Moreover, the relation (3) holds in case $m = \ell$ by setting $N(0) = 4/3$ to complete the summation $N(\ell) = 3 \sum_{i=1}^{\ell-1} 4^{\ell-i} + 4 = 4^{\ell}$. ■

III. D-LOCO ENCODING-DECODING RULE

A. Encoding-Decoding Rule for $\ell = 3$

Now, we derive a formula that relates the lexicographic index of a D-LOCO codeword in the codebook to the codeword symbols. We call this formula the *encoding-decoding rule* of D-LOCO codes since it is the foundation of the D-LOCO encoding and decoding algorithms illustrated by Examples 2, 3, and 4 below.

Below, we study the encoding-decoding rule:

$$g : \mathcal{D}_{m,\ell} \rightarrow \{0, 1, \dots, N_{\text{D}}(m, \ell) - 1\}$$

that gives the index of any codeword in the codebook of $\mathcal{D}_{m,\ell}$ that is ordered lexicographically.

In order to find the *index* $g(\mathbf{c})$ of a codeword $\mathbf{c} = c_{m-1}c_{m-2} \dots c_1c_0$, we first study the contribution of each symbol c_i (for $0 \leq i \leq m-1$) to $g(\mathbf{c})$. This contribution is denoted by $g_i(c_i)$, and we have

$$g(\mathbf{c}) = \sum_{i=0}^{m-1} g_i(c_i). \quad (8)$$

In [24], $g_i(c_i)$ is formulated as follows:

$$g_i(c_i) = \sum_{c'_i < c_i} N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}c'_i), \quad (9)$$

where $N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}c'_i)$ is the number of all codewords in $\mathcal{D}_{m,\ell}$ that start with the sequence $\mathbf{s} = c_{m-1}c_{m-2} \dots c_{i+1}c'_i$ from the left. We will compute $N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}c'_i)$ by counting the codewords $\mathbf{d} = c'_id_{i-1} \dots d_1d_0$ in $\mathcal{D}_{i+1,\ell}$ such that $c_{m-1}c_{m-2} \dots c_{i+1}c'_id_{i-1} \dots d_1d_0$ is a codeword in $\mathcal{D}_{m,\ell}$. For ease of expression, $c_{m-1}c_{m-2} \dots c_{i+1}c'_id_{i-1} \dots d_1d_0$ will be called the *wedge* of \mathbf{s} and \mathbf{d} , and the operation itself will be called *wedging*.

Throughout the remaining part of this subsection, we set $\ell = 3$. Here, $N_{\Lambda}(m)$ denotes the number of all codewords in $\mathcal{D}_{m,3}$ whose first letter is Λ , and $N_{\Lambda,k}(m)$ (for $1 \leq k \leq 3$) denotes the number of all codewords $\mathbf{c} = c_{m-1}c_{m-2} \dots c_1c_0$ in $\mathcal{D}_{m,3}$ whose first k letters are all Λ and $c_{m-k-1} \neq \Lambda$. Note that using (5) and the intrinsic symmetry of the code ($N_{A,k}(m) = N_{T,k}(m) = N_{G,k}(m) = N_{C,k}(m)$), we have the relation

$$3N(m-k) = 4N_{\Lambda,k}(m), \quad \text{for } 1 \leq k \leq 3. \quad (10)$$

Remark 1. For Theorems 1–3, we define $c_i \triangleq \zeta$, for all $i > m-1$, to represent “out of codeword bounds” (see [24]).

Theorem 1. The encoding-decoding rule $g : \mathcal{D}_{m,3} \rightarrow \{0, 1, \dots, N(m) - 1\}$ is as follows:

$$g(\mathbf{c}) = \frac{3}{4} \sum_{i=0}^{m-1} \sum_{j=1}^3 \sum_{k=1}^j (\mathbf{a}_{i,k} + \mathbf{t}_{i,k} + \mathbf{g}_{i,k}) N(i+j-3), \quad (11)$$

where $N(0) \triangleq 4/3$, $N(-1) \triangleq 0$, and $N(-2) \triangleq 0$. Moreover, for all $\Pi > \Delta$ and for each $\Delta \in \{A, T, G\}$,

$$\begin{aligned} \delta_{i,1} &= 1 \text{ only if } c_i = \Pi \text{ and } c_{i+1} \neq \Delta, \\ \delta_{i,2} &= 1 \text{ only if } c_{i+1}c_i = \Delta\Pi \text{ and } c_{i+2} \neq \Delta, \\ \delta_{i,3} &= 1 \text{ only if } c_{i+2}c_{i+1}c_i = \Delta\Delta\Pi \text{ and } c_{i+3} \neq \Delta, \end{aligned} \quad (12)$$

and in all other cases, $\delta_{i,k} = 0$ for $i \in \{0, 1, \dots, m-1\}$ and $k \in \{1, 2, 3\}$.

Here, $\Pi > \Delta$ is according to the lexicographic ordering rule, and δ stands for the small letter in $\{\mathbf{a}, \mathbf{t}, \mathbf{g}\}$ corresponding to Δ in $\{A, T, G\}$.

Proof: We study the symbol contributions $g_i(c_i)$ for each symbol c_i in the alphabet $\{A, T, G, C\}$:

Case 1: If $c_i = A$, it is clear that $g_i(A) = 0$.

Case 2: If $c_i = T$, then

$$g_i(T) = N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}A). \quad (13)$$

Case 2(a): If $c_{i+1} \neq A$, then $\mathbf{s} = c_{m-1}c_{m-2} \dots c_{i+1}A$ can be wedged with any codeword in $\mathcal{D}_{i+1,3}$ which starts with A from the left. In other words and using (10),

$$\begin{aligned} N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}A) &= N_A(i+1) \\ &= N_{A,1}(i+1) + N_{A,2}(i+1) + N_{A,3}(i+1) \\ &= \frac{3}{4}[N(i) + N(i-1) + N(i-2)]. \end{aligned} \quad (14)$$

Case 2(b): If $c_{i+1} = A$ but $c_{i+2} \neq A$, then $\mathbf{s} = c_{m-1}c_{m-2} \dots c_{i+2}AA$ can be wedged with any codeword in $\mathcal{D}_{i+1,3}$ belonging to Subgroup A(1) or Subgroup A(2). Thus,

$$\begin{aligned} N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+2}AA) &= N_{A,1}(i+1) + N_{A,2}(i+1) \\ &= \frac{3}{4}[N(i) + N(i-1)]. \end{aligned} \quad (15)$$

Case 2(c): If $c_{i+1} = c_{i+2} = A$ but $c_{i+3} \neq A$, then $\mathbf{s} = c_{m-1}c_{m-2} \dots c_{i+3}AAA$ can be wedged with any codeword in $\mathcal{D}_{i+1,3}$ belonging to Subgroup A(1), i.e.,

$$\begin{aligned} N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+3}AAA) &= N_{A,1}(i+1) = \frac{3}{4}N(i). \end{aligned} \quad (16)$$

Since these subcases are all disjoint, we simply have

$$\begin{aligned} g_i(T) &= N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}A) \\ &= \frac{3}{4} \left[(\mathbf{a}_{i,1} + \mathbf{a}_{i,2} + \mathbf{a}_{i,3})N(i) \right. \\ &\quad \left. + (\mathbf{a}_{i,1} + \mathbf{a}_{i,2})N(i-1) + \mathbf{a}_{i,1}N(i-2) \right] \\ &= \frac{3}{4} \sum_{j=1}^3 \sum_{k=1}^j \mathbf{a}_{i,k} N(i+j-3). \end{aligned} \quad (17)$$

where $N(0) \triangleq 4/3$, $N(-1) \triangleq 0$, and $N(-2) \triangleq 0$. Moreover, $\mathbf{a}_{i,1} = 1$ only if $c_i = T$ and $c_{i+1} \neq A$, $\mathbf{a}_{i,2} = 1$ only if $c_{i+1}c_i = AT$ and $c_{i+2} \neq A$, $\mathbf{a}_{i,3} = 1$ only if $c_{i+2}c_{i+1}c_i = AAT$ and $c_{i+3} \neq A$, and

$a_{i,k} = 0$ otherwise.

Case 3: If $c_i = G$, then

$$g_i(G) = N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}A) + N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}T). \quad (18)$$

We study the term $N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}T)$ in various cases below.

Case 3(a): If $c_{i+1} \neq T$, then $\mathbf{s} = c_{m-1}c_{m-2} \dots c_{i+1}T$ can be wedged with any codeword in $\mathcal{D}_{i+1,3}$ which starts with T . In other words,

$$\begin{aligned} N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}T) &= N_T(i+1) \\ &= N_{T,1}(i+1) + N_{T,2}(i+1) + N_{T,3}(i+1) \\ &= \frac{3}{4}[N(i) + N(i-1) + N(i-2)]. \end{aligned} \quad (19)$$

Case 3(b): If $c_{i+1} = T$ but $c_{i+2} \neq T$, then $\mathbf{s} = c_{m-1}c_{m-2} \dots c_{i+2}TT$ can be wedged with any codeword in $\mathcal{D}_{i+1,3}$ belonging to Subgroup $T(1)$ or Subgroup $T(2)$. Thus,

$$\begin{aligned} N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+2}TT) &= N_{T,1}(i+1) + N_{T,2}(i+1) \\ &= \frac{3}{4}[N(i) + N(i-1)]. \end{aligned} \quad (20)$$

Case 3(c): If $c_{i+1} = c_{i+2} = T$ but $c_{i+3} \neq T$, then $\mathbf{s} = c_{m-1}c_{m-2} \dots c_{i+3}TTT$ can be wedged with any codeword in $\mathcal{D}_{i+1,3}$ belonging to Subgroup $T(1)$, i.e.,

$$\begin{aligned} N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+3}TTT) &= N_{T,1}(i+1) = \frac{3}{4}N(i). \end{aligned} \quad (21)$$

Since these subcases are all disjoint, we simply have the expression in (22), where $N(0) \triangleq 4/3$. Moreover,

$a_{i,1} = 1$ only if $c_i = G$ and $c_{i+1} \neq A$,
 $a_{i,2} = 1$ only if $c_{i+1}c_i = AG$ and $c_{i+2} \neq A$,
 $a_{i,3} = 1$ only if $c_{i+2}c_{i+1}c_i = AAG$ and $c_{i+3} \neq A$, and
 $a_{i,k} = 0$ otherwise; in addition,
 $t_{i,1} = 1$ only if $c_i = G$ and $c_{i+1} \neq T$,
 $t_{i,2} = 1$ only if $c_{i+1}c_i = TG$ and $c_{i+2} \neq T$,
 $t_{i,3} = 1$ only if $c_{i+2}c_{i+1}c_i = TTG$ and $c_{i+3} \neq T$, and
 $t_{i,k} = 0$ otherwise.

Case 4: One can similarly study the symbol contribution $g_i(c_i)$ for $c_i = C$ and find the number $N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}G)$ of codewords in $\mathcal{D}_{m,3}$ that start with $\mathbf{s} = c_{m-1}c_{m-2} \dots c_{i+1}G$ as well to obtain $g_i(C)$.

Adding all $g_i(c_i)$'s up, we obtain the encoding-decoding rule as given in the theorem statement. ■

Remark 2. Note that we have stated the encoding-decoding rule in Theorem 1 such that it aligns with its generalizations in the next section and in the Appendix. However, in our examples, we will use it in the form (23) where $\delta_{i,k}$'s are as in the statement of Theorem 1. Observe that (11) emerges directly from substituting (3) for $N(i+1)$ in (23).

Example 2. Consider the encoding-decoding rule

$$g : \mathcal{D}_{4,3} \rightarrow \{0, 1, \dots, 251\},$$

where $N(-2) \triangleq 0$, $N(-1) \triangleq 0$, $N(0) \triangleq 4/3$, $N(1) = 4$, $N(2) = 16$, $N(3) = 64$, and $N(4) = 4^4 - 4 = 252$. We discuss some instances and provide a detailed explanation for the codeword $ATGC$:

- For the codeword $\mathbf{c} = AAAT \in \mathcal{D}_{4,3}$, we have $a_{i,k} = t_{i,k} = g_{i,k} = 0$ for all $i \in \{0, 1, 2, 3\}$ and $k \in \{1, 2, 3\}$. Therefore,

$$g(AAAT) = g_0(T) = 0.$$

- For the codeword $\mathbf{c} = ATAT \in \mathcal{D}_{4,3}$, we have $a_{2,2} = a_{0,2} = 1$ and all other variables are zero. Therefore,

$$\begin{aligned} g(ATAT) &= g_2(T) + g_0(T) \\ &= \frac{1}{4}(N(3) - 3N(0)) \\ &\quad + \frac{1}{4}(N(1) - 3N(-2)) = 15 + 1 = 16. \end{aligned}$$

- For the codeword $\mathbf{c} = ATGC \in \mathcal{D}_{4,3}$, we have $a_{2,2} = a_{1,1} = t_{1,2} = a_{0,1} = t_{0,1} = g_{0,2} = 1$ and all other variables are zero. Note that by (23),

1. $g_3(A) = 0$,
2. As $c_3c_2 = AT$ and $c_4 = \zeta$, $a_{2,2} = 1$ so that

$$g_2(T) = \frac{a_{2,2}}{4}(N(3) - N(0)),$$

3. As $c_2c_1 = TG$ and $c_3 \neq T$, $a_{1,1} = t_{1,2} = 1$ so that

$$g_1(G) = \frac{a_{1,1}}{4}N(2) + \frac{t_{1,2}}{4}(N(2) - N(-1)), \text{ and}$$

4. As $c_1c_0 = GC$ and $c_2 \neq G$, $a_{0,1} = t_{0,1} = g_{0,2} = 1$ so that

$$g_0(C) = \frac{a_{0,1} + t_{0,1}}{4}N(1) + \frac{g_{0,2}}{4}(N(1) - N(-2)).$$

Therefore,

$$\begin{aligned} g(ATGC) &= \frac{1}{4}(N(3) - 3N(0)) \\ &\quad + \frac{1}{4}N(2) + \frac{1}{4}(N(2) - 3N(-1)) \\ &\quad + \frac{2}{4}N(1) + \frac{1}{4}(N(1) - 3N(-2)) \\ &= 15 + 4 + 4 + 2 + 1 = 26. \end{aligned}$$

- For the codeword $\mathbf{c} = GGGC \in \mathcal{D}_{4,3}$, we have $a_{3,1} = t_{3,1} = a_{2,1} = t_{2,1} = a_{1,1} = t_{1,1} = a_{0,1} = t_{0,1} = 1$ and all other variables are zero. Therefore,

$$\begin{aligned} g(GGGC) &= \frac{2}{4}N(4) + \frac{2}{4}N(3) \\ &\quad + \frac{2}{4}N(2) + \frac{2}{4}N(1) = 168. \end{aligned}$$

- For the codeword $\mathbf{c} = CCCG \in \mathcal{D}_{4,3}$, we have $a_{3,1} = t_{3,1} = g_{3,1} = a_{2,1} = t_{2,1} = g_{2,1} = a_{1,1} = t_{1,1} = g_{1,1} = a_{0,1} = t_{0,1} = 1$ and all other variables are zero. Therefore,

$$\begin{aligned} g(CCCG) &= \frac{3}{4}N(4) + \frac{3}{4}N(3) + \frac{3}{4}N(2) + \frac{2}{4}N(1) \\ &= 189 + 48 + 12 + 2 = 251. \end{aligned}$$

$$\begin{aligned}
g_i(G) &= N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}A) + N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}T) \\
&= \frac{3}{4} \left[(\mathbf{a}_{i,1} + \mathbf{t}_{i,1} + \mathbf{a}_{i,2} + \mathbf{t}_{i,2} + \mathbf{a}_{i,3} + \mathbf{t}_{i,3})N(i) \right. \\
&\quad \left. + (\mathbf{a}_{i,1} + \mathbf{t}_{i,1} + \mathbf{a}_{i,2} + \mathbf{t}_{i,2})N(i-1) + (\mathbf{a}_{i,1} + \mathbf{t}_{i,1})N(i-2) \right] \\
&= \frac{3}{4} \sum_{j=1}^3 \sum_{k=1}^j (\mathbf{a}_{i,k} + \mathbf{t}_{i,k})N(i+j-3). \tag{22}
\end{aligned}$$

$$\begin{aligned}
g(\mathbf{c}) &= \sum_{i=0}^{m-1} \left[\frac{\mathbf{a}_{i,1} + \mathbf{t}_{i,1} + \mathbf{g}_{i,1}}{4} N(i+1) + \frac{\mathbf{a}_{i,2} + \mathbf{t}_{i,2} + \mathbf{g}_{i,2}}{4} (N(i+1) - 3N(i-2)) \right. \\
&\quad \left. + \frac{3(\mathbf{a}_{i,3} + \mathbf{t}_{i,3} + \mathbf{g}_{i,3})}{4} N(i) \right]. \tag{23}
\end{aligned}$$

B. Encoding-Decoding Rule for General ℓ

Below, we study the symbol contributions of each symbol for general $\ell \geq 1$, and we obtain a formula (in terms of cardinalities $N(i)$) for the encoding-decoding rule $g : \mathcal{D}_{m,\ell} \rightarrow \{0, 1, \dots, N(m) - 1\}$ that gives the index of codewords in $\mathcal{D}_{m,\ell}$. Recall that $N_{\Lambda}(m)$ denotes the number of all codewords in $\mathcal{D}_{m,\ell}$ whose first letter from the left is Λ , and $N_{\Lambda,k}(m)$ (for $1 \leq k \leq \ell$) denotes the number of all codewords in $\mathcal{D}_{m,\ell}$ whose first k letters are Λ and $c_{m-k-1} \neq \Lambda$. Moreover, we have the relation $3N_{\Lambda}(m-k) = 4N_{\Lambda,k}(m)$ for $k \geq 1$.

Theorem 2. *The encoding-decoding rule $g : \mathcal{D}_{m,\ell} \rightarrow \{0, 1, \dots, N(m) - 1\}$, for general $\ell \geq 1$, is as follows:*

$$g(\mathbf{c}) = \frac{3}{4} \sum_{i=0}^{m-1} \sum_{j=1}^{\ell} \sum_{k=1}^j (\mathbf{a}_{i,k} + \mathbf{t}_{i,k} + \mathbf{g}_{i,k})N(i+j-\ell), \tag{24}$$

where $N(0) \triangleq 4/3$ and $N(-1) \triangleq \dots \triangleq N(2-\ell) \triangleq N(1-\ell) \triangleq 0$. Moreover, for all $\Pi > \Delta$ and for each $\Delta \in \{A, T, G\}$,

$$\begin{aligned}
\delta_{i,1} &= 1 \text{ only if } c_i = \Pi \text{ and } c_{i+1} \neq \Delta, \\
\delta_{i,k} &= 1 \text{ only if } c_{i+k-1} \dots c_{i+1}c_i = \Delta^{k-1}\Pi \text{ and } c_{i+k} \neq \Delta, \tag{25}
\end{aligned}$$

where $2 \leq k \leq \ell$ (these are $\ell-1$ statements), and in all other cases, $\delta_{i,k} = 0$ for $i \in \{1, 2, \dots, m\}$ and $k \in \{1, 2, \dots, \ell\}$. Here, $\Pi > \Delta$ is according to the lexicographic ordering rule, and δ stands for the small letter in $\{\mathbf{a}, \mathbf{t}, \mathbf{g}\}$ corresponding to Δ in $\{A, T, G\}$.

Proof: First, recall that $c_i \triangleq \zeta$ (a letter outside of the alphabet), for all $i > m-1$ (see Remark 1).

Symbol contributions are as follows:

Case 1: If $c_i = A$, it is clear that $g_i(A) = 0$.

Case 2: If $c_i = T$, then

$$\begin{aligned}
g_i(T) &= N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}A) \\
&= \mathbf{a}_{i,1}[N_{A,1}(i+1) + N_{A,2}(i+1) + \dots + N_{A,\ell}(i+1)] \\
&\quad + \mathbf{a}_{i,2}[N_{A,1}(i+1) + N_{A,2}(i+1) + \dots + N_{A,\ell-1}(i+1)]
\end{aligned}$$

$$\begin{aligned}
&+ \dots \\
&+ \mathbf{a}_{i,\ell-1}[N_{A,1}(i+1) + N_{A,2}(i+1)] + \mathbf{a}_{i,\ell}N_{A,1}(i+1) \\
&= (\mathbf{a}_{i,1} + \mathbf{a}_{i,2} + \dots + \mathbf{a}_{i,\ell})N_{A,1}(i+1) + \dots \\
&+ (\mathbf{a}_{i,1} + \mathbf{a}_{i,2})N_{A,\ell-1}(i+1) + \mathbf{a}_{i,1}N_{A,\ell}(i+1) \\
&= \frac{3}{4} \sum_{j=1}^{\ell} \sum_{k=1}^j \mathbf{a}_{i,k}N(i+j-\ell), \tag{26}
\end{aligned}$$

where $N(0) \triangleq 4/3$ and $N(-1) \triangleq \dots \triangleq N(2-\ell) \triangleq N(1-\ell) \triangleq 0$. Moreover,

$\mathbf{a}_{i,1} = 1$ only if $c_i = A$ and $c_{i+1} \neq A$,

$\mathbf{a}_{i,k} = 1$ only if $c_{i+k-1} \dots c_{i+1}c_i = \mathbf{A}^{k-1}T$ and $c_{i+k} \neq A$ where $2 \leq k \leq \ell$, and

$\mathbf{a}_{i,k} = 0$ otherwise.

Case 3 and Case 4: One can similarly study and find the numbers $N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}T)$ and $N_{\text{symb}}(m, c_{m-1}c_{m-2} \dots c_{i+1}G)$ as well to obtain $g_i(G)$ and $g_i(G)$.

Adding all $g_i(c_i)$'s up, we obtain the encoding-decoding rule as given in the theorem statement. ■

Example 3. Consider the encoding-decoding rule

$$g : \mathcal{D}_{5,4} \rightarrow \{0, 1, \dots, 1019\}$$

where $N(-3) \triangleq N(-2) \triangleq N(-1) \triangleq 0$, $N(0) \triangleq 4/3$, $N(1) = 4$, $N(2) = 16$, $N(3) = 64$, $N(4) = 256$, and $N(5) = 4^5 - 4 = 1020$. We discuss some instances and provide a detailed explanation for the codeword $TAATT$:

- For the codeword $\mathbf{c} = AAAAT \in \mathcal{D}_{5,4}$, we have $\mathbf{a}_{i,k} = \mathbf{t}_{i,k} = \mathbf{g}_{i,k} = 0$ for all $i \in \{0, 1, 2, 3, 4\}$ and $k \in \{1, 2, 3, 4\}$. Therefore,

$$g(AAAAT) = g_0(T) = 0.$$

- For the codeword $\mathbf{c} = TAATT \in \mathcal{D}_{5,4}$, we have $\mathbf{a}_{4,1} = \mathbf{a}_{1,3} = \mathbf{a}_{0,1} = 1$ and all other variables are zero. Note that by (24),

$$1. \quad g_3(A) = g_2(A) = 0,$$

2. As $c_4 = T$ and $c_5 = \zeta$, $\mathbf{a}_{4,1} = 1$ so that

$$g_4(T) = \frac{3\mathbf{a}_{4,1}}{4}(N(4) + N(3) + N(2) + N(1)),$$

3. As $c_3c_2c_1 = AAT$ and $c_4 \neq A$, $\mathbf{a}_{1,3} = 1$

$$g_1(T) = \frac{3\mathbf{a}_{1,3}}{4}(N(1) + N(0)), \text{ and}$$

4. As $c_1c_0 = TT$ and $c_2 \neq T$, $\mathbf{a}_{0,1} = 1$ so that

$$g_0(T) = \frac{3\mathbf{a}_{0,1}}{4}N(0).$$

Therefore,

$$\begin{aligned} g(TAATT) &= g_4(T) + g_1(T) + g_0(T) \\ &= \frac{3}{4}(N(4) + N(3) + N(2) + N(1)) \\ &\quad + \frac{3}{4}(N(1) + N(0)) + \frac{3}{4}N(0) \\ &= 255 + 4 + 1 = 260. \end{aligned}$$

- For the codeword $\mathbf{c} = GGGGC \in \mathcal{D}_{5,4}$, we have $\mathbf{a}_{4,1} = \mathbf{t}_{4,1} = \mathbf{a}_{3,1} = \mathbf{t}_{3,1} = \mathbf{a}_{2,1} = \mathbf{t}_{2,1} = \mathbf{a}_{1,1} = \mathbf{t}_{1,1} = \mathbf{a}_{0,1} = \mathbf{t}_{0,1} = 1$ and all other variables are zero. Therefore,

$$\begin{aligned} g(GGGGC) &= g_4(G) + g_3(G) + g_2(G) + g_1(G) + g_0(C) \\ &= \frac{3}{4}(2N(4) + 2N(3) + 2N(2) + 2N(1)) \\ &\quad + \frac{3}{4}(2N(3) + 2N(2) + 2N(1) + 2N(0)) \\ &\quad + \frac{3}{4}(2N(2) + 2N(1) + 2N(0)) \\ &\quad + \frac{3}{4}(2N(1) + 2N(0)) + \frac{3}{4}(2N(0)) \\ &= 510 + 128 + 32 + 8 + 2 = 680. \end{aligned}$$

- For the codeword $\mathbf{c} = CATGC \in \mathcal{D}_{5,4}$, we have $\mathbf{a}_{4,1} = \mathbf{t}_{4,1} = \mathbf{g}_{4,1} = \mathbf{a}_{2,2} = \mathbf{a}_{1,1} = \mathbf{t}_{1,2} = \mathbf{a}_{0,1} = \mathbf{t}_{0,1} = \mathbf{g}_{0,2} = 1$ and all other variables are zero. Therefore,

$$\begin{aligned} g(CATGC) &= g_4(C) + g_2(T) + g_1(G) + g_0(C) \\ &= \frac{3}{4}(3N(4) + 3N(3) + 3N(2) + 3N(1)) \\ &\quad + \frac{3}{4}(N(2) + N(1) + N(0)) \\ &\quad + \frac{3}{4}(2N(1) + 2N(0)) + \frac{3}{4}(3N(0)) \\ &= 765 + 16 + 8 + 3 = 792. \end{aligned}$$

- For the codeword $\mathbf{c} = CCCCCG \in \mathcal{D}_{5,4}$, we have $\mathbf{a}_{i,1} = \mathbf{t}_{i,1} = \mathbf{g}_{i,1} = 1$, for all $i \in \{1, 2, 3, 4\}$, $\mathbf{a}_{0,1} = \mathbf{t}_{0,1} = 1$, and all other variables are zero. Therefore,

$$\begin{aligned} g(CCCCCG) &= g_4(C) + g_3(C) + g_2(C) + g_1(C) + g_0(G) \\ &= \frac{3}{4}(3N(4) + 3N(3) + 3N(2) + 3N(1)) \\ &\quad + \frac{3}{4}(3N(3) + 3N(2) + 3N(1) + 3N(0)) \end{aligned}$$

$$\begin{aligned} &+ \frac{3}{4}(3N(2) + 3N(1) + 3N(0)) \\ &+ \frac{3}{4}(3N(1) + 3N(0)) + \frac{3}{4}(2N(0)) \\ &= 765 + 192 + 48 + 12 + 2 = 1019. \end{aligned}$$

Remark 3. We provide additional generalization of Theorem 2 in the Appendix to cover q -ary constrained codes eliminating runs of length exceeding $\ell \geq 1$ for any $q \geq 2$ (see Theorem 3).

Remark 4. The above examples illustrate the use of the encoding-decoding rule for the decoding procedure. In the next section, Example 4 briefly illustrates the use of the encoding-decoding rule for the encoding procedure. For the encoding-decoding algorithms, see Section V.

IV. BRIDGING AND ERROR DETECTION

In this section, we present our bridging schemes. Using a single codeword for the whole DNA strand results in higher encoding-decoding complexity and codeword-to-message error propagation [24], [26]. Hence, it is essential while encoding data in a DNA strand to concatenate shorter codewords in order to prevent forbidden patterns at the transition between consecutive codewords. This process is called bridging. Our bridging schemes do not allow same-symbol runs of length higher than 3, i.e., they are designed for D-LOCO codes with $\ell \geq 3$. We note that these schemes are selected to also enable error detection (which will be discussed in this section) without increasing the $AT - GC$ disparity (which will be discussed in the next section).

A. Bridging Scheme I

This is a one-symbol bridging scheme. Let \mathbf{c}_1 and \mathbf{c}_2 be two consecutive codewords in the DNA data stream, where \mathbf{c}_1 ends with the symbol Λ_1 and \mathbf{c}_2 starts with the symbol Λ_2 . We can bridge the two codewords \mathbf{c}_1 and \mathbf{c}_2 by inserting a symbol from the set $\{A, T, G, C\} \setminus \{\Lambda_1, \Lambda_2\}$. Moreover, this single bridging symbol enables encoding 1-bit of information as follows:

- To encode 0 (from the input message stream), we pick the letter having the lowest lexicographic index in $\{A, T, G, C\} \setminus \{\Lambda_1, \Lambda_2\}$.
- To encode 1, we pick the letter having the highest lexicographic index in $\{A, T, G, C\} \setminus \{\Lambda_1, \Lambda_2\}$.

B. Bridging Scheme II

We now discuss a simple three-symbol bridging scheme, which has two versions, to ensure single substitution error detection per codeword. Let \mathbf{c}_1 and \mathbf{c}_2 be two consecutive codewords, where \mathbf{c}_1 ends with the symbol Λ_1 and \mathbf{c}_2 starts with the symbol Λ_2 . Let also Λ_3 denote the check-sum of \mathbf{c}_1 , i.e., the check-sum of the symbols in \mathbf{c}_1 , where this check-sum is computed according to the correspondence ($c_i \longleftrightarrow a_i$):

$$\begin{aligned} A &\longleftrightarrow 0 \pmod{4}, & G &\longleftrightarrow 2 \pmod{4}, \\ T &\longleftrightarrow 1 \pmod{4}, & C &\longleftrightarrow 3 \pmod{4}. \end{aligned} \quad (27)$$

We define the integer (mod 4) equivalent of symbol c_i according to (27) as a_i . Note that Λ_3 will be updated shortly. Here, we bridge the two codewords \mathbf{c}_1 and \mathbf{c}_2 as follows:

- (i) Assign Λ_3 as the middle bridging symbol.
- (ii) Update the check-sum Λ_3 by adding (mod 4) the integer representation of the extra two message stream bits to encode within bridging (equivalent to one extra letter in total).
- (iii) Set Λ_4 to the letter having the lowest lexicographic index in $\{A, T, G, C\} \setminus \{\Lambda_1, \Lambda_3\}$ if the first bit to encode is 0.
- (iv) Set Λ_4 to the letter having the highest lexicographic index in $\{A, T, G, C\} \setminus \{\Lambda_1, \Lambda_3\}$ if the first bit to encode is 1.
- (v) Similarly, choose Λ_5 from the set $\{A, T, G, C\} \setminus \{\Lambda_2, \Lambda_3\}$ to encode the second message stream bit.

For clarity, the sequence order is $\Lambda_1 \Lambda_4 \Lambda_3 \Lambda_5 \Lambda_2$. This version is called Bridging Scheme II-A. A different treatment of Λ_5 creates the second version of the scheme.

If we instead choose Λ_5 as the letter having the highest lexicographic index in the set $\{A, T\} \setminus \{\Lambda_2\}$ in case $\Lambda_3 \in \{G, C\}$ and in the set $\{G, C\} \setminus \{\Lambda_2\}$ in case $\Lambda_3 \in \{A, T\}$, this version is called Bridging Scheme II-B. This version has a balancing advantage (see Remark 8 in Section V for more details). Observe that in this case, there is only 1 extra bit (say b) encoded within bridging. Thus, Λ_3 is updated by adding (mod 4) the integer representation of the binary 2-tuple $b0$.

We illustrate how to apply this bridging scheme in the following example:

Example 4. We encode the 38-bit binary message stream

$$\mathbf{b} = 1010.1000.1100.1111.1010.1010.1101.1010.0111.11$$

into codewords in $\mathcal{D}_{9,3}$ by applying Bridging Scheme II-A (b_i denotes the i^{th} digit of \mathbf{b} , starting from b_1 at the left):

- From Proposition 1, $N(5) = 996$, $N(6) = 3936$, $N(7) = 15552$, $N(8) = 61452$, and $N(9) = 242820$.
- Our rule is $g : \mathcal{D}_{9,3} \rightarrow \{0, 1, \dots, 242819\}$ in (23), where the message length is $\lfloor \log_2(242819) \rfloor = 17$. The first 17 bits form a message that corresponds to 86431 in decimal integer. We briefly illustrate how to obtain the codeword of index 86431 via the D-LOCO encoding-decoding rule:
 - Initialize residual = 86431.
 - At $i = 8$, $\frac{1}{4}N(9) \leq \text{residual} < \frac{2}{4}N(9)$. Consequently, $a_8 = 1$. Update residual = $86431 - \frac{1}{4}N(9) = 25726$.
 - At $i = 7$, $\frac{1}{4}(N(8) - N(5)) \leq \text{residual} < \frac{2}{4}N(8) = 30726$. Consequently, $a_7 = 1$. Update residual = $25726 - \frac{1}{4}(N(8) - N(5)) = 11110$.
 - At $i = 6$, $\frac{2}{4}N(7) \leq \text{residual} < \frac{3}{4}N(7) = 11664$. Consequently, $a_6 = 2$. Update residual = $11110 - \frac{2}{4}N(7) = 3334$.

After repeating this procedure until $i = 0$, we finally obtain

$$\mathbf{a}_1 = 112321323 \implies \mathbf{c}_1 = TTGCGTTCGC.$$

The bits b_{20} – b_{36} form a message that corresponds to 44455 in decimal integer, and the codeword corresponding to index 44455 is obtained similarly:

$$\mathbf{a}_2 = 023300311 \implies \mathbf{c}_2 = AGCCAACCTT.$$

- Since the check-sum of the first codeword is $1 + 1 + 2 + 3 + 2 + 1 + 3 + 2 + 3 \pmod{4} = 2$ and

$$b_{18}b_{19} = 01 \longleftrightarrow 1 \pmod{4},$$

the middle bridging symbol is C , corresponding to 3.

- Given that $b_{18} = 0$ in the binary message stream, the first bridging symbol between $TTGCGTTCGC$ and the check-sum symbol C is A (which is the letter having the lowest lexicographic index in $\{A, T, G\}$).
- Similarly, given that $b_{19} = 1$ in the message stream, the third bridging symbol between $AGCCAACCTT$ and the check-sum symbol C is G (which is the letter having the highest lexicographic index in $\{T, G\}$).
- By computing the check-sum of the second codeword and using $b_{37}b_{38}$, the last 3 coded symbols become CAC .

Hence, the given binary message stream is encoded as $TTGCGTTCGCACGAGCCAACCTTAC$. Note that if we drop b_{19} and b_{38} from the message stream above and adopt Bridging Scheme II-B, the resulting 36-bit message stream is encoded as $TTGCGTTCGCAGTAGCCAACCTTGCT$.

Remark 5. Note that if the written \mathbf{c}_1 has a single substitution error (or if an error occurs on the check-sum symbol itself), then \mathbf{c}_1 and the check-sum Λ_3 will be inconsistent, and in this case, the error is detected. Similarly, in case there is an error at the first or the third bridging symbol, Λ_4 or Λ_5 , it is again detected since the check-sum Λ_3 is computed by taking the extra information (two message stream bits) encoded in these bridging symbols into account. This verifies the error detection property for Bridging Scheme II-A.

Remark 6. Note that the bridging letters Λ_4 and Λ_5 are used to separate the check-sum symbol from the codewords in order for forbidden patterns not to show up at the transition. However, one symbol is the minimum we can allocate for the check-sum given the run-length constraint. In the next subsection, we show how using more symbols for the check-sum can reduce the probability of not detecting errors.

C. Bridging Scheme III

We now discuss a five-symbol bridging scheme that ensures single substitution error detection per codeword for a D-LOCO code forbidding runs of length at least 4, i.e., $\ell = 3$, and achieves lower probability of no-detection in case multiple errors occur. Let \mathbf{c} and \mathbf{d} in $\mathcal{D}_{m,3}$ be two consecutive codewords whose length is divisible by 3, i.e., $m = 3m'$ for some integer m' . Moreover, we have $\mathbf{c} = \mathbf{c}_1\mathbf{c}_2\mathbf{c}_3$. Suppose \mathbf{c} ends with the symbol Λ_1 and \mathbf{d} starts with the symbol Λ_2 . Let also $\Lambda_{3,i}$ denote the check-sum of the codeword \mathbf{c}_i for $i = 1, 2, 3$. Here, we bridge the two codewords \mathbf{c} and \mathbf{d} as follows:

- (i) Assign $\Lambda_3 \triangleq \Lambda_{3,1}\Lambda_{3,2}\Lambda_{3,3}$ as the middle bridging pattern.
- (ii) Set Λ_4 to the letter having the highest lexicographic index in the set $\{A, T\} \setminus \{\Lambda_1\}$ in case $\Lambda_{3,1} \in \{G, C\}$ and in the set $\{G, C\} \setminus \{\Lambda_1\}$ in case $\Lambda_{3,1} \in \{A, T\}$.
- (iii) Set Λ_5 to the letter having the highest lexicographic index in the set $\{A, T\} \setminus \{\Lambda_2\}$ in case $\Lambda_{3,3} \in \{G, C\}$ and in the set $\{G, C\} \setminus \{\Lambda_2\}$ in case $\Lambda_{3,3} \in \{A, T\}$.

For clarity, the sequence order is $\Lambda_1 \Lambda_4 \Lambda_{3,1} \Lambda_{3,2} \Lambda_{3,3} \Lambda_5 \Lambda_2$.

Example 5. If we drop the four bits b_{18} , b_{19} , b_{37} , and b_{38} from the binary message stream \mathbf{b} in Example 4 and adopt Bridging Scheme III, the resulting 34-bit message stream is encoded as $TTGCGTTCGCAGACAGCCAACCTTCTCTC$.

D. Probability of Not Detecting Errors

Now, we study the probability of missing substitution errors when Bridging Schemes III and II-B are applied. The focus is on detecting errors in codewords, check-sum symbols, and bridging symbols within which message stream bits are encoded. Below, we start with a simple DNA storage channel where substitution errors dominate the error profile, which allows us to compare the two bridging schemes.

We assume that we have a $(1-p, p/3, p/3, p/3)$ -symmetric DNA storage channel with symbol substitution error rate $p \in [0, 1]$. This is a channel with 4-ary input and 4-ary output such that

$$\begin{aligned}\mathbb{P}(Y = \Lambda | X = \Lambda) &= 1 - p \\ \mathbb{P}(Y = \Lambda' | X = \Lambda) &= p/3,\end{aligned}$$

for any letters $\Lambda \in \{A, T, G, C\}$ and $\Lambda' \in \{A, T, G, C\} \setminus \{\Lambda\}$. For example, Organick et al. performed extensive experiments and reported substitution, deletion, and insertion rates as 4.5×10^{-3} , 1.5×10^{-3} , and 5.4×10^{-4} , respectively [38]. We can see that the substitution error rate here is the dominant error rate, and this value of 4.5×10^{-3} can be considered a typical value of p based on [38].

We start the analysis with Bridging Scheme III. Suppose we use codewords in $\mathcal{D}_{3m', \ell}$ and this bridging scheme is applied. For $i \in \{1, 2, 3\}$, let

$$\begin{aligned}U_i &\triangleq \text{the event that the check-sum represented by} \\ &\Lambda_{3,i} \text{ is satisfied in the read codeword } \mathbf{c} = \mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3, \\ &\text{i.e., } \Lambda_{3,i} = \sum_{k=0}^{m'-1} a_{i,k} \pmod{4} \text{ for } \mathbf{c}_i, \\ E_i &\triangleq \text{the event that there are errors in } \mathbf{c}_i \text{ or in } \Lambda_{3,i}, \\ P_i &\triangleq \mathbb{P}(U_i \wedge E_i) \\ &= \text{the probability that there are errors in } \mathbf{c}_i \\ &\text{or in } \Lambda_{3,i} \text{ but are undetected (check-sum satisfied).}\end{aligned}$$

Clearly, $P_1 = P_2 = P_3$ for the three constituent sequences. Next, we give upper bounds for P_1 .

Observe that if a single error occurs in \mathbf{c}_1 or in $\Lambda_{3,1}$, it is detected. Moreover, the probability of no-detection in case r errors occur, where $r \geq 2$, is at most $1/3$ because the number of error patterns at r specific locations which (i) do not affect the check-sum and (ii) do not yield a forbidden pattern is at most 3^{r-1} out of 3^r possible error patterns. Thus,

$$P_1 \leq \frac{1}{3} \left(1 - (1-p)^{m'+1} - (m'+1)p(1-p)^{m'} \right), \quad (\text{Bound I}). \quad (28)$$

In fact, via a more careful analysis, we can give a closer upper bound for P_1 as follows. Observe that for r specific locations, the integer-equivalent options of the error patterns that will go undetected are

$$1-3, 2-2, 3-1 \text{ when } r = 2, \text{ and}$$

$$1-1-2, 1-2-1, 2-1-1, 2-3-3, 3-2-3, 3-3-2 \text{ when } r = 3.$$

We denote this number of error-pattern options by $C(r)$. Since every undetected r -error pattern comes from a unique

$(r-1)$ -error pattern that is guaranteed to be detected (its error check-sum is not $0 \pmod{4}$), we have the following non-homogeneous linear recurrence relation

$$C(r) = 3^{r-1} - C(r-1). \quad (29)$$

The closed-form solution of this relation that satisfies $C(2) = 3$ can be derived using the z-transform, and it is

$$C(r) = \frac{3^r - 3(-1)^{r-1}}{4}. \quad (30)$$

Hence, the probability of no-detection in case of r errors, $r \geq 2$, is at most $C(r)/3^r = [1 - (-1/3)^{r-1}]/4$, and we express a closer upper bound (Bound II) for P_1 in (31).

The frame-level probability P_{unn} that there are errors in the codeword \mathbf{c} and/or the symbols $\Lambda_{3,1}\Lambda_{3,2}\Lambda_{3,3}$ but are undetected is the probability that all the check-sums are satisfied and at least one of the events E_1, E_2 , and E_3 occurs. Thus, we have the following expression for P_{unn} :

$$\begin{aligned}P_{\text{unn}} &= \mathbb{P}(U_1 \wedge U_2 \wedge U_3 \wedge (E_1 \vee E_2 \vee E_3)) \\ &= 3P_1(1-p)^{2(m'+1)} + 3P_1^2(1-p)^{m'+1} + P_1^3. \quad (32)\end{aligned}$$

As for Bridging Scheme II-B, the frame-level probability P_{unn} that there are errors in the coded sequence $\mathbf{c} \Lambda_4 \Lambda_3$, where $\mathbf{c} \in \mathcal{D}_{m, \ell}$, but are undetected can be directly bounded using Bound II in (31) by replacing $m'+1$ by $m+2$ (these are m symbols in \mathbf{c} , Λ_4 , and Λ_3).

Fig. 1 and Fig. 2 compare the upper bound on the probability of no-detection P_{unn} (using Bound II) of Bridging Scheme III and Bridging Scheme II-B at various values of the channel substitution error rate p . Fig. 1 compares the no-detection performance of Bridging Scheme III for the code $\mathcal{D}_{3m', 3}$ with $m' = 13$ and of Bridging Scheme II-B for the code $\mathcal{D}_{m, 3}$ with $m = 21$. The rates of these two codes are 0.8542 and 0.8636, respectively. Fig. 2 compares the no-detection performance of Bridging Scheme III for the code $\mathcal{D}_{3m', 3}$ with $m' = 21$ and of Bridging Scheme II-B for the code $\mathcal{D}_{m, 3}$ with $m = 33$. The rates of these two codes are 0.9028 and 0.9044, respectively. The choice of code lengths is such that the rates are close for the two coding schemes being compared. Note that code rate discussions are presented in Section VI, and this rate depends on the D-LOCO code, its bridging, and its balancing penalty.

There are three takeaways from the two figures. First, both bridging schemes offer notably low no-detection probabilities for substitution error rates at or below 0.01. These error rates are of practical importance in DNA storage systems, and we refer the reader to [38], where Illumina NextSeq sequencing is adopted. Second, Bridging Scheme III outperforms Bridging Scheme II-B in terms of no-detection probability by up to 1 order of magnitude in the high substitution error rate regime.² These error rates are common for sequencing methods such as Oxford Nanopore sequencing [39]. Third, the gain of Bridging Scheme III over Bridging Scheme II-B increases as the code rate increases, i.e., as the length increases.

Observe that if a D-LOCO code is paired with an error-correction code such that the former is the inner code (closer

²If the comparison is at fixed frame length for similar rates, this gain becomes even higher since the frame length is $3m'+5$ for Bridging Scheme III while it is only $m+3$ for Bridging Scheme II-B.

$$P_1 \leq \sum_{r=2}^{m'+1} \left[\frac{1 - (-1/3)^{r-1}}{4} \right] \binom{m'+1}{r} p^r (1-p)^{m'+1-r} = \frac{1}{3} \binom{m'+1}{2} p^2 (1-p)^{m'-1} + \frac{2}{9} \binom{m'+1}{3} p^3 (1-p)^{m'-2} + \frac{7}{27} \binom{m'+1}{4} p^4 (1-p)^{m'-3} + \dots + \left[\frac{1 - (-1/3)^{m'}}{4} \right] p^{m'+1}, \text{ (Bound II).} \quad (31)$$

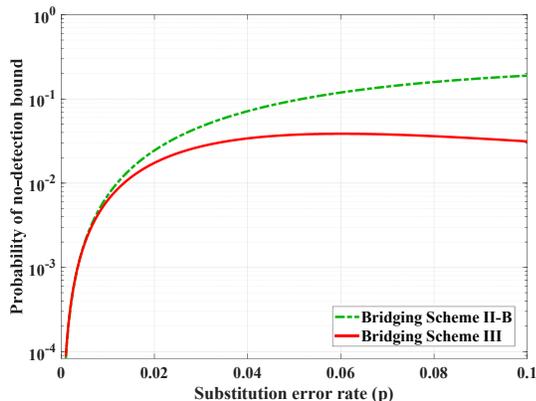


Fig. 1. Upper bounds on the probability of no-detection for $\mathcal{D}_{3m',3}$ and Bridging Scheme III versus $\mathcal{D}_{m,3}$ and Bridging Scheme II-B, $m' = 13$ and $m = 21$.

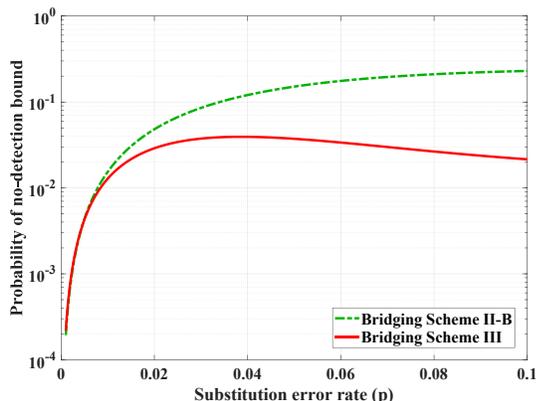


Fig. 2. Upper bounds on the probability of no-detection for $\mathcal{D}_{3m',3}$ and Bridging Scheme III versus $\mathcal{D}_{m,3}$ and Bridging Scheme II-B, $m' = 21$ and $m = 33$.

to the channel), achieving low no-detection probability makes the task of the error-correction decoder easier. Observe also that Bridging Scheme III, because of the higher code length required to achieve the same rate, incurs higher storage overhead and error propagation compared with Bridging Scheme II-B (see the following sections, [15], and [24]). Finally, we note that similar conclusions can also be reached for $\ell > 3$.

V. ALGORITHMS AND BALANCING THE DNA SEQUENCE

In this section, we first present the encoding and decoding algorithms of D-LOCO codes equipped with Bridging Scheme I, and then discuss how to balance the GC-content of the designed DNA sequence.

We now discuss how to balance the DNA sequence or stream of bridged D-LOCO codewords in a way that incurs

Algorithm 1 Encoding D-LOCO Codes with Bridging Scheme I

- 1: **Inputs:** Incoming stream of binary messages and the highest run-length allowed ℓ .
- 2: Use the cardinalities $N_D(r, \ell)$, $r \in \{\ell + 1, \ell + 2, \dots\}$ (computed offline by (3)), where $N_D(r, \ell) = 4^r$ for $0 < r \leq \ell$ and $N_D(0, \ell) = 4/3$.
- 3: Specify m as the smallest r in Step 2 to achieve the desired rate. Then, $s = \lceil \log_2 N_D(m, \ell) \rceil$.
- 4: **for** each incoming message \mathbf{b} of length $s + 1$ **do**
- 5: Compute $g(\mathbf{c}) = \text{decimal}(\mathbf{b})$.
- 6: Initialize residual with $g(\mathbf{c})$ and c_i with ζ for $i \geq m$. (ζ indicates out of codeword bounds)
- 7: **for** $i \in \{m - 1, m - 2, \dots, 0\}$ **do** (in order)
- 8: For each $\chi \in \{T, G, C\}$, set $c_i = \chi$ (temporarily), determine the coefficients $\mathbf{a}_{i,k}, \mathbf{t}_{i,k}, \mathbf{g}_{i,k}$ for $1 \leq k \leq \ell$ based on (25), and then compute $M(i, j, \chi) = \sum_{k=1}^j (\mathbf{a}_{i,k} + \mathbf{t}_{i,k} + \mathbf{g}_{i,k})$ for all $1 \leq j \leq \ell$.
- 9: **for** $\chi \in \{T, G, C\}$ **do**
- 10: Compute $\text{contrib}(i, \chi) = \frac{3}{4} \sum_{j=1}^{\ell} M(i, j, \chi) N(i + j - \ell)$.
- 11: **end for**
- 12: **if** residual $\geq \text{contrib}(i, C)$ **then**
- 13: Encode $c_i = C$.
- 14: **else if** residual $\geq \text{contrib}(i, G)$ **then**
- 15: Encode $c_i = G$.
- 16: **else if** residual $\geq \text{contrib}(i, T)$ **then**
- 17: Encode $c_i = T$.
- 18: **else**
- 19: Encode $c_i = A$.
- 20: **end if**
- 21: residual \leftarrow residual $- \text{contrib}(i, c_i)$, $c_i \neq A$.
- 22: **if** (not first codeword) $\wedge (i = m - 1)$ **then**
- 23: Bridge with a symbol other than c_{m-1} or the right-most symbol of the previous codeword based on the last bit of the incoming message as described in Subsection IV-A.
- 24: **end if**
- 25: **end for**
- 26: **end for**
- 27: **Output:** Outgoing stream of bridged D-LOCO codewords.

Algorithm 2 Decoding D-LOCO Codes with Bridging Scheme I

- 1: **Inputs:** Incoming stream of 4-ary D-LOCO codewords, in addition to m and s .
 - 2: Use the cardinalities $N_D(r, \ell)$, $r \in \{\ell+1, \ell+2, \dots, m-1\}$ (computed offline by (3)), where $N_D(r, \ell) = 4^r$ for $0 < r \leq \ell$ and $N_D(0, \ell) = 4/3$.
 - 3: **for** each incoming codeword \mathbf{c} of length m **do**
 - 4: Initialize $g(\mathbf{c})$ with 0 and c_i with ζ for $i \geq m$. (ζ indicates out of codeword bounds)
 - 5: **for** $i \in \{m-1, m-2, \dots, 0\}$ **do** (in order)
 - 6: Determine the coefficients $\mathbf{a}_{i,k}, \mathbf{t}_{i,k}, \mathbf{g}_{i,k}$ for $1 \leq k \leq \ell$ based on (25).
 - 7: Compute $\text{contrib}(c_i) = \frac{3}{4} \sum_{j=1}^{\ell} \sum_{k=1}^j (\mathbf{a}_{i,k} + \mathbf{t}_{i,k} + \mathbf{g}_{i,k}) N(i+j-\ell)$.
 - 8: $g(\mathbf{c}) \leftarrow g(\mathbf{c}) + \text{contrib}(c_i)$.
 - 9: **end for**
 - 10: Compute $\mathbf{b} = \text{binary}(g(\mathbf{c}))$, which has length s .
 - 11: Concatenate the bit encoded in the next bridging symbol (as described in Subsection IV-A). Then, skip this bridging symbol.
 - 12: **end for**
 - 13: **Output:** Outgoing stream of binary messages.
-

minimal rate penalty. Balancing is needed to have a GC-content close to %50 for the DNA strand in order to achieve high reliability of the strand synthesized [1], [2].

Definition 2. The disparity, denoted by $p(\mathbf{c})$, of a codeword \mathbf{c} in $\mathcal{D}_{m,\ell}$ is defined as the difference between the total number of symbols G, C and the total number of symbols A, T , i.e.,

$$p(\mathbf{c}) = |G| + |C| - |A| - |T|, \quad (33)$$

where $|\Lambda|$ denotes the number of occurrences of the symbol Λ in \mathbf{c} . The absolute value of $p(\mathbf{c})$ is called the absolute disparity of \mathbf{c} . The global disparity of a DNA sequence/stream of length M consisting of K codewords \mathbf{c}_i , i in $\{1, 2, \dots, K\}$, along with their bridging symbols is defined as the absolute value of the sum of disparities of all codewords and their bridging in the DNA sequence, and it is given by

$$p^{\text{global}} = \left| \sum_{i=1}^K p(\mathbf{c}_i) + \psi \right| \in [0, M], \quad (34)$$

where ψ is the cumulative disparity resulting from all bridging symbols.³ The global disparity fraction is defined as $p^{\text{global}}/M \in [0, 1]$. Note that if the global disparity fraction is in $[0, 0.1]$, then the GC-content of the DNA sequence is between %45 and %55 (see also [29] for further discussions regarding the GC-content of a DNA sequence).

Observe that our bridging schemes introduced in Section IV contribute to ensuring low global disparity fraction of the DNA sequence. For some of our bridging schemes, the global disparity fraction is in $[0, \frac{1}{K})$ as discussed in Remark 8.

³For ranges of integer variables, such as the disparity, the notation $[a, b]$ means $\{a, a+1, \dots, b\}$ and the notation (a, b) means $\{a+1, a+2, \dots, b-1\}$. We use $[a, b]$ and (a, b) for brevity.

A. Balancing the D-LOCO Codes of Odd Lengths

Suppose each message is encoded into a codeword in $\mathcal{D}_{m,\ell}$, where m is odd and Bridging Scheme I is applied. Note that each sequence of odd length has necessarily nonzero disparity. For a codeword $\mathbf{c} \in \mathcal{D}_{m,\ell}$, let $\bar{\mathbf{c}}$ be the codeword in $\mathcal{D}_{m,\ell}$ obtained by replacing each A, T, G, C 's in \mathbf{c} by C, G, T, A , respectively. Here, $\bar{\mathbf{c}}$ will be called the *complement* of \mathbf{c} as $g(\bar{\mathbf{c}}) + g(\mathbf{c}) = N(m) - 1$ (see also [26]). For example, the complement of $GAATC$ is $TCCGA$ and vice versa. Observe that this is possible for any $\mathbf{c} \in \mathcal{D}_{m,\ell}$ because of the intrinsic symmetry of the D-LOCO code stemming from the intrinsic symmetry of the set of forbidden patterns. Observe also the relation between disparities: $p(\bar{\mathbf{c}}) = -p(\mathbf{c})$.

Lemma 1. Using the above setup, there is a balancing procedure so that a D-LOCO coded sequence of length $K(m+1)$ has global disparity in $[-m-1, m+1]$, for odd m . In particular, the order of the global disparity is $O(m)$, and thus, it is independent of the overall sequence length for fixed m .

Proof: We prove this by induction on K . If $K = 1$, the result is immediate. For an induction hypothesis, assume that a coded sequence \mathbf{s} of length $K(m+1)$ has global disparity in $[-m-1, m+1]$. The $(K+1)^{\text{th}}$ codeword \mathbf{c} can be replaced, if necessary, by its complement $\bar{\mathbf{c}}$ so that it has opposite disparity sign to the one of \mathbf{s} (while the new bridging letter can have any disparity). Thus, the resulting coded sequence will have disparity in $[-m-1, m+1]$. The minimum (resp., maximum) disparity is achieved, for example, if \mathbf{s} has disparity $-m-1$ (resp., $m+1$) and the new codeword along with its bridging symbol have disparity 0. This finishes the induction argument. ■

Remark 7. Using the above balancing procedure, there are two codewords to encode each message. Thus, one can use at most “half of the D-LOCO codebook” for distinct messages, i.e., precisely $2^{\lfloor \log_2(\frac{1}{2}N(m)) \rfloor}$ -many codewords correspond to unique messages each. Hence, our D-LOCO codes achieve the minimum possible rate loss; specifically, the one-bit penalty in the message length achieved by our balancing procedure results in the minimum rate loss. Moreover, this balancing penalty does not affect the capacity achievability of D-LOCO codes (see Section VI in [26] for a thorough discussion).

Remark 8. We can similarly show the following:

(i) For odd m , a coded sequence of length $K(m+3)$, where Bridging Scheme II-A is applied, has global disparity in $[-m-2K-1, m+2K+1]$,

(ii) If we go without the extra 1-bit of information encoded in the bridging symbol Λ_5 and adopt Bridging Scheme II-B, we can attain global disparity in the range $[-m-1, m+1]$ for a coded sequence of length $K(m+3)$. In this case, the global disparity fraction is in the range $[0, \frac{1}{K})$ and for $K = 10$, the GC-content of the DNA sequence is guaranteed to be between 45% and 55%. An even better result is obtained for $K = 25$, where the GC-content of the DNA sequence is guaranteed to be between 48% and 52%.

(iii) Similarly and for odd $m = 3m'$, a coded sequence of length $K(m+5)$ has global disparity in $[-m-1, m+1]$ when Bridging Scheme III is applied.

TABLE I
NORMALIZED RATES WHEN DIFFERENT BRIDGING SCHEMES ARE APPLIED AT VARIOUS LENGTHS FOR $\ell = 3$

m	R_1	m	R_2	m	R_3	m'	R_4
9	0.8500	9	0.7500	9	0.7083	5	0.7000
13	0.8929	13	0.8125	13	0.7812	7	0.7692
21	0.9318	21	0.8750	21	0.8541	11	0.8421
33	0.9559	33	0.9167	33	0.9027	17	0.8929
51	0.9712	51	0.9444	51	0.9351	21	0.9044
99	0.9800	99	0.9657	99	0.9607	33	0.9375
Capacity	0.9912	Capacity	0.9912	Capacity	0.9912	Capacity	0.9912

Observe that our disparity range is for any number of codewords K , small, medium, or large. Therefore, our balancing procedure not only achieves a global balancing criterion, but also achieves a local balancing one.

VI. ACHIEVABLE RATES FOR $\ell = 3$ AND LITERATURE COMPARISON

An *FSTD* is a state diagram that represents the infinitude of a sequence in which, some chosen patterns are forbidden. The *FSTD* corresponding to the set \mathcal{F} of patterns consisting of runs of length $\ell + 1$ (see Definition 1) is given in Fig. 3. This *FSTD* has ℓ states, and state F_j represents the case where the last $j + 1$ generated symbols are $\Lambda' \Lambda^j$, where $\Lambda \in \{A, T, G, C\}$ and $\Lambda' \in \{A, T, G, C\} \setminus \{\Lambda\}$ arbitrarily. In particular, for $\ell = 3$, state F_1 represents the case where the last two generated symbols are $\Lambda' \Lambda$, F_2 represents the case where the last three symbols are $\Lambda' \Lambda \Lambda$, and F_3 represents the case where the last four symbols are $\Lambda' \Lambda \Lambda \Lambda$, where $\Lambda \in \{A, T, G, C\}$ and $\Lambda' \in \{A, T, G, C\} \setminus \{\Lambda\}$ arbitrarily. The symbols on the directed transition edges represent the currently generated symbols. The adjacency matrix of the *FSTD* for fixed $\ell \geq 3$ is denoted by $\mathbf{F}^{(\ell)}$, and it is of size $\ell \times \ell$. The entry $f_{h,p}$, $1 \leq h, p \leq \ell$, is the number of times state F_h is connected to state F_p (from F_h to F_p). Consequently,

$$\mathbf{F}^{(\ell)} = \begin{bmatrix} 3 & & & \\ \vdots & & & \\ 3 & & \mathbf{I}_{\ell-1} & \\ 3 & 0 & \dots & 0 \end{bmatrix}, \quad (35)$$

where $\mathbf{I}_{\ell-1}$ is the identity matrix of size $(\ell - 1) \times (\ell - 1)$, and

$$\mathbf{F}^{(3)} = \begin{bmatrix} 3 & 1 & 0 \\ 3 & 0 & 1 \\ 3 & 0 & 0 \end{bmatrix}. \quad (36)$$

The characteristic polynomial of $\mathbf{F}^{(3)}$ is then:

$$\det(\beta I - \mathbf{F}^{(3)}) = \beta^3 - 3\beta^2 - 3\beta - 3. \quad (37)$$

Therefore, the largest real positive eigenvalue of $\mathbf{F}^{(3)}$ is $\beta_{\max} = 3.9514$, and the normalized capacity for $\ell = 3$ is $C^{(3)} = \log_4(\beta_{\max}) = 0.9912$.

We now discuss the finite-length rates of D-LOCO codes for each bridging scheme we propose and with the requirement of balancing.

- The normalized rate R_1 of the D-LOCO code $\mathcal{D}_{m,\ell}$ when Bridging Scheme I is adopted is given by

$$R_1 = \frac{\lfloor \log_2(\frac{1}{2}N(m)) \rfloor + 1}{2(m+1)} = \frac{\lfloor \log_2(N(m)) \rfloor}{2(m+1)}. \quad (38)$$

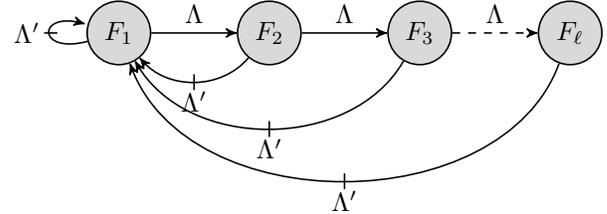


Fig. 3. An *FSTD* of \mathcal{F} -constrained sequences, where $\Lambda' \in \{A, T, G, C\} \setminus \{\Lambda\}$. Note that Λ represents the last generated symbol at any state. Upon entering F_1 , Λ is always updated to become the relevant Λ' , and Λ' symbols on different *FSTD* edges are not necessarily the same.

- The normalized rate R_2 of the D-LOCO code $\mathcal{D}_{m,\ell}$ when Bridging Scheme II-A is adopted is given by

$$R_2 = \frac{\lfloor \log_2(\frac{1}{2}N(m)) \rfloor + 2}{2(m+3)} = \frac{\lfloor \log_2(N(m)) \rfloor + 1}{2(m+3)}. \quad (39)$$

- The normalized rate R_3 of the D-LOCO code $\mathcal{D}_{m,\ell}$ when Bridging Scheme II-B is adopted is given by

$$R_3 = \frac{\lfloor \log_2(\frac{1}{2}N(m)) \rfloor + 1}{2(m+3)} = \frac{\lfloor \log_2(N(m)) \rfloor}{2(m+3)}. \quad (40)$$

- The normalized rate R_4 of the D-LOCO code $\mathcal{D}_{3m',\ell}$ when Bridging Scheme III is adopted is given by

$$R_4 = \frac{\lfloor \log_2(\frac{1}{2}N(3m')) \rfloor}{2(3m'+5)} = \frac{\lfloor \log_2(N(3m')) \rfloor - 1}{2(3m'+5)}. \quad (41)$$

For balanced D-LOCO codes, Table I shows the normalized rates at various values of m when different bridging schemes are applied using (38)–(41).

Next, we present brief comparisons between D-LOCO codes and other codes designed for similar goals:

- 1) Nguyen et al. [31] introduced constrained codes that are capable of correcting a single error of any type based on enumerative approach (Method A) as well as a sequence replacement method (Method B) for DNA data storage. While their codes from Method A are rate-wise efficient, D-LOCO codes offer higher rates at lower lengths and also simpler encoding-decoding compared with their unrank-rank approach (see also [25]).
- 2) Improving [29], Wang et al. [30] offered an efficient coding scheme, a product of which is a code with a normalized rate of 0.8125 at length 8 that eliminates homopolymers of length at least 4 and has a guaranteed 40% – 60% *GC*-content. This code is based on look-up tables. The nature of our work differs from theirs mainly in three aspects. First, we have more control over the

TABLE II
COMPARISON BETWEEN OUR CODING SCHEMES ELIMINATING HOMOPOLYMERS OF LENGTH ≥ 4 WITH INCREASING RATES

Strand sequence length	GC -content	Single substitution error detection	Normalized rate	Storage overhead	Length and bridging method
288	45% – 55%	Yes	0.8125	703 bits	$m' = 9, K = 9, \text{III}$
240	45% – 55%	Yes	0.8541	421 bits	$m = 21, K = 10, \text{II-B}$
180	45% – 55%	No	0.9167	273 bits	$m = 17, K = 10, \text{I}$
324	42% – 58%	Yes	0.9167	1057 bits	$m = 33, K = 9, \text{II-A}$
220	45% – 55%	No	0.9318	421 bits	$m = 21, K = 10, \text{I}$

disparity, or equivalently the GC -content, of the DNA sequence, both globally and locally. One can choose a practical code length and bridging scheme to achieve low global disparity fraction such as 0.1, equivalently 45% – 55% GC -content (see Table II), or even lower. Second, our encoding and decoding algorithms are based on small-size adders, offering low complexity. Third, we also propose novel ideas for bridging to guarantee a single substitution error detection.

- 3) Park et al. [36] proposed an iterative encoding algorithm that has a mapping table with 48 3-tuple 4-ary entries as a building block for a constrained code that addresses the GC -content and the maximum homopolymer length requirements. There is a similar table for the iterative decoding algorithm as well. For instance, they achieve a normalized code rate of 0.9165 and 45% – 55% GC -content range. Our approach differs from theirs in terms of encoding-decoding, and we offer local balancing as well as substitution error detection. Moreover, the storage overhead associated with their encoding-decoding algorithms is higher than that associated with our algorithms since we only need to store the cardinalities to be used in the execution of the rule (see Section VII).
- 4) Liu et al. in [37] proposed a constrained coding scheme based on the unrank-rank approach, enumerating all constrained sequences that achieve a GC -content in $[0.5 - \epsilon, 0.5 + \epsilon]$ and do not contain homopolymers of length larger than ℓ . Their encoding-decoding algorithms have polynomial execution time and storage overhead. They also offer a coding scheme that satisfies a local GC -content constraint on sequence prefixes in order to further improve immunity against errors. Since both are constructed by employing enumerative coding techniques, their codes and our D-LOCO codes are capacity-achieving. Since the time complexity of the encoding-decoding algorithms of our D-LOCO codes is algorithmically $O(m)$ and implementation-wise $O(m^2)$ with respect to the code length m (see the detailed discussion at the end of Subsection VII-B), they are at least as efficient as those of the codes proposed in [37]. D-LOCO codes also have only $O(m^2)$ storage overhead (see (43)). We opt to use moderate lengths in order to mitigate codeword-to-message error propagation. We also take finite-length features into account, such as the effect of bridging and flooring (see (38)–(41)), while computing our rates in Table I. Furthermore, our codes offer substitution error-detection, reconfigurability and efficient local balancing (see Subsection VII-E).

VII. SOME PROPERTIES OF D-LOCO CODES

We now discuss additional properties of D-LOCO codes regarding

- capacity-achievability,
- complexity and storage,
- ease of reconfigurability,
- error propagation,
- parallelism, and
- local GC -content balance.

A. Capacity-Achievability

The *normalized asymptotic rate*, i.e., the normalized capacity, of a constrained code having length m and forbidding the patterns in \mathcal{F} is

$$C^{(\ell)} = \lim_{m \rightarrow \infty} \frac{\log_2(N(m))}{2m}, \quad (42)$$

which is the base-4 logarithm of the largest real positive root of the polynomial

$$x^\ell - 3x^{\ell-1} - 3x^{\ell-2} - \dots - 3$$

(consistent with the recursive relation in Proposition 1) [3].

We note the following about D-LOCO codes:

- (i) All codewords satisfying the constraint are included in the D-LOCO codebook.
- (ii) The number of added bits/symbols for bridging is independent of m for all bridging schemes proposed.
- (iii) The rate loss due to our balancing has the highest possible vanishing rate with respect to m , which is $O(1)/O(m)$. This rate loss asymptotically goes to 0.

Using (i) and (ii) along with all the rate equations (38)–(41) for all bridging schemes, we deduce that a D-LOCO code $\mathcal{D}_{m,\ell}$ is **capacity-achieving**. Using (iii) and the same set of rate equations, we deduce that our balancing procedure has no effect on capacity-achievability.

B. Complexity and Storage Overhead

The encoding algorithm of D-LOCO codes is based on comparisons and subtractions, whereas the decoding algorithm mainly uses additions. The size of the adders used to perform these tasks is the base-2 logarithm of the maximum value $g(\mathbf{c})$ can take that corresponds to a message, and it is given by:

$$\left\lceil \log_2 \left(\frac{1}{2} N(m) \right) \right\rceil,$$

which is the message length (see Remark 7). Table III illustrates the encoding and decoding complexity of D-LOCO

TABLE III
 ADDER SIZES REQUIRED AT VARIOUS LENGTHS FOR $\ell = 3$ WHEN
 DIFFERENT BRIDGING SCHEMES ARE APPLIED

I, II-A, or II-B is applied		III is applied	
m	Adder size	m'	Adder size
9	16 bits	5	28 bits
21	40 bits	11	64 bits
33	64 bits	17	100 bits
51	100 bits	21	123 bits
99	195 bits	33	195 bits

codes through the size of the adders to be used for different lengths and different bridging schemes adopted. For example, for a D-LOCO code with Bridging Scheme II-A, if a rate of 0.75 is satisfactory, adders of size just 16 bits are all that is needed. In case the rate needs to be 0.8750, adders of size 40 bits should be used. Moreover, for a D-LOCO code with Bridging Scheme III, if a rate of 0.70 is satisfactory, small adders of size just 28 bits are all that is needed. In case the rate needs to be about 0.8421, adders of size 64 bits should be used. All rates are normalized.

As for the storage overhead, we need to store the values in $\{3N(i)/4 \mid 0 \leq i \leq m-1\}$, for code length m , offline in order to execute the encoding-decoding rule in (11) and also in (24). Thus, this overhead (in bits) is expressed as follows:

$$\text{storage} = \sum_{i=0}^{m-1} \left[\log_2 \left(\frac{3}{4} N(i) \right) \right]. \quad (43)$$

For say $m = 27$ (and $m' = 9$), we need to store the values in $\{3N(i)/4 \mid 0 \leq i \leq 26\}$ for encoding-decoding. This requires only 703-bit offline memory, where we achieve a normalized rate of 0.8125 by adopting Bridging Scheme III. The same rate is achieved in [30] for a content-balanced RLL code at length 8 and binary message length 13, but this technique is based on lookup tables, which incur higher complexity. Observe that all constrained coding techniques that are based on lookup tables incur higher storage overhead.

For $m = 17$, we need to store the values in $\{3N(i)/4 \mid 0 \leq i \leq 16\}$ for encoding-decoding. This requires 273-bit offline memory, where we achieve a normalized rate of 0.9167 and global disparity fraction 0.1 (equivalently, 45% – 55% GC-content) by adopting Bridging Scheme I for $K = 10$ (see Lemma 1). Alternatively, for $m = 33$, we need to store the values in $\{3N(i)/4 \mid 0 \leq i \leq 32\}$ for encoding-decoding. This requires 1057-bit offline memory, where we achieve a normalized rate of 0.9167 and global disparity fraction

$$\frac{p^{\text{global}}}{M} = \frac{33 + 2 \times 9 + 1}{9(33 + 3)} = 0.1605$$

(equivalently, 42% – 58% GC-content) by adopting Bridging Scheme II-A for $K = 9$ (see Remark 8 in Section V above). A rate of 0.9165 along with 45% – 55% GC-content range are achieved in [36], and their iterative encoding-decoding algorithms are based on 48-entry mapping tables.

Observe that if the DNA storage system can afford adders of higher sizes, our D-LOCO codes can offer notably higher rates along with notably lower global disparity fractions compared with the ones mentioned above.

The time complexity of the decoding (encoding) algorithm of D-LOCO codes is algorithmically linear with respect to the code length m , i.e., $O(m)$. As we perform additions (subtractions) of at most $2m$ -bit numbers at each iteration, implementation-wise we offer polynomial-time complexity, more precisely $O(m^2)$ (see Section V for encoding-decoding algorithms as well as Table III for adder sizes).

From (43), the storage overhead is $O(m^2)$. Note that a lookup table technique to design a constrained code of length m for DNA storage, where all valid sequences are stored, will have storage overhead of $O(4^m \cdot m)$, which is remarkably higher than $O(m^2)$.

Remark 9. Note that symbol contributions can also be computed offline and used as inputs to the adder, provided that storage overhead requirements allow that. This results in an algorithmically linear decoder where only a single addition of at most $2m$ -bit numbers is required at every iteration.

C. Reconfigurability

Just like all LOCO codes [24], D-LOCO codes are also easily reconfigurable. In particular, the same set of adders used to encode-decode a specific D-LOCO code can be reconfigured to encode-decode another D-LOCO code of different length and/or run-length constraint. Reconfigurability can play a pivotal role as the storage system ages and its performance gradually degrades. For instance, we may need to switch the run-length from ℓ to ℓ' , where $\ell' < \ell$, in order to address the need of an aging, more error-prone system in which, homopolymers of shorter lengths become detrimental. While this will result in a rate loss, it can maintain the same level of reliability with time. Such reconfiguration is achieved simply by changing the inputs of the adders, i.e., the cardinalities, from $\{N_D(i+1-\ell), \dots, N_D(i)\}$ to $\{N_D(i+1-\ell'), \dots, N_D(i)\}$ through multiplexers to find the symbol contribution $g_i(c_i)$. Note that the system allows this reconfiguration since the required size of the adders gets lower as we move from ℓ to $\ell' < \ell$.

Provided that storage overhead requirements allow it, one can also compute the contributions of each symbol offline for $\mathcal{D}_{m,\ell}$ and $\mathcal{D}_{m,\ell'}$ and switch from the former to the latter through multiplexers in order to find the index of a codeword via m additions (see Remark 9).

D. Error Propagation and Parallelism

In D-LOCO codes, error propagation does not occur from a codeword into another. Moderate message lengths are preferred, however, in order to mitigate any possible codeword-to-message error propagation (see also [26] and [24]). Furthermore, as they have fixed length, D-LOCO codes enable simultaneous encoding and decoding of different codewords within the complexity margin of the system, which can remarkably increase the speed of writing and reading.

E. Local Balancing

Definition 3. ([37, Definition 2]) A DNA sequence/stream of length M satisfies μ -prefix validity if for every prefix S of it,

i.e., for every subsequence consisting of the first S -many terms of it, $1 \leq S \leq M$, the disparity of S satisfies the inequality:

$$|p(S)| \leq 2\mu, \quad (44)$$

for non-negative μ .

Recall that for a DNA sequence consisting of codewords in $\mathcal{D}_{m,\ell}$ of odd length m , when

- Bridging Scheme I and
- our balancing procedure

are applied, its global disparity is in $[-m-1, m+1]$ (see Lemma 1). Therefore, (44) is satisfied for all prefixes of length divisible by $m+1$ if we set $\mu = \lceil (m+1)/2 \rceil = (m+1)/2$. For the remaining lengths, we first write S as a concatenation of S_0 and S_1 , where S_0 is the subsequence up to the last bridging symbol, and S_1 is the subsequence after the last bridging symbol. Without loss of generality, we consider the case of S_0 having a positive disparity. Then, μ can be set to

$$\lceil ((m+1) + (m-1)/2)/2 \rceil = \lceil (3m+1)/4 \rceil,$$

where the middle term $(m-1)/2$ is due to the maximum possible contribution of S_1 to the disparity of S given that the disparity of S_0 is at most $m+1$. Hence, such a DNA sequence satisfies $\lceil (3m+1)/4 \rceil$ -prefix validity for $m \geq 5$. This remains true if we adopt Bridging Schemes II-B or III instead (see Remark 8(ii) and Remark 8(iii)).

VIII. CONCLUSION

We introduced D-LOCO codes, a family of constrained codes designed for DNA data storage. D-LOCO codes are equipped with lexicographic ordering of codewords, which enables simple bijective mapping-demapping between an index set and the codebook. We derived the mathematical rule governing this mapping-demapping, which leads to systematic, reconfigurable, and low-complexity encoding and decoding algorithms. We discussed four bridging schemes, three of which guarantee single substitution error detection, and also studied the probability of missing errors. We showed how D-LOCO codes can be balanced, locally and globally, with minimal rate loss. D-LOCO codes are capacity-achieving, and they have remarkably high finite-length rates at affordable complexities even with error-detection and balancing features. Future work includes extending D-LOCO codes to forbid other detrimental patterns as well as combining constrained codes with error-correction codes for DNA data storage.

APPENDIX

In this appendix, we present a generalized version of D-LOCO codes, namely GD-LOCO codes, for any alphabet size.

Definition 4. For an integer $q \geq 2$, let $E = \{\Delta^r \mid 0 \leq r \leq q-1\}$ be a lexicographically-ordered alphabet with q elements such that $\Delta^r < \Delta^s$ for $0 \leq r < s \leq q-1$. For $\ell \geq 1$, the GD-LOCO code $\mathcal{GD}_{m,\ell}$ is defined as the set of all codewords of length m defined over E that do not contain runs of length exceeding ℓ .

We denote the cardinality of $\mathcal{GD}_{m,\ell}$ by $N(m)$ (instead of $N_{\text{GD}}(m, \ell)$) for the ease of notation.

Proposition 2. ([27, Equation 1]) The cardinality $N(m)$ of the GD-LOCO code $\mathcal{GD}_{m,\ell}$, where $\ell \geq 1$, satisfies the following recursive relation for $m \geq \ell$:

$$N(m) = (q-1) \sum_{i=1}^{\ell} N(m-i). \quad (45)$$

For $0 \leq m \leq \ell$,

$$N(0) \triangleq \frac{q}{q-1}, \text{ and } N(m) = q^m \text{ for } 1 \leq m \leq \ell.$$

Theorem 3. The encoding-decoding rule $g : \mathcal{GD}_{m,\ell} \rightarrow \{0, 1, \dots, N(m)-1\}$, for general $\ell \geq 1$, is as follows:

$$g(\mathbf{c}) = \frac{q-1}{q} \sum_{i=0}^{m-1} \sum_{j=1}^{\ell} \sum_{k=1}^j \left(\sum_{\Delta \in E^*} \delta_{i,k} \right) N(i+j-\ell), \quad (46)$$

where $N(0) \triangleq q/(q-1)$, $N(-1) \triangleq \dots \triangleq N(2-\ell) \triangleq N(1-\ell) \triangleq 0$, and $E^* = E \setminus \{\Delta^{q-1}\}$. Moreover, for all $\Pi > \Delta$ and for each $\Delta \in E \setminus \{\Delta^{q-1}\}$,

$$\begin{aligned} \delta_{i,1} &= 1 \text{ only if } c_i = \Pi \text{ and } c_{i+1} \neq \Delta, \\ \delta_{i,k} &= 1 \text{ only if } c_{i+k-1} \dots c_{i+1} c_i = \Delta^{k-1} \Pi \text{ and } c_{i+k} \neq \Delta, \end{aligned} \quad (47)$$

where $2 \leq k \leq \ell$ (these are $\ell-1$ statements), and in all other cases, $\delta_{i,k} = 0$ for $i \in \{1, 2, \dots, m\}$ and $k \in \{1, 2, \dots, \ell\}$.

Here, $\Pi = \Delta^s > \Delta = \Delta^r$, for some $0 \leq r < s \leq q-1$, is according to the lexicographic ordering rule, and $\delta = \delta^r$ in $\{\delta^0, \delta^1, \dots, \delta^{q-2}\}$ stands for the small letter corresponding to $\Delta = \Delta^r$ in $\{\Delta^0, \Delta^1, \dots, \Delta^{q-2}\}$.

Recall that $c_i \triangleq \zeta$ (a letter outside of the alphabet), for all $i > m-1$ (see Remark 1).

Remark 10. For $q = 2$, the alphabet $E = \{\Delta^0, \Delta^1\}$ can be defined as the Galois field $\text{GF}(2) = \{0, 1\}$. Lexicographically-ordered RLL (LO-RLL) codes can be designed as shown in [4]. For $\ell \geq 2$, define the difference vector \mathbf{v} of a GD-LOCO codeword \mathbf{c} in $\mathcal{GD}_{m,\ell}$ as $\mathbf{v} = [v_{m-2} v_{m-3} \dots v_0]$, with $v_i = c_{i+1} + c_i$ over $\text{GF}(2)$, for all $i \in \{0, 1, \dots, m-2\}$. Observe that for any codeword \mathbf{c} in $\mathcal{GD}_{m,\ell}$, its difference vector satisfies the $(0, \ell-1)$ RLL constraint. Moreover, each codeword in the $(0, \ell-1)$ LO-RLL code of length $m-1$ can be derived from a unique codeword among the GD-LOCO codewords starting with 0 from the left in $\mathcal{GD}_{m,\ell}$ by computing the difference vectors of all such codewords (the difference vectors of the remaining codewords will be duplicates). Consequently, the cardinality $N_{\text{RLL}}(m-1, \ell-1)$ of the $(0, \ell-1)$ LO-RLL code is given by:

$$N_{\text{RLL}}(m-1, \ell-1) = \frac{1}{2} N_{\text{GD}}(m, \ell). \quad (48)$$

This observation leads to a simple way of constructing and indexing $(0, \ell-1)$ LO-RLL codes via GD-LOCO codes with parameters $q = 2$ and ℓ .

Remark 11. For $q = 16$, consider the GD-LOCO code $\mathcal{GD}_{m,1}$ consisting of codewords with no identical consecutive symbols. Note that if we convert codewords in $\mathcal{GD}_{m,1}$ to 4-ary codewords, where each 16-ary symbol is mapped to a unique 2-tuple of 4-ary symbols, we obtain a set of 4-ary codewords

of length $2m$ that do not contain any patterns of the form $\Lambda_1\Lambda_2\Lambda_1\Lambda_2\Lambda_1$, where Λ_1 and Λ_2 can be the same symbol in the alphabet of size 4. If this alphabet is $\{A, T, G, C\}$, such 4-ary coding scheme allows runs of length at most 4 and also eliminates short tandem repeats such as AGAGAG.

ACKNOWLEDGMENT

The authors would like to thank Özge Simay Demirci and Selin Sönmez for their assistance in carrying out this research. Furthermore, they would like to thank the Guest Editor Prof. Eitan Yaakobi for his effective handling of the article, and thank the anonymous reviewers for their constructive comments.

REFERENCES

- [1] 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, May 2013.
- [2] J. J. Schwartz, C. Lee, and J. Shendure, "Accurate gene synthesis with tag-directed retrieval of sequence-verified DNA molecules," *Nat. Methods*, vol. 9, no. 9, pp. 913–915, Aug. 2012.
- [3] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, Jul. 1948.
- [4] D. T. Tang and R. L. Bahl, "Block codes for a class of constrained noiseless channels," *Inf. and Control*, vol. 17, no. 5, pp. 436–461, Dec. 1970.
- [5] P. A. Franaszek, "Sequence-state methods for run-length-limited coding," *IBM J. Res. Develop.*, vol. 14, no. 4, pp. 376–383, Jul. 1970.
- [6] R. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes—An application of symbolic dynamics to information theory," *IEEE Trans. Inf. Theory*, vol. 29, no. 1, pp. 5–22, Jan. 1983.
- [7] P. Siegel, "Recording codes for digital magnetic storage," *IEEE Trans. Magn.*, vol. 21, no. 5, pp. 1344–1349, Sep. 1985.
- [8] R. Karabed and P. H. Siegel, "Coding for higher-order partial-response channels," in *Proc. SPIE Int. Symp. Voice, Video, and Data Commun.*, M. R. Raghuvver, S. A. Dianat, S. W. McLaughlin, and M. Hassner, Eds., Philadelphia, PA, Oct. 1995, vol. 2605, pp. 115–126.
- [9] A. Sharov and R. M. Roth, "Two-dimensional constrained coding based on tiling," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1800–1807, Apr. 2010.
- [10] A. Kato and K. Zeger, "On the capacity of two-dimensional run-length constrained channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1527–1540, Jul. 1999.
- [11] B. Dabak, A. Hareedy, and R. Calderbank, "Non-binary constrained codes for two-dimensional magnetic recording," *IEEE Trans. Magn.*, vol. 56, no. 11, pp. 1–10, Nov. 2020.
- [12] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Device Lett.*, vol. 23, no. 5, pp. 264–266, May 2002.
- [13] V. Taranalli, H. Uchikawa, and P. H. Siegel, "Error analysis and inter-cell interference mitigation in multi-level cell flash memories," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, UK, Jun. 2015, pp. 271–276.
- [14] A. Hareedy, B. Dabak, and R. Calderbank, "Managing device lifecycle: Reconfigurable constrained codes for M/T/Q/P-LC Flash memories," *IEEE Trans. Inf. Theory*, vol. 67, no. 1, pp. 282–295, Jan. 2021.
- [15] A. Hareedy, S. Zheng, P. Siegel, and R. Calderbank, "Efficient constrained codes that enable page separation in modern Flash memories," *IEEE Trans. Commun.*, vol. 71, no. 12, pp. 6834–6848, Dec. 2023.
- [16] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2260–2299, Oct. 1998.
- [17] X. Li, S. Zhou, and L. Zou, "Design of DNA storage coding with enhanced constraints," *Entropy*, vol. 24, no. 8, pp. 1151, Aug. 2022.
- [18] R. Heckel, G. Mikutis, and R. N. Grass, "Characterization of the DNA data storage channel," *Sci. Rep.*, vol. 9, no. 9663, Jul. 2019, doi.org/10.1038/s41598-019-45832-6.
- [19] J. Centers, X. Tan, A. Hareedy, and R. Calderbank, "Power spectra of constrained codes with level-based signaling: Overcoming finite-length challenges," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 4971–4986, Aug. 2021.
- [20] T. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 73–77, Jan. 1973.
- [21] V. Braun and K. A. S. Immink, "An enumerative coding technique for DC-free runlength-limited sequences," *IEEE Trans. Commun.*, vol. 48, no. 12, pp. 2024–2031, Dec. 2000.
- [22] J. Gu and T. E. Fuja, "A new approach to constructing optimal block codes for runlength-limited channels," *IEEE Trans. Inf. Theory*, vol. 40, no. 3, pp. 774–785, May 1994.
- [23] I. F. Blake, "The enumeration of certain run length sequences," *Inf. and Control*, vol. 55, no. 1–3, pp. 222–237, Oct.–Dec. 1982.
- [24] A. Hareedy, B. Dabak, and R. Calderbank, "The secret arithmetic of patterns: A general method for designing constrained codes based on lexicographic indexing," *IEEE Trans. Inf. Theory*, vol. 68, no. 9, pp. 5747–5778, Sep. 2022.
- [25] A. Hareedy and R. Calderbank, "Asymmetric LOCO codes: Constrained codes for Flash memories," *Proc. 57th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Monticello, IL, USA, Sep. 2019, pp. 124–131.
- [26] A. Hareedy and R. Calderbank, "LOCO codes: Lexicographically-ordered constrained codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 6, pp. 3572–3589, Jun. 2020.
- [27] K. A. S. Immink and K. Cai, "Design of capacity-approaching constrained codes for DNA-based storage systems," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 224–227, Feb. 2018.
- [28] K. A. S. Immink and K. Cai, "Properties and constructions of constrained codes for DNA-based data storage," *IEEE Access*, vol. 8, pp. 49523–49531, Mar. 2020.
- [29] W. Song, K. Cai, M. Zhang, and C. Yuen, "Codes with run-length and GC-content constraints for DNA-based data storage," *IEEE Commun. Lett.*, vol. 22, no. 10, pp. 2004–2007, Oct. 2018.
- [30] Y. Wang, M. Noor-A-Rahim, E. Gunawan, Y. L. Guan, and C. L. Poh, "Construction of bio-constrained code for DNA data storage," *IEEE Commun. Lett.*, vol. 23, no. 6, pp. 963–966, Jun. 2019.
- [31] T. T. Nguyen, K. Cai, K. A. S. Immink, and H. M. Kiah, "Capacity-approaching constrained codes with error correction for DNA-based data storage," *IEEE Trans. Inf. Theory*, vol. 67, no. 8, pp. 5602–5613, Aug. 2021.
- [32] T. T. Nguyen, K. Cai, and P. H. Siegel, "Every bit counts: A new version of non-binary VT codes with more efficient encoder," *Proc. IEEE Int. Conf. Commun. (ICC)*, Rome, Italy, May–Jun. 2023, pp. 5477–5482.
- [33] X. Li, M. Chen, and H. Wu, "Multiple errors correction for position-limited DNA sequences with GC balance and no homopolymer for DNA-based data storage," *Brief. Bioinform.*, vol. 24, no. 1, Jan. 2023.
- [34] Z. Yan, C. Liang, and H. Wu, "A Segmented-edit error-correcting code with re-synchronization function for DNA-based storage systems," *IEEE Trans. Emerg. Top. Comput.*, vol. 11, no. 3, pp. 605–618, Jul.–Sep. 2023.
- [35] Z. Yan, G. Qu, and H. Wu, "A novel soft-in soft-out decoding algorithm for VT codes on multiple received DNA strands," *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Taipei, Taiwan, Jun. 2023, pp. 838–843.
- [36] S.-J. Park, Y. Lee, and J.-S. No, "Iterative coding scheme satisfying GC balance and run-length constraints for DNA storage with robustness to error propagation," *J. Commun. Netw.*, vol. 24, no. 3, pp. 283–291, Jun. 2022.
- [37] Y. Liu, X. He, and X. Tang, "Capacity-achieving constrained codes with GC-content and runlength limits for DNA storage," *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Espoo, Finland, Jun.–Jul. 2022, pp. 198–203.
- [38] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. N. Takahashi, S. Newman, H.-Y. Parker, C. Rashtchian, K. Stewart, G. Gupta, R. Carlson, J. Mulligan, D. Carmean, G. Seelig, L. Ceze, and K. Strauss, "Random access in large-scale DNA data storage," *Nat. Biotechnol.* vol. 36, no. 3, pp. 242–258, Mar. 2018.
- [39] O. Milenkovic, R. Gabrys, H. M. Kiah, and S. M. H. Tabatabaei Yazdi, "Exabytes in a test tube," *IEEE Spectr.*, vol. 55, no. 5, pp. 40–45, May 2018.

Canberk İrimağzı received the B.Sc. degree in Mathematics from Koç University, Turkey, and the M.A. degree in Mathematics from the University of Wisconsin-Madison, USA. He is currently pursuing the Ph.D. degree in Cryptography at the Institute of Applied Mathematics, Middle East Technical University (METU), Turkey. His current research interests include applied algebra, coding theory, and DNA data storage.

Yusuf Uslan is an undergraduate student at the Electrical and Electronics Engineering Department of Middle East Technical University (METU), Turkey. His research interests include coding theory, signal processing, and DNA data storage.

Ahmed Hareedy (Member, IEEE) is an Assistant Professor with the Department of Electrical and Electronics Engineering at Middle East Technical University (METU), Turkey. He is also an Affiliated Faculty Member with the Institute of Applied Mathematics at METU, Turkey. He is interested in questions in coding/information theory that are fundamental to opportunities created by the current, unparalleled access to data and computing. He received the Bachelor and M.S. degrees in Electronics and Communications Engineering from Cairo University, Egypt, in 2006 and 2011, respectively. He received the Ph.D. degree in Electrical and Computer Engineering from the University of California, Los Angeles (UCLA), USA, in 2018. He was a Postdoctoral Associate with the Department of Electrical and Computer

Engineering at Duke University, USA, between 2018 and 2021. He worked with Mentor Graphics Corporation (currently, Siemens EDA) between 2006 and 2014. He worked as an Error-Correction Coding Architect with Intel Corporation in the summers of 2015 and 2017.

Dr. Hareedy won the 2018–2019 Distinguished Ph.D. Dissertation Award in Signals and Systems from the Department of Electrical and Computer Engineering at UCLA. He is a recipient of the Best Paper Award from the 2015 IEEE Global Communications Conference (GLOBECOM), Selected Areas in Communications, Data Storage Track. He won the 2017–2018 Dissertation Year Fellowship (DYF) at UCLA. He won the 2016–2017 Electrical Engineering Henry Samueli Excellence in Teaching Award for teaching Probability and Statistics at UCLA. He is a recipient of the Memorable Paper Award from the 2018 Non-Volatile Memories Workshop (NVMW) in the area of devices, coding, and information theory. He is a recipient of the 2018–2019 Best Student Paper Award from the IEEE Data Storage Technical Committee (DSTC). He has been awarded the TÜBİTAK 2232-B International Fellowship for Early Stage Researchers in 2022. He is currently a Guest Editor of the Special Issue on Data Storage of the IEEE BITS THE INFORMATION THEORY MAGAZINE.