Correcting a Single Indel/Edit for DNA-Based Data Storage: Linear-Time Encoders and Order-Optimality

Kui Cai, *Senior Member, IEEE*, Yeow Meng Chee, *Senior Member, IEEE*, Ryan Gabrys, *Member, IEEE*, Han Mao Kiah, *Member, IEEE*, and Tuan Thanh Nguyen, *Member, IEEE*

***Abstract*— An indel refers to a single insertion or deletion, while an edit refers to a single insertion, deletion or substitution. In this article, we investigate codes that correct either a single indel or a single edit and provide linear-time algorithms that encode binary messages into these codes of length n. Over the quaternary alphabet, we provide two linear-time encoders. One corrects a single edit with $\lceil \log n \rceil + O(\log \log n)$ redundancy bits, while the other corrects a single indel with $\lceil \log n \rceil + 2$ redundant bits. These two encoders are *order-optimal*. The former encoder is the first known order-optimal encoder that corrects a single edit, while the latter encoder (that corrects a single indel) reduces the redundancy of the best known encoder of Tenengolts (1984) by at least four bits. Over the DNA alphabet, we impose an additional constraint: the GC-*balanced constraint* and require that exactly half of the symbols of any DNA codeword to be either C or G. In particular, via a modification of Knuth's balancing technique, we provide a linear-time map that translates binary messages into GC-balanced codewords and the resulting codebook is able to correct a single indel or a single edit. These are the first known constructions of GC-balanced codes that correct a single indel or a single edit.**

***Index Terms*— Error correction codes, DNA-based data storage, encoding, decoding, GC-balanced codes.**

I. Introduction

ADVANCES in synthesis and sequencing technologies have made DNA macromolecules an attractive medium for digital information storage. Besides being biochemically robust, DNA strands offer ultrahigh storage densities of 10^{15}-10^{20} bytes per gram of DNA, as demonstrated in recent experiments (see [32, Table 1]). These synthetic DNA strands may be stored *ex vivo* or *in vivo*. When the DNA strands are stored *ex vivo* or in a non-biological environment, code design takes into account the synthesising and sequencing platforms being used (see [33] for a survey). In contrast, when the DNA strands are stored *in vivo* or recombined with the DNA of a living organism, we design codes to correct errors due to the biological mutations [17].

Common to both environments are errors due to *insertion*, *deletion* and *substitution*. For example, Organick *et al.* recently stored 200MB of data in 13 million DNA strands and reported insertion, deletion and substitution rates to be 5.4×10^{-4}, 1.5×10^{-3} and 4.5×10^{-3}, respectively [22]. When DNA strands are stored *in vivo*, these errors are collectively termed as *point mutations* and occur during the process of DNA replication [8]. For convenience, we refer to a single insertion or deletion as an *indel*, and a single insertion, deletion or substitution as an *edit*. Given that current synthesis technologies produce strands of lengths 100 to 200 and given the low raw error rates reported by experiments [12], [22], we expect most DNA strands to be corrupted by at most one edit error. Therefore, in this work, we focus on codes that combat either a *single indel* or a *single edit* and provide efficient methods of encoding binary messages into these codes. Furthermore, we envision that the single-indel / edit-correcting codes proposed in this article to be used as inner codes in a concatenated coding scheme. Then the outer codes can be used to correct the blocks where more than one error occur in practice. Analysis of such concatenation schemes is beyond the scope of this article.

Now, to correct a single indel, we have the celebrated class of Varshamov-Tenengolts (VT) codes. While Varshamov and Tenengolts introduced the binary version to correct asymmetric errors [31], Levenshtein later provided a linear-time decoding algorithm to correct a single indel for the VT codes [19]. In the same paper, Levenshtein modified the VT construction to correct a single edit. In both constructions, the number of redundant bits is $\log n + O(1)$, where n is the length of a codeword. Unless otherwise stated, all logarithms are taken base two.

The VT construction partitions all binary words of length n into certain $n+1$ classes, where each class is a VT code. Curiously, even though efficient decoding was known since 1965, a linear-time algorithm to encode into a VT code was only

Manuscript received October 13, 2019; revised July 25, 2020; accepted November 9, 2020. Date of publication January 6, 2021; date of current version May 20, 2021. The work of Kui Cai and Tuan Thanh Nguyen was supported by the Singapore Ministry of Education Academic Research Fund Tier 2 under Grant MOE2016-T2-2-054. This article was presented in part at the 2019 IEEE International Symposium on Information Theory and in part at the 2020 IEEE International Conference on Acoustics, Speech, and Signal Processing. *(Corresponding author: Tuan Thanh Nguyen.)*

Kui Cai and Tuan Thanh Nguyen are with Science, Mathematics and Technology Cluster, the Singapore University of Technology and Design, Singapore 487372 (e-mail: cai_kui@sutd.edu.sg; tuanthanh_nguyen@sutd.edu.sg).

Yeow Meng Chee is with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 119077 (e-mail: pvocym@nus.edu.sg).

Ryan Gabrys is with the Spawar Systems Center, San Diego, CA 92152 USA (e-mail: ryan.gabrys@gmail.com).

Han Mao Kiah is with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371 (e-mail: hmkiah@ntu.edu.sg).

Communicated by O. Milenkovic, Guest Editor-in-Chief for the Special Issue: "From Deletion-Correction to Graph Reconstruction: In Memory of Vladimir I. Levenshtein."

Digital Object Identifier 10.1109/TIT.2021.3049627

proposed by Abdel-Ghaffar and Ferriera in 1998 [1]. In the same paper, Abdel-Ghaffar and Ferriera also adapted their linear-time encoding schemes to a more general class of codes called Constantin-Rao codes. Later, in 1999, Saowapa *et al.* adopted a similar approach to design a linear-time encoder for codes correcting a single edit [24].

A nonbinary version of the VT codes was proposed by Tenengolts [28], who also provided a linear-time method to correct a single indel. In the same paper, Tenengolts also provided an efficient encoder that corrects a single indel. For the quaternary alphabet, this encoder requires at least $\log n + 7$ bits for words of length $n \geqslant 20$. However, the codewords obtained from this encoder are not contained in a single non-binary VT code (see Section II-B for a discussion). Hence, recently, Abroshan *et al.* presented a systematic encoder that maps words into a single non-binary VT code [3]. Unfortunately, the redundancy of this encoder is $\lceil \log n \rceil(\log q + 1) + 2(\log q - 1)$, and when $q = 4$, the redundancy is $3\lceil \log n \rceil + 2$. To the best of our knowledge, there is no known efficient construction for q-ary codes (or even quaternary codes) that can correct a single edit.

To further reduce errors, we also impose certain weight constraints on the individual codewords. Specifically, the GC-*content* of a DNA string refers to the percentage of nucleotides that corresponds to G or C, and DNA strings with GC-content that are too high or too low are more prone to both synthesis and sequencing errors (see for example, [23], [34]). Therefore, most work use DNA strings whose GC-content is close to 50% as codewords and use randomizing techniques to encode binary message into the latter [22]. Recently, in addition to the GC-content constraints, Immink and Cai studied the homopolymer runlength constraint for DNA codewords [14]. Hence, in this work, in addition to correcting either a single indel or a single edit, we also provide linear-time encoders that map binary messages into codewords that have GC-content exactly 50%. To the best of our knowledge, no such codebooks are known prior to this work.

In summary, our broad objective is to provide practical quaternary codes that correct either a single indel or a single edit. In some instances, we also require the codewords to obey certain weight constraints. Besides minimizing the number of redundant bits, we also equip our codes with efficient encoders. Our specific contributions are as follows.

(A) In Section III, we focus on correcting a single indel. Instead of Tenengolts' quaternary codes, we investigate a class of binary codes by Levenshtein that corrects a burst of errors. With suitable modifications, we present a linear-time quaternary encoder that corrects a single indel with $\lceil \log n \rceil + 2$ bits of redundancy. This construction improves the encoder of Tenengolts [28] by reducing the redundancy by *at least four bits*. We then proceed to extend and generalize this construction so as to design efficient encoder for codes capable of correcting a burst of indels with $\log n + O(\log \log n)$ bits of redundancy. Here, a burst of indels refers to either a burst of deletions or a burst of insertions.

(B) In Section IV, we focus on quaternary codes that correct a single edit. Specifically, we design two classes of quaternary codes. The first class of codes incurs $2\lceil \log n \rceil + 2$ bits of redundancy, while the second class of codes incurs only $\log n + O(\log \log n)$ bits of redundancy and is thus order-optimal. Even though the former is not order-optimal, it outperforms the latter class when $n \leqslant 512$. In Section IV-C, we study a type of edits specific to the DNA storage channel and provide an efficient encoder that correct such edits with $\log n + \log \log n + O(1)$ bits of redundancy.

(C) In Section V, we impose the GC-content constraints to words in our codebook. Specifically, we encode binary messages to GC-balanced codewords. Via a modification of Knuth's balancing method, we obtain linear-time GC-balanced single indel/edit-correcting encoders.

We first go through certain notation and define the problem. For the convenience of the reader, relevant notation and terminology referred to throughout the paper is summarized in Table I.

II. Preliminary

Let Σ denote an *alphabet* of size q. For any positive integer $m < n$, we let $[m, n]$ denote the set $\{m, m+1, \ldots, n\}$ and $[n] = [1, n]$.

Given two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$, we let $\boldsymbol{xy}$ denote the *concatenation* of the two sequences. In the special case where $\boldsymbol{x}, \boldsymbol{y} \in \Sigma^n$, we use $\boldsymbol{x}||\boldsymbol{y}$ to denote their *interleaved sequence* $x_1y_1x_2y_2\ldots x_ny_n$. For a subset $I = \{i_1, i_2, \ldots, i_k\}$ of coordinates, we use $\boldsymbol{x}|_I$ to denote the vector $x_{i_1}x_{i_2}\ldots x_{i_k}$.

Let $\boldsymbol{x} \in \Sigma^n$. We are interested in the following *error balls*:

$$\mathcal{B}^{\text{indel}}(\boldsymbol{x}) \triangleq \{\boldsymbol{x}\} \cup \{\boldsymbol{y} : \boldsymbol{y} \text{ is obtained from } \boldsymbol{x} \text{ via a single insertion or single deletion}\},$$

$$\mathcal{B}^{\text{edit}}(\boldsymbol{x}) \triangleq \{\boldsymbol{x}\} \cup \{\boldsymbol{y} : \boldsymbol{y} \text{ is obtained from } \boldsymbol{x} \text{ via a single insertion or single deletion, or single substitution}\}.$$

Observe that when $\boldsymbol{x} \in \Sigma^n$, both $\mathcal{B}^{\text{indel}}(\boldsymbol{x})$ and $\mathcal{B}^{\text{edit}}(\boldsymbol{x})$ are subsets of $\Sigma^{n-1} \cup \Sigma^n \cup \Sigma^{n+1}$. Hence, for convenience, we use Σ^{n*} to denote the set $\Sigma^{n-1} \cup \Sigma^n \cup \Sigma^{n+1}$.

Let $\mathcal{C} \subseteq \Sigma^n$. We say that $\mathcal{C}$ *corrects a single indel* if $\mathcal{B}^{\text{indel}}(\boldsymbol{x}) \cap \mathcal{B}^{\text{indel}}(\boldsymbol{y}) = \varnothing$ for all distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$. Similarly, $\mathcal{C}$ *corrects a single edit* if $\mathcal{B}^{\text{edit}}(\boldsymbol{x}) \cap \mathcal{B}^{\text{edit}}(\boldsymbol{y}) = \varnothing$ for all distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$. In this work, not only are we interested in constructing large codes that correct a single indel or edit, we desire efficient encoders that map binary messages into these codes.

Definition 1: The map $\text{Enc} : \{0,1\}^m \to \Sigma^n$ is a *single-indel-correcting encoder* if there exists a *decoder* map $\text{Dec} : \Sigma^{n*} \to \{0,1\}^m$ such that the following hold.

(i) For all $\boldsymbol{x} \in \{0,1\}^m$, we have $\text{Dec} \circ \text{Enc}(\boldsymbol{x}) = \boldsymbol{x}$.

(ii) If $\boldsymbol{c} = \text{Enc}(\boldsymbol{x})$ and $\boldsymbol{y} \in \mathcal{B}^{\text{indel}}(\boldsymbol{c})$, then $\text{Dec}(\boldsymbol{y}) = \boldsymbol{x}$.

Hence, we have that the code $\mathcal{C} = \{\boldsymbol{c} : \boldsymbol{c} = \text{Enc}(\boldsymbol{x}), \boldsymbol{x} \in \{0,1\}^m\}$ corrects a single indel and $|\mathcal{C}| = 2^m$. The *redundancy of the encoder* is measured by the value $n \log q - m$ (in bits). A *single-edit-correcting encoder* is similarly defined.

Therefore, our design objectives for a single-indel-correcting or single-edit-correcting encoder are as follows.

- The redundancy is $K \log n + o(\log n)$, where K is a constant to be minimized. When $K = 1$, we say that the encoder is *order-optimal*.

TABLE I
NOTATION SUMMARY

Notation	Description
Σ	alphabet of size q
Σ_4	quaternary alphabet, i.e. $q = 4, \Sigma_4 = \{0,1,2,3\}$
$\mathcal{D}$	DNA alphabet, $\mathcal{D} = \{\mathtt{A},\mathtt{T},\mathtt{C},\mathtt{G}\}$
$\boldsymbol{xy}$	the concatenation of two sequences
$\boldsymbol{x}\|\|\boldsymbol{y}$	the interleaved sequence
$\boldsymbol{\sigma}, \boldsymbol{U_\sigma}, \boldsymbol{L_\sigma}$	a DNA sequence $\boldsymbol{\sigma}$, the upper sequence of $\boldsymbol{\sigma}$, and the lower sequence of $\boldsymbol{\sigma}$
Ψ	the one-to-one map that converts a DNA sequence to a binary sequence
$\mathrm{Syn}(\boldsymbol{x})$	the syndrome of a sequence $\boldsymbol{x}$
$\mathrm{Rsyn}(\boldsymbol{x})$	the run-syndrome of a sequence $\boldsymbol{x}$
indel	single insertion or single deletion
edit	single insertion, or single deletion, or single substitution
b-burst-indel	b consecutive insertions or b consecutive deletions
b-burst-edit	b consecutive insertions, or b consecutive deletions, or b consecutive substitutions
$\mathcal{B}^{\mathrm{indel}}(\boldsymbol{x})$	the set of words that can be obtained from $\boldsymbol{x}$ via at most a single indel
$\mathcal{B}^{\mathrm{edit}}(\boldsymbol{x})$	the set of words that can be obtained from $\boldsymbol{x}$ via at most a single edit
$\mathcal{B}^{\mathrm{indel}}_{b-\mathrm{burst}}(\boldsymbol{x})$	the set of words that can be obtained from $\boldsymbol{x}$ via a b-burst-indel
$\mathcal{B}^{\mathrm{edit}}_{b-\mathrm{burst}}(\boldsymbol{x})$	the set of words that can be obtained from $\boldsymbol{x}$ via a b-burst-edit
$\mathrm{Bal}_k(n)$	the set of quaternary words of length n that is k-sum-balanced

Code and Encoder / Decoder	Description	Redundancy	Remark
$\mathrm{VT}_a(n)$	binary Varshamov-Tenengolts code that corrects a single indel		Section II-B
$\mathrm{L}_a(n)$	binary Levenshtein code that corrects a single edit		Section II-B
$\textsc{Enc}_{\mathbb{L}}, \textsc{Dec}_{\mathbb{L}}$	encoder and decoder for $\mathrm{L}_a(n)$	$\lceil \log n \rceil + 1$ bits	Section II-C
$\mathrm{T}_{a,b}(n;q)$	nonbinary VT code that corrects a single indel		Section II-B
$\mathrm{L}^{\mathrm{burst}}_a(n)$	binary Levenshtein code that corrects a 2-burst-indel		Section III-A
$\mathcal{C}_a(n)$	DNA code that corrects a single indel		Section III-A
$\textsc{Enc}_{\mathbb{I}}, \textsc{Dec}_{\mathbb{I}}$	encoder and decoder for $\mathcal{C}_a(n)$	$\lceil \log n \rceil + 2$ bits	Section III-B
$\mathrm{SVT}_{c,d,P}(n)$	Shifted-VT code that correct a single indel/edit provided that the error is located within P consecutive positions		Section III-C
$\textsc{Enc}_{SVT}$	encoder for $\mathrm{SVT}_{c,d,P}(n)$	$\lceil \log P \rceil + 2$ bits	Section III-C
$\textsc{Enc}_{RLL}$	encoder that outputs a binary sequence that the longest run is at most $\lceil \log n \rceil + 3$	one bit	Section III-C
$\mathcal{C}^{\mathrm{indel}}_{b-\mathrm{burst}}(n, P, r; a, c, d)$	DNA code that corrects a b-burst-indel		Section III-C
$\textsc{Enc}^{\mathrm{indel}}_{b-\mathrm{burst}}$	encoder for $\mathcal{C}^{\mathrm{indel}}_{b-\mathrm{burst}}(n, P, r; a, c, d)$	$\log n + O(\log \log n)$ bits	Section III-C
$\textsc{Enc}^{A}_{\mathbb{E}}$	encoder for DNA codes that corrects a single edit	$2\lceil \log n \rceil + 2$ bits	Section IV-A
$\mathcal{C}^{B}(n; a, b, c, d)$	order-optimal DNA code that corrects a single general edit		Section IV-B
$\mathcal{C}^{\mathrm{nt}}_{a,b,c}(n, r, P)$	order-optimal DNA code that corrects a single nucleotide edit		Section IV-C
$\textsc{Enc}^{\mathrm{nt}}_{\mathbb{E}}, \textsc{Dec}^{\mathrm{nt}}_{\mathbb{E}}$	the encoder and decoder for $\mathcal{C}^{\mathrm{nt}}_{a,b,c}(n, r, P)$	$\log n + O(\log \log n)$ bits	Section IV-C
$\textsc{Enc}_{\mathtt{GC}}, \textsc{Dec}_{\mathtt{GC}}$	encoder and decoder for $\mathtt{GC}$-balanced DNA code that corrects a single edit	$3\lceil \log n \rceil + 2$ bits	Section V

- The encoder $\textsc{Enc}$ can be computed in time $O(n)$.
- The decoder $\textsc{Dec}$ can be computed in time $O(n)$.

A. DNA Alphabet

When $q = 4$, we denote the alphabet by $\mathcal{D} = \{\mathtt{A},\mathtt{T},\mathtt{C},\mathtt{G}\}$ and consider the following one-to-one correspondence between $\mathcal{D}$ and two-bit sequences:

$$\mathtt{A} \leftrightarrow 00, \quad \mathtt{T} \leftrightarrow 01, \quad \mathtt{C} \leftrightarrow 10, \quad \mathtt{G} \leftrightarrow 11.$$

Therefore, given a sequence $\boldsymbol{\sigma} \in \mathcal{D}^n$, we have a corresponding binary sequence $\boldsymbol{x} \in \{0,1\}^{2n}$ and we write $\boldsymbol{x} = \Psi(\boldsymbol{\sigma})$.

Let n be even. We say that $\boldsymbol{\sigma} \in \mathcal{D}^n$ is $\mathtt{GC}$-*balanced* if the number of symbols in $\boldsymbol{\sigma}$ that correspond to $\mathtt{C}$ and $\mathtt{G}$ is $n/2$. On the other hand, we say that $\boldsymbol{x} \in \{0,1\}^n$ is *balanced* if the number of ones in $\boldsymbol{x}$ is $n/2$. For DNA-based storage, we are interested in codewords that are $\mathtt{GC}$-balanced.

Definition 2: A single-indel-correcting encoder $\textsc{Enc} : \{0,1\}^m \to \mathcal{D}^n$ is a $\mathtt{GC}$-*balanced single-indel-correcting encoder* if $\textsc{Enc}(\boldsymbol{x})$ is $\mathtt{GC}$-balanced for all $\boldsymbol{x} \in \{0,1\}^m$. A $\mathtt{GC}$-*balanced single-edit-correcting encoder* is similarly defined.

Given $\boldsymbol{\sigma} \in \mathcal{D}^n$, let $\boldsymbol{x} = \Psi(\boldsymbol{\sigma}) \in \{0,1\}^{2n}$ and we set $\boldsymbol{U_\sigma} = x_1x_3\cdots x_{2n-1}$ and $\boldsymbol{L_\sigma} = x_2x_4\cdots x_{2n}$. In other words, $\boldsymbol{\sigma} = \Psi^{-1}(\boldsymbol{U_\sigma}\|\boldsymbol{L_\sigma})$. We refer to $\boldsymbol{U_\sigma}$ and $\boldsymbol{L_\sigma}$ as the *upper sequence* and *lower sequence* of $\boldsymbol{\sigma}$, respectively. The following example demonstrates the relation between $\boldsymbol{\sigma}$, $\boldsymbol{U_\sigma}$ and $\boldsymbol{L_\sigma}$.

Example 1: Suppose that $\boldsymbol{\sigma} = \mathtt{ACAGTG}$ and we check that $\boldsymbol{\sigma}$ is $\mathtt{GC}$-balanced. Now, $\boldsymbol{x} \triangleq \Psi(\boldsymbol{\sigma}) = 001000110111$ and we write $\boldsymbol{U_\sigma}$ and $\boldsymbol{L_\sigma}$ as follows.

$\boldsymbol{\sigma}$	A	C	A	G	T	G
$\boldsymbol{U_\sigma}$	0	1	0	1	0	1
$\boldsymbol{L_\sigma}$	0	0	0	1	1	1

We make certain observations on $\boldsymbol{\sigma}$, $\boldsymbol{U_\sigma}$ and $\boldsymbol{L_\sigma}$.

Proposition 1: Let $\boldsymbol{\sigma} \in \mathcal{D}^n$. Then the following are true.

(a) $\boldsymbol{\sigma}$ is $\mathtt{GC}$-balanced if and only if $\boldsymbol{U_\sigma}$ is balanced.
(b) $\boldsymbol{\sigma'} \in \mathcal{B}^{\mathrm{indel}}(\boldsymbol{\sigma})$ implies that $\boldsymbol{U_{\sigma'}} \in \mathcal{B}^{\mathrm{indel}}(\boldsymbol{U_\sigma})$ and $\boldsymbol{L_{\sigma'}} \in \mathcal{B}^{\mathrm{indel}}(\boldsymbol{L_\sigma})$.

(c) $\boldsymbol{\sigma}' \in \mathcal{B}^{\rm edit}(\boldsymbol{\sigma})$ implies that $\boldsymbol{U}_{\boldsymbol{\sigma}'} \in \mathcal{B}^{\rm edit}(\boldsymbol{U}_{\boldsymbol{\sigma}})$ and $\boldsymbol{L}_{\boldsymbol{\sigma}'} \in \mathcal{B}^{\rm edit}(\boldsymbol{L}_{\boldsymbol{\sigma}})$.

Remark 1: The statement in Proposition 1 can be made stronger. Suppose that there is an indel at position i of $\boldsymbol{\sigma}$. Then there is exactly one indel at the same position i in both upper and lower sequences of σ. For example, consider $\boldsymbol{\sigma} = \texttt{ACAGTG}$ as in Example 1. If the third nucleotide $\texttt{A}$ is deleted, we obtain $\boldsymbol{\sigma}' = \texttt{ACGTG}$ and hence, $\boldsymbol{U}_{\boldsymbol{\sigma}'} = 01101$ and $\boldsymbol{L}_{\boldsymbol{\sigma}'} = 00111$. Furthermore, we observe that $\Psi(\boldsymbol{\sigma}) = \boldsymbol{U}_{\boldsymbol{\sigma}}||\boldsymbol{L}_{\boldsymbol{\sigma}}$ suffers a *burst of deletions of length two*. In our example, $\Psi(\boldsymbol{\sigma}) = 001000110111$ while $\Psi(\boldsymbol{\sigma}') = 0010110111$. In Section IV, we make use of this observation to reduce the redundancy of our encoders.

B. Previous Works

The *binary VT syndrome* of a binary sequence $\boldsymbol{x} \in \{0,1\}^n$ is defined to be $\mathrm{Syn}(\boldsymbol{x}) = \sum_{i=1}^n i x_i$.

For $a \in \mathbb{Z}_{n+1}$, let

$$\mathrm{VT}_a(n) = \{\boldsymbol{x} \in \{0,1\}^n : \mathrm{Syn}(\boldsymbol{x}) = a \pmod{n+1}\}. \quad (1)$$

Then $\mathrm{VT}_a(n)$ form the family of binary codes known as the *Varshamov-Tenengolts codes* [19]. These codes can correct a single indel and Levenshtein later provided a linear-time decoding algorithm [19]. For any n, we know that there exists $a \in \mathbb{Z}_{n+1}$ such that $\mathrm{VT}_a(n)$ has at least $2^n/(n+1)$ codewords. However, the first known linear-time encoder that maps binary messages into $\mathrm{VT}_a(n)$ was only described in 1998, when Abdel-Ghaffar and Ferreira gave a linear-time systematic encoder with redundancy $\lceil \log(n+1) \rceil$.

To also correct a substitution, Levenshtein [19] constructed the following code

$$\mathrm{L}_a(n) = \{\boldsymbol{x} \in \{0,1\}^n : \mathrm{Syn}(\boldsymbol{x}) = a \pmod{2n}\}, \quad (2)$$

and provided a decoder that corrects a single edit. In 1999, Saowapa *et al.* [24] gave a linear-time systematic encoder with redundancy $\lceil \log n \rceil + 1$. The encoding method is crucial in the construction of our subsequent encoders throughout this article. For ease of exposition, we review the method in Subsection II-C, and refer it as the **Levenshtein-encoder**, or **Encoder** $\mathbb{L}$.

Theorem 1 (Levenshtein [19] and Saowapa et al. [24]): Let $\mathrm{L}_a(n)$ be as defined in (2). There exists a linear-time decoding algorithm $\mathrm{DEC}_a^L : \{0,1\}^{n*} \to \mathrm{L}_a(n)$ such that the following hold. If $\boldsymbol{c} \in \mathrm{L}_a(n)$ and $\boldsymbol{y} \in \mathcal{B}^{\rm edit}(\boldsymbol{c})$, then $\mathrm{DEC}_a^L(\boldsymbol{y}) = \boldsymbol{c}$.

In 1984, Tenengolts [28] generalized the binary VT codes to nonbinary ones. Tenengolts defined the *signature* of a q-ary vector $\boldsymbol{x}$ of length n to be the binary vector $\alpha(\boldsymbol{x})$ of length $n-1$, where $\alpha(x)_i = 1$ if $x_{i+1} \geq x_i$, and 0 otherwise, for $i \in [n-1]$. For $a \in \mathbb{Z}_n$ and $b \in \mathbb{Z}_q$, set

$$\mathrm{T}_{a,b}(n;q) \triangleq \Big\{ \boldsymbol{x} \in \mathbb{Z}_q^n : \ \alpha(\boldsymbol{x}) \in \mathrm{VT}_a(n-1) \text{ and } \sum_{i=1}^n x_i = b \pmod{q} \Big\}.$$

Then Tenengolts showed that $T_{a,b}(n;q)$ corrects a single indel and there exists a and b such that the size of $\mathrm{T}_{a,b}(n;q)$ is at least $q^n/(qn)$. These codes are known to be order-optimal [16], [28].

Surprisingly, to also correct a single substitution, i.e. to correct a single edit, it is not straightforward as in the binary case. One possible strategy is to adapt Levenshtein's method by changing the modulo value in Tenengolts' nonbinary single-deletion correcting codes [1]. Unfortunately, this is not possible to adopt this strategy for Tenengolts' codes. The reason is that it is possible for two distinct words that differ in a single position to share the same signature. For example, consider the ternary words $\boldsymbol{x} = (2,2,2,1)$ and $\boldsymbol{x}' = (2,2,2,0)$. Their signatures are both $\alpha(\boldsymbol{x}) = \alpha(\boldsymbol{x}') = (1,1,0)$ and thus, any syndromes computed based on their signatures result in the same value. Hence, a novel strategy is required and we provide one in Section IV-B.

In the same paper, Tenengolts also provided a systematic q-ary single-indel-encoder with redundancy $\log n + C_q$, where n is the length of a codeword and C_q is independent of n. When $q = 4$, we have that $7 \leqslant C_4 \leqslant 10$ for $n \geqslant 20$. However, for this encoder, the codewords do not belong to $\mathrm{T}_{a,b}(n;q)$ for some fixed values of a and b. Recently, Abroshan *et al.* provided a method to systematically encode q-ary messages into $\mathrm{T}_{a,b}(n;q)$ [3]. However, the redundancy of such encoder is much larger compared with Tenengolts' work. Specifically, the encoder of Abroshan *et al.* [3] uses at least $(\log q + 1)\lceil \log n \rceil + 2(\log q - 1)$ bits of redundancy and particularly, when $q = 4$, the redundancy is $3\lceil \log n \rceil + 2$.

Note that to correct a single indel, it is not necessary that all codewords to belong to the same coset $\mathrm{T}_{a,b}(n;q)$. Nevertheless, when the words share the same VT parameters, Abroshan *et al.* [3] demonstrated that these codes can be adapted to correct multiple errors. This is in the context of *segmented edits* [3], [21].

C. Levenshtein Encoder

Recall the definition of $\mathrm{L}_a(n)$ in (2) and recall that $\mathrm{L}_a(n)$ is a binary code that can correct a single edit. For completeness, we summarize the systematic encoder for $\mathrm{L}_a(n)$, proposed by Saowapa *et al.* [24], as follows.

Encoder $\mathbb{L}$. Given n, set $t \triangleq \lceil \log n \rceil$ and $m \triangleq n - t - 1$.

INPUT: $\boldsymbol{x} \in \{0,1\}^m$

OUTPUT: $\boldsymbol{c} \triangleq \mathrm{ENC}_{\mathbb{L}}(\boldsymbol{x}) \in \mathrm{L}_a(n)$

(I) Set $S \triangleq \{2^{j-1} : j \in [t]\} \cup \{n\}$ and $I \triangleq [n] \setminus S$.

(II) Consider $\boldsymbol{c}' \in \{0,1\}^n$, where $\boldsymbol{c}'|_I = \boldsymbol{x}$ and $\boldsymbol{c}'|_S = 0$. Compute the difference $d' \triangleq a - \mathrm{Syn}(\boldsymbol{c}') \pmod{2n}$. In the next step, we modify $\boldsymbol{c}'$ to obtain a codeword $\boldsymbol{c}$ with $\mathrm{Syn}(\boldsymbol{c}) = a \pmod{2n}$.

(III) We have the following two cases.

- Suppose that $d' < n$. Let $y_{t-1} \ldots y_1 y_0$ be the binary representation of d'. In other words, $d' = \sum_{i=0}^{t-1} y_i 2^i$. Then we set $c_{2^{j-1}} = y_{j-1}$ for $j \in [t]$, $c_n = 0$ and $\mathbf{c}|_I = \boldsymbol{c}'|_I$ to obtain $\boldsymbol{c}$.
- Suppose that $n \leq d' < 2n$. We now compute the difference $d'' \triangleq d' - n \pmod{2n}$ and hence, $d'' <$

[1]In the binary case, Levenshtein increased the modulo value from $(n+1)$ to modulo $2n$ to correct a single substitution.

n. Consequently, the binary representation of d'' is of length $t = \lceil \log n \rceil$ and let it be $y_{t-1} \dots y_1 y_0$. As before, we set $c_{2^{j-1}} = y_{j-1}$ for $j \in [t]$, $c_n = 1$ and $\boldsymbol{c}|_I = \boldsymbol{c}'|_I$ to obtain $\boldsymbol{c}$.

We illustrate Encoder $\mathbb{L}$ via an example.

Example 2: Consider $n = 10$ and $a = 0$. Then $t = 4$ and $m = 5$. Suppose that the message is $\boldsymbol{x} = 11011$ and we compute $\textsc{Enc}_{\mathbb{L}}(\boldsymbol{x}) \triangleq \boldsymbol{c} = c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} \in \mathrm{L}_0(10)$.

(I) Set $S = \{1, 2, 4, 8, 10\}$ and $I = \{3, 5, 6, 7, 9\}$.

(II) Then we set $\boldsymbol{c}'|_I = 11011$ to obtain $\boldsymbol{c}' = 0010101010$. We then compute $d' = a - \mathrm{Syn}(\boldsymbol{c}') = 16 \pmod{20}$.

(III) Since $d' > 10$, we compute $d'' = d' - 10 = 6 \pmod{20}$. The binary representation of 6 is 0110. Therefore, we set $c_1 = 0$, $c_2 = 1$, $c_4 = 1$, $c_8 = 0$. Since $d' > 10$, we set $c_{10} = 1$. In summary, $\boldsymbol{c} = 0111101011$. We can verify that $\mathrm{Syn}(\boldsymbol{c}) = 0 \pmod{20}$.

For completeness, we summarize the corresponding decoder for $\mathrm{L}_a(n)$ as follows.

Decoder $\mathbb{L}$. For any n, set $m = n - \lceil \log n \rceil - 1$.

Input: $\boldsymbol{y} \in \{0,1\}^{n*}$

Output: $\boldsymbol{x} = \textsc{Dec}_{\mathbb{L}}(\boldsymbol{y}) \in \{0,1\}^m$

(I) Using Theorem 1, set $\boldsymbol{c} \triangleq \textsc{Dec}_a^L(\boldsymbol{y})$.

(II) Set $\boldsymbol{x} \triangleq \boldsymbol{c}|_I$, where I is defined by Encoder $\mathbb{L}$.

It is not hard to use Encoder $\mathbb{L}$ to construct an efficient encoder for DNA alphabet and the output codewords can correct a single edit. For any DNA strand $\boldsymbol{\sigma}$, we can use Encoder $\mathbb{L}$ to encode the upper sequence $\boldsymbol{U}_{\boldsymbol{\sigma}}$ and lower sequence $\boldsymbol{L}_{\boldsymbol{\sigma}}$ into $\mathrm{L}_a(n)$. If $\boldsymbol{U}_{\boldsymbol{\sigma}}$ and $\boldsymbol{L}_{\boldsymbol{\sigma}}$ can correct a single edit, according to Proposition 1, $\boldsymbol{\sigma}$ can correct a single edit. This construction costs $2\lceil \log n \rceil + 2$ bits of redundancy.

To correct a single indel, we can modify Encoder $\mathbb{L}$ to lower the redundancy to $\lceil \log n \rceil + 2$ bits when $q = 4$.

III. Encoders Correcting a Single Indel

In this section, we design efficient encoders for non-binary codes correcting a single indel. As mentioned earlier, the Tenengolts' codes correct a single indel for alphabet size $q > 2$ and recently, Abroshan *et al.* presented a systematic encoder that maps words into these codes [3]. Their encoder was built on the framework proposed in [1] in the context of Constantin-Rao codes. However, adapting the methods in [1] for the non-binary signatures is not straightforward. Hence, the redundancy of the encoder proposed by Abroshan *et al.* is at least $\lceil \log n \rceil(\log q + 1) + 2(\log q - 1)$. For the specific case when $q = 4$, the redundancy is $3\lceil \log n \rceil + 2$ which exceeds $\log n$ by a significant amount.

To construct an order-optimal linear-time encoder, we look at another code. In Section III-A, instead of Tenengolts' quaternary codes, we investigate a class of binary codes proposed by Levenshtein to correct a single burst of deletions. We then show that these binary codes can be transformed into quaternary single-indel-correcting codes and provide a corresponding linear-time encoder in Section III-B. In Section III-C, we continue our investigation and provide encoders that correct single bursts of indels.

A. Code Construction

Recall that in Remark 1, we observed that when an indel occurs in $\boldsymbol{\sigma} \in \mathcal{D}^n$, the binary sequence $\Psi(\boldsymbol{\sigma})$ has a burst of indels of length two. In other words, we are interested in binary codes that correct a single burst of indels of length two. To do so, we have the following construction by Levenshtein [19].

For $\boldsymbol{x} \in \{0,1\}^n$, we write $\boldsymbol{x}$ as the concatenation of s substrings $\boldsymbol{x} = \boldsymbol{u}_0 \boldsymbol{u}_1 \dots \boldsymbol{u}_{s-1}$, where each substring $\boldsymbol{u}_i$ contains identical bits, while substrings $\boldsymbol{u}_i$ and $\boldsymbol{u}_{i+1}$ contain different bits. Each substring $\boldsymbol{u}_i$ is also known as a *run* in $\boldsymbol{x}$. Let r_i be the length of the run $\boldsymbol{u}_i$. The *run-syndrome* of the binary word $\boldsymbol{x}$, denoted by $\mathrm{Rsyn}(\boldsymbol{x})$, is defined as follows.

$$\mathrm{Rsyn}(\boldsymbol{x}) = \sum_{i=1}^{s-1} i r_i. \tag{3}$$

Example 3: The word 0010110 has five runs, namely, $\boldsymbol{u}_0 = 00$, $\boldsymbol{u}_1 = 1$, $\boldsymbol{u}_2 = 0$, $\boldsymbol{u}_3 = 11$ and $\boldsymbol{u}_4 = 0$. Hence, $r_1 = r_2 = r_4 = 1$, $r_0 = r_3 = 2$ and $\mathrm{Rsyn}(\boldsymbol{x}) = 13$.

Theorem 2 (Levenshtein [20]): For $a \in \mathbb{Z}_{2n}$, set

$$\mathrm{L}_a^{\mathrm{burst}}(n) \triangleq \{\boldsymbol{x} \in \{0,1\}^n : \mathrm{Rsyn}(0\boldsymbol{x}) = a \pmod{2n}\}. \tag{4}$$

Then code $\mathrm{L}_a^{\mathrm{burst}}(n)$ can correct a burst of indel of length two. Furthermore, there exists a linear-time algorithm $\textsc{Dec}_a^{\mathrm{burst}}$ such that $\textsc{Dec}_a^{\mathrm{burst}}(\boldsymbol{y}) = \boldsymbol{c}$ for all $\boldsymbol{y} \in \mathcal{B}^{\mathrm{burst}}(\boldsymbol{c})$ and $\boldsymbol{c} \in \mathrm{L}_a^{\mathrm{burst}}(n)$. Here, $\mathcal{B}^{\mathrm{burst}}(\boldsymbol{c})$ refers to the error ball with respect to a single burst of indel of length two centered at $\boldsymbol{c}$.

Using this family of codes, we have a code over $\mathcal{D}$ that corrects a single indel. For $a \in \mathbb{Z}_{2n}$, set

$$\mathcal{C}_a(n) \triangleq \left\{\Psi^{-1}(\boldsymbol{c}) : \boldsymbol{c} \in \mathrm{L}_a^{\mathrm{burst}}(2n)\right\}. \tag{5}$$

To design a linear-time encoder for $\mathcal{C}_a(n)$, we make use of the following relation between run-syndrome and VT-syndrome of a binary word.

Lemma 1 (Levenshtein [20]): Define $\Phi : \{0,1\}^n \to \{0,1\}^n$ such that

$$\Phi(\boldsymbol{x})_i = \begin{cases} x_i + x_{i+1} \pmod 2 & \text{when } i \in [n-1], \\ x_n & \text{when } i = n. \end{cases} \tag{6}$$

Then Φ is an one-to-one map, and we have that

$$\mathrm{Rsyn}(0\boldsymbol{x}) = -\mathrm{Syn}(\Phi(\boldsymbol{x})) \pmod{2n}. \tag{7}$$

Example 4 (Example 3 Continued): Consider $\boldsymbol{x} = 010110$. We have $\Phi(\boldsymbol{x}) = 111010$ and $-\mathrm{Syn}(\Phi(\boldsymbol{x})) = -11 = 1 \pmod{12}$. On the other hand, $0\boldsymbol{x} = 0010110$ and indeed, $\mathrm{Rsyn}(0\boldsymbol{x}) = 13 = 1 \pmod{12}$.

B. Efficient Encoder Correcting a Single Indel

We now present an efficient method to translate binary sequences into $\mathcal{C}_a(n)$ and hence, obtain a linear-time single-indel-correcting encoder over $\mathcal{D}$. We refer this as Encoder $\mathbb{I}$.

Encoder $\mathbb{I}$. Given n, set $m = 2n - \lceil \log n \rceil - 2$.

Input: $\boldsymbol{x} \in \{0,1\}^m$

Output: $\boldsymbol{\sigma} = \textsc{Enc}_{\mathbb{I}}(\boldsymbol{x}) \in \mathcal{C}_a(n)$, where $\mathcal{C}_a(n)$ is defined in (5)

(I) Observe that $m = 2n - \lceil \log 2n \rceil - 1$. Using Encoder $\mathbb{L}$, compute $\boldsymbol{c} \in \mathrm{L}_{-a}(2n)$. In other words, $\mathrm{Syn}(\boldsymbol{c}) = -a \pmod{4n}$.
(II) Compute $\boldsymbol{c}' \triangleq \Phi^{-1}(\boldsymbol{c})$ as defined in (6). Hence, $\mathrm{Rsyn}(0\boldsymbol{c}') = a \pmod{4n}$.
(III) Set $\boldsymbol{\sigma} \triangleq \Psi^{-1}(\boldsymbol{c}')$.

Example 5: Consider $n = 5$, $m = 2n - \lceil \log 2n \rceil - 1 = 5$, $a = 0$. We encode $\boldsymbol{x} = 11000$.

(I) Encode $\boldsymbol{x}$ to a codeword $\boldsymbol{c} \in \mathrm{L}_0(10)$ using Encoder $\mathbb{L}$. Hence, $\boldsymbol{c} = 0110100001$.
(II) Next, we compute $\boldsymbol{c}' = \Phi^{-1}(\boldsymbol{c}) = 0010011111$.
(III) Hence, we obtain $\boldsymbol{\sigma} = \Psi^{-1}(\boldsymbol{c}') = \mathtt{ACTGG}$.

Proposition 2: Encoder $\mathbb{I}$ is correct and has redundancy $\lceil \log n \rceil + 2$ bits. In other words, $\mathrm{ENC}_{\mathbb{I}}(\boldsymbol{x}) \in \mathcal{C}_a(n)$ for all $\boldsymbol{x} \in \{0,1\}^m$.

Proof: Let $\boldsymbol{\sigma} \triangleq \mathrm{ENC}_{\mathbb{I}}(\boldsymbol{x})$. From Remark 1, it suffices to show that $\boldsymbol{c}' \triangleq \Psi(\boldsymbol{\sigma})$ belongs to $\mathrm{L}_a^{\mathrm{burst}}(2n)$, or $\mathrm{Rsyn}(0\boldsymbol{c}') = a \pmod{4n}$. This follows from Lemma 1 and the fact that $\boldsymbol{c} = \Phi(\boldsymbol{c}')$ and $\mathrm{Syn}(\boldsymbol{c}) = -a \pmod{4n}$. ■

Remark 2: Encoder $\mathbb{I}$ runs in linear-time and the redundancy is $\lceil \log n \rceil + 2$ bits. As mentioned earlier, one may use the systematic q-ary single-indel-encoder introduced by Tenengolts [28] for $q = 4$. The redundancy of such encoder is $\log n + c$, where $7 \leqslant c \leqslant 10$ for $n \geqslant 20$. In other words, in general, Encoder $\mathbb{I}$ improves the redundancy by at least four bits.

For completeness, we state the corresponding decoder, Decoder $\mathbb{I}$, for DNA codes that correct a single indel.

Decoder $\mathbb{I}$. For any n, set $m = 2n - \lceil \log n \rceil - 2$.
INPUT: $\boldsymbol{\sigma} \in \mathcal{D}^{n*}$
OUTPUT: $\boldsymbol{x} = \mathrm{DEC}_{\mathbb{I}}(\boldsymbol{\sigma}) \in \{0,1\}^m$

(I) Compute $\widehat{\boldsymbol{c}'} \triangleq \Psi(\boldsymbol{\sigma})$.
(II) Using Theorem 2, compute $\boldsymbol{c}' \triangleq \mathrm{DEC}_a^{\mathrm{burst}}\left(\widehat{\boldsymbol{c}'}\right)$.
(III) Set $\boldsymbol{c} \triangleq \Phi(\boldsymbol{c}')$.
(IV) Set $\boldsymbol{x} \triangleq \boldsymbol{c}|_I$, where I is defined by Encoder $\mathbb{L}$.

Encoder $\mathbb{I}$ can be extended to obtain linear-time q-ary single-indel-correcting encoders with redundancy $(1/2 \log q) \log n + O(1)$. This improves the encoder of Abroshan *et al.* that uses $(\log q + 1) \log n + O(1)$ bits of redundancy [3]. Unfortunately, unlike Tenengolts' encoder [28], this q-ary encoder is not order-optimal.

C. Efficient Encoder for Codes Correcting a Burst of Indels

Recently, Schoeny *et al.* constructed binary codes that corrects a single burst of indels of length b with $\log n + o(\log n)$ bits of redundancy for fixed values of b [25]. Here, we extend our techniques to provide linear-time encoders for the codes of Schoeny *et al.*, and hence obtain order-optimal linear-time burst-indel-correcting encoders for alphabet of size q, $q > 2$. In this article, we focus on the case $q = 4$. The work can be easily extended and generalized to any alphabet size. We first introduce the definition of burst of indels.

Let $\boldsymbol{x} \in \Sigma^n$. We refer to a *b-burst of deletions* when exactly b consecutive deletions have occurred, i.e., from $\boldsymbol{x}$, and we obtain a subsequence $\boldsymbol{x}' = (x_1, x_2, \ldots, x_i, x_{i+b+1}, \ldots, x_n) \in \Sigma^{n-b}$. Similarly, we refer to a *b-burst of insertions* when exactly b consecutive insertions have occurred, i.e., from $\boldsymbol{x}$, and we obtain $\boldsymbol{x}'' = (x_1, x_2, \ldots, x_j, y_1, y_2, \ldots, y_b, x_{j+1}, \ldots, x_n) \in \Sigma^{n+b}$. A *b-burst of indels* refers to either a b-burst of deletions or a b-burst of insertions. We define the *b-burst error ball*:

$$\mathcal{B}_{b-\mathrm{burst}}^{\mathrm{indel}}(\boldsymbol{x}) \triangleq \{\boldsymbol{x}\} \cup \Big\{\boldsymbol{y} : \boldsymbol{y} \text{ is obtained from } \boldsymbol{x} \text{ via a } b\text{-burst of indels}\Big\}.$$

Let $\mathcal{C} \subseteq \Sigma^n$. We say that $\mathcal{C}$ is a *b-burst-indels-correcting code* if $\mathcal{B}_{b-\mathrm{burst}}^{\mathrm{indel}}(\boldsymbol{x}) \cap \mathcal{B}_{b-\mathrm{burst}}^{\mathrm{indel}}(\boldsymbol{y}) = \varnothing$ for all distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$.

In the binary case, Schoeny *et al.* represent the codewords of length n in the b-burst-indels-correcting code as $b \times n/b$ codeword arrays, where b divides n. Thus, for a codeword $\boldsymbol{x}$, the codeword array $A_b(\boldsymbol{x})$ is formed by b rows and n/b columns, and the codeword is transmitted column-by-column. Observe that a b-burst deletes (or inserts) in $\boldsymbol{x}$ exactly one bit from each row of the array $A_b(\boldsymbol{x})$. Here, the ith row of the array is denoted by $A_b(\boldsymbol{x})_i$.

$$A_b(\boldsymbol{x}) = \begin{bmatrix} x_1 & x_{b+1} & \cdots & x_{(j-1)b+1} & \cdots & x_{(n/b-1)b+1} \\ x_2 & x_{b+2} & \cdots & x_{(j-1)b+2} & \cdots & x_{(n/b-1)b+2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_b & x_{2b} & \cdots & x_{jb} & \cdots & x_n \end{bmatrix}.$$

Construction 1 (Schoeny et al. [25]): Given $n > 0, b = o(n)$, Schoeny *et al.* then construct a b-burst-indels-correcting code of length n as follows. For a codeword $\boldsymbol{x}$, the codeword array $A_b(\boldsymbol{x})$ satisfies the following constraints.

- The first row in the array is a VT-code which also restricts the longest run of 0's or 1's to be at most $r = \log 2n + O(1)$. Schoeny *et al.* also provided the *runlength-limited encoder* which uses only one redundancy bit in order to encode binary vectors of maximum run length at most $\lceil \log n \rceil + 3$ (see [25, Appendix B]).
- Each of the remaining $(b-1)$ rows in the array is then encoded using a modified version of the VT-code, which they refer as *shifted VT (SVT) code*. This code corrects a single indel in each row provided the indel position is known to be within P consecutive positions. To obtain the desired redundancy, Schoeny *et al.* set $P = r + 1 = \log 2n + O(1)$.

Formally, the following results were provided by Schoeny *et al.* [25].

Theorem 3 (Schoeny et al. [25]): There exists a pair of linear-time algorithms $\mathrm{ENC}_{RLL} : \{0,1\}^{n-1} \to \{0,1\}^n$ and $\mathrm{DEC}_{RLL} : \{0,1\}^n \to \{0,1\}^{n-1}$ such that the following holds. If $\mathrm{ENC}_{RLL}(\boldsymbol{x}) = \boldsymbol{y}$, then $\boldsymbol{y}$ is a binary vector of maximum run length at most $\lceil \log n \rceil + 3$ and $\mathrm{DEC}_{RLL}(\boldsymbol{y}) = \boldsymbol{x}$.

For $0 \leq c < P$ and $d \in \{0,1\}$, the *shifted VT-code* $\mathrm{SVT}_{c,d,P}(n)$ is defined as

$$\mathrm{SVT}_{c,d,P}(n) \triangleq \Big\{\boldsymbol{x} : \mathrm{Syn}(\boldsymbol{x}) = c \pmod{P} \text{ and } \sum_{i=1}^{n} x_i = d \pmod{2}\Big\}.$$

Theorem 4 (Schoeny et al. [25]): The code $\mathrm{SVT}_{c,d,P}(n)$ can correct a single indel given knowledge of the location of the deleted (or inserted) bit to within P consecutive positions. Furthermore, there exists a linear-time algorithm $\text{DEC}^{SVT}_{c,d,P}$ such that the following holds. If $\boldsymbol{c} \in \text{SVT}_{c,d,P}(n)$, $\boldsymbol{y} \in \mathcal{B}^{\text{indel}}(\boldsymbol{c})$ and the deleted (or inserted) index belongs to $\mathbb{J} \triangleq [j, j+P-1]$ for some $1 \leqslant j \leqslant n-P$, then $\text{DEC}^{SVT}_{c,d,P}(\boldsymbol{y}, \mathbb{J}) = \boldsymbol{c}$.

We now modify the construction of Schoeny *et al.* to obtain a quaternary linear-time b-burst-indels-correcting encoder with redundancy $\log n + o(\log n)$. Observe that if we convert a quaternary sequence of length n into binary sequence of length $2n$, then a b-burst-indels in quaternary sequence results in a $2b$-burst-indels in the corresponding binary sequence. Suppose that we want to encode messages into quaternary code of length $n = bN$.

Construction 2: Let $n = bN, P, r > 0, P \geq r+1$. Given $a \in \mathbb{Z}_N, c \in \mathbb{Z}_P$, and $d \in \mathbb{Z}_2$, let $\mathcal{C}^{\text{indel}}_{b-\text{burst}}(n, P, r; a, c, d)$ be the code contains all codewords $\boldsymbol{\sigma} \in \mathcal{D}^{bN}$ such that when we view $\boldsymbol{x} = \Psi(\boldsymbol{\sigma}) \in \{0,1\}^{2bN}$ as the array $A_{2b}(\boldsymbol{x})$, the following constraints are satisfied.

- The first row $A_{2b}(\boldsymbol{x})_1 \in \text{L}_a(N)$ and the longest run of 0's or 1's is at most r.
- For $2 \leq i \leq 2b$, the ith row $A_{2b}(\boldsymbol{x})_i \in \text{SVT}_{c,d,P}(N)$.

Clearly, $\mathcal{C}^{\text{indel}}_{b-\text{burst}}(n, P, r; a, c, d)$ is a b-burst-indels-correcting code. Before we provide the encoder for $\mathcal{C}^{\text{indel}}_{b-\text{burst}}(n, P, r; a, c, d)$, an encoder for $\text{SVT}_{c,d,P}(n)$ is needed. One can easily modify the encoder for VT-code to obtain an efficient encoder for shifted VT-code with redundancy $\lceil \log P \rceil + 1$. For completeness, we describe the encoder for shifted VT-code $\text{SVT}_{c,d,P}(n)$.

SVT-Encoder. Given n, c, d, P, set $t \triangleq \lceil \log P \rceil$ and $m \triangleq n - t - 1$.

INPUT: $\boldsymbol{x} \in \{0,1\}^m$
OUTPUT: $\boldsymbol{c} \triangleq \text{ENC}_{SVT}(\boldsymbol{x}) \in \text{SVT}_{c,d,P}(n)$

(I) Set $S \triangleq \{2^{j-1} : j \in [t]\} \cup \{P\}$ and $I \triangleq [n] \setminus S$.
(II) Consider $\boldsymbol{c}' \in \{0,1\}^n$, where $\boldsymbol{c}'|_I = \boldsymbol{x}$ and $\boldsymbol{c}'|_S = 0$. Compute the difference $d' \triangleq a - \text{Syn}(\boldsymbol{c}') \pmod{P}$. In the next step, we modify $\boldsymbol{c}'$ to obtain a codeword $\boldsymbol{c}$ with $\text{Syn}(\boldsymbol{c}) = a \pmod{P}$.
(III) Let $y_{t-1} \ldots y_1 y_0$ be the binary representation of d'. In other words, $d' = \sum_{i=0}^{t-1} y_i 2^i$. Then we set $c_{2^{j-1}} = y_{j-1}$ for $j \in [t]$.
(IV) Finally, we set the bit x_P as the parity check bit that satisfies $x_P = d - \sum_{i \in [n] \setminus P} x_i \pmod{2}$.

To conclude this subsection, we provide a linear-time encoder for $\mathcal{C}^{\text{indel}}_{b-\text{burst}}(n; a, c, d)$.

b-Burst-Indels-Encoder. Given $n = bN, r = 2\lceil \log N \rceil + 4, P = r+1, a \in \mathbb{Z}_N, c \in \mathbb{Z}_P, d \in \mathbb{Z}_2$, set $t \triangleq \lceil \log P \rceil$ and $m \triangleq 2bN - \lceil \log N \rceil - (2b-1)(t+1) - 2$.

INPUT: $\boldsymbol{x} \in \{0,1\}^m$
OUTPUT: $\boldsymbol{c} \triangleq \text{ENC}^{\text{indel}}_{b-\text{burst}}(\boldsymbol{x}) \in \mathcal{C}^{\text{indel}}_{b-\text{burst}}(n; a, c, d)$

(I) Set $\boldsymbol{x}_1$ be the first $(N - \lceil \log N \rceil - 2)$ bits in $\boldsymbol{x}$ and for $2 \leq i \leq 2b$, set $\boldsymbol{x}_i$ be the subsequence of length $(N - t - 1)$ of $\boldsymbol{x}$ such that $\boldsymbol{x} = \boldsymbol{x}_1 \boldsymbol{x}_2 \ldots \boldsymbol{x}_{2b}$.
(II) Let $\boldsymbol{y}'_1 = \text{ENC}_{RLL}(\boldsymbol{x}_1)$ and use Encoder $\mathbb{L}$ as described in Subsection II-C to encode $\boldsymbol{y}_1 = \text{ENC}_{\mathbb{L}}(\boldsymbol{y}'_1)$.
(III) For $2 \leq i \leq 2b$, use SVT Encoder ENC_{SVT} to encode $\boldsymbol{y}_i = \text{ENC}_{SVT}(\boldsymbol{x}_i) \in \text{SVT}_{c,d,P}(n)$.
(IV) Finally, set $\boldsymbol{y} = \boldsymbol{y}_1 || \boldsymbol{y}_2 || \ldots || \boldsymbol{y}_{2b}$ and output $\boldsymbol{\sigma} \triangleq \Psi^{-1}(\boldsymbol{y})$.

For given $n = bN$, the b-Burst-Indels-Encoder costs $\lceil \log N \rceil + O(\log \log N) = \log n + O(\log \log n)$ bits of redundancy.

Proposition 3: Let $b = \Theta(1)$. The b-Burst-Indels-Encoder is correct. In other words, $\text{ENC}^{\text{indel}}_{b-\text{burst}}(\boldsymbol{x}) \in \mathcal{C}^{\text{indel}}_{b-\text{burst}}(n, P, r; a, c, d)$ for all $\boldsymbol{x} \in \{0,1\}^m$.

Proof: Let $\boldsymbol{\sigma} \triangleq \text{ENC}^{\text{indel}}_{b-\text{burst}}(\boldsymbol{x})$ and $\boldsymbol{y} = \Psi(\boldsymbol{\sigma})$. It is sufficient to show that when we view $\boldsymbol{y}$ as the array $A_{2b}(\boldsymbol{y})$ the constraints in Construction 2 are satisfied. Based on our encoder, $\boldsymbol{y} = \boldsymbol{y}_1 || \boldsymbol{y}_2 || \ldots || \boldsymbol{y}_{2b}$ and $\boldsymbol{y}_i = \text{ENC}_{SVT}(\boldsymbol{x}_i) \in \text{SVT}_{c,d,P}(n)$ for $2 \leq i \leq 2b$. It remains to show that the first row $\boldsymbol{y}_1 \in \text{L}_a(N)$ and the longest run of 0's or 1's is at most r. Observe that $\boldsymbol{y}'_1 = \text{ENC}_{RLL}(\boldsymbol{x}_1)$, which implies the longest run of 0's or 1's in $\boldsymbol{y}'_1$ is at most $(\lceil \log N \rceil + 3)$ according to Theorem 3. Therefore, the maximum run in $\boldsymbol{y}_1$ after $\text{ENC}_{\mathbb{L}}$ is at most $(\lceil \log N \rceil + 3) + (\lceil \log N \rceil + 1) = 2\lceil \log n \rceil + 4 = r$ (refer to Subsection III-C, Encoder $\mathbb{L}$). ■

IV. ENCODERS CORRECTING A SINGLE EDIT

In this section, we present efficient encoders for codes over $\mathcal{D}^n$ that can correct a single edit. Unlike indel error, the main challenge to design codes correcting edit is that when substitution error occurs in a DNA strand $\boldsymbol{\sigma}$, it might not affect $\boldsymbol{U_\sigma}$ or $\boldsymbol{L_\sigma}$. Therefore, putting a VT constraint in either one of these sequences might not tell any information about the loss in the other sequence. We illustrate this scenario through the example below.

Example 6: Suppose that $\boldsymbol{\sigma} = $ ACAGTG. Suppose a substitution error occurs at the third symbol, changing A to T, and we received $\boldsymbol{\sigma}_1 = $ ACTGTG. On the other hand, suppose a substitution error also occurs at the third symbol, changing A to C, and we received $\boldsymbol{\sigma}_2 = $ ACCGTG. We then see the change in $\boldsymbol{U_\sigma}$ and $\boldsymbol{L_\sigma}$.

$\boldsymbol{\sigma}$	A	C	A	G	T	G
$\boldsymbol{U_\sigma}$	0	1	0	1	0	1
$\boldsymbol{L_\sigma}$	0	0	0	1	1	1

$\boldsymbol{\sigma}_1$	A	C	T	G	T	G
$\boldsymbol{U_\sigma}$	0	1	0	1	0	1
$\boldsymbol{L_\sigma}$	0	0	1	1	1	1

$\boldsymbol{\sigma}_2$	A	C	C	G	T	G
$\boldsymbol{U_\sigma}$	0	1	1	1	0	1
$\boldsymbol{L_\sigma}$	0	0	0	1	1	1

According to Proposition 1, if $\boldsymbol{\sigma}' \in \mathcal{B}^{\text{edit}}(\boldsymbol{\sigma})$ implies that $\boldsymbol{U_{\sigma'}} \in \mathcal{B}^{\text{edit}}(\boldsymbol{U_\sigma})$ and $\boldsymbol{L_{\sigma'}} \in \mathcal{B}^{\text{edit}}(\boldsymbol{L_\sigma})$. Therefore, a simple solution is to encode both of the sequences $\boldsymbol{U_\sigma}$ and $\boldsymbol{L_\sigma}$ into $\text{L}_a(n)$ using Encoder $\mathbb{L}$. Recall that $\text{L}_a(n)$ can detect and correct a single edit. Hence, $\boldsymbol{\sigma} = \Psi^{-1}(\boldsymbol{U_\sigma} || \boldsymbol{L_\sigma})$ can correct a single edit. This simple encoder costs $2\lceil \log n \rceil + 2$ bits of redundancy, which is at most twice the optimal. For completeness, we present the encoder below and refer this as the *Encoder A*. We remark that Encoder A is crucial in construction of GC-balanced codebooks in Section V.

A. First Class of Single-Edit-Correcting Codes

Proposition 4: Set $m = 2(n - \lceil \log n \rceil - 1)$. There exists a linear-time single-edit-correcting encoder $\textsc{Enc}_{\mathbb{E}}^{A} : \{0,1\}^m \rightarrow \mathcal{D}^n$ with redundancy $2\lceil \log n \rceil + 2$.

Proof: We describe the single-edit-correcting encoder. Consider the message $\boldsymbol{x}_1\boldsymbol{x}_2 \in \{0,1\}^m$ with $|\boldsymbol{x}_1| = |\boldsymbol{x}_2| = n - \lceil \log n \rceil - 1$. For $i \in [2]$, set $\boldsymbol{c}_i = \textsc{Enc}_{\mathbb{L}}(\boldsymbol{x}_i) \in \{0,1\}^n$. Then set $\textsc{Enc}_{\mathbb{E}}^{A}(\boldsymbol{x}_1\boldsymbol{x}_2) = \Psi^{-1}(\boldsymbol{c}_1 || \boldsymbol{c}_2)$. ■

An alternative approach to correct a single edit is to consider the quaternary Hamming Code $\mathcal{C}_H$ with $\log(3n+1) \approx \log n + 1.58$ bits of redundancy. If we partition the codewords in $\mathcal{C}_H$ into $4n$ equivalence classes according to their VT parameters, we are then guaranteed a code that corrects a single edit with redundancy at most $2\log n + 3.58$. In contrast, the linear-time encoder in Proposition 4 has redundancy $2\lceil \log n \rceil + 2$.

Furthermore, the best known linear-time encoder that maps messages into one such class is the one by Abroshan *et al.* and the encoder introduces additional $3\lceil \log n \rceil + 2$ redundant bits [3]. Thus, an efficient single-edit-correcting encoder obtained via the construction has redundancy approximately $4\log n + 3.58$.

B. Order-Optimal Quaternary Codes Correcting a Single Edit

In this subsection, we consider the quaternary alphabet $\Sigma_4 = \{0,1,2,3\}$ as a subset of integers. Our main result in this subsection is the construction of a quaternary single-edit-correcting code that has redundancy $\log n + O(\log\log n)$, where n is the length of the codeword.

As noted in Section II, the $\log n$-bit VT-syndrome computed on the signature of a word is unable to uniquely identify single substitution errors. Hence, we introduce another $\log n$-bit VT-syndrome computed on the *non-binary word* itself to correct single substitution errors. However, this incurs roughly $2\log n$ bits of redundancy. To achieve order-optimality, *á la* Schoeny *et al.* [25], we instead use the *shorter* shifted VT syndrome for the signature. To correct single indels with the shifted VT codes, we then impose a certain *constraint* on all codewords. With the appropriate choice of parameters, we demonstrate that the resulting redundancy is as desired. We outline our subsection as follows.

(I) We first define the sum-balanced constraint in Definition 3 and show that most quaternary words obey this constraint in Lemma 2.

(II) In Construction 3, we defined our single-edit-correcting codes $\mathcal{C}^B$. Lemmas 4 and 6 then provide the decoding algorithms for single substitutions and deletions, respectively.

(III) In Theorem 6, we provide a choice of parameters that achieve order-optimality. The remaining of the subsection describes an efficient encoder for $\mathcal{C}^B$.

First, we define the *sum-balanced constraint.*

Definition 3: Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \Sigma_4^n$. A window $\boldsymbol{W}$ of length k of $\boldsymbol{x}$, i.e. $\boldsymbol{W} = (x_{i+1}, \ldots, x_{i+k})$ is called *sum-balanced* if $k < \sum_{x_j \in \mathbf{W}} x_j < 2k$. A word $\boldsymbol{x}$ is *k-sum-balanced* if every window $\boldsymbol{W}$ of the word $\boldsymbol{x}$ is sum-balanced whenever the window length is at least k.

A key ingredient of our code construction is the set of all k-sum-balanced words.

$$\mathrm{Bal}_k(n) \triangleq \{\boldsymbol{x} \in \Sigma_4^n : \boldsymbol{x} \text{ is } k\text{-sum-balanced}\}.$$

We have the following properties of $\mathrm{Bal}_k(n)$. The first lemma states that whenever $k = \Omega(\log n)$, the set $\mathrm{Bal}_k(n)$ incurs at most one symbol of redundancy.

Lemma 2: Given $n \geqslant 4$, if $k = 36\log n$, then $|\mathrm{Bal}_k(n)| \geq 4^{n-1}$.

To prove this lemma, we require *Hoeffding's inequality* [13].

Theorem 5 (Hoeffding's Inequality): Let $Z_1, Z_2, \ldots, Z_n$ be independent bounded random variables such that $a_i \leqslant Z_i \leqslant b_i$ for all i. Let $S_n = \sum_{i=1}^n Z_i$. For any $t > 0$, we have

$$P(S_n - E[S_n] \geq t) \leq e^{-2t^2/\sum_{i=1}^n (b_i - a_i)^2}.$$

Proof of Lemma 2: Let $\boldsymbol{x}$ be a uniformly at random selected element from Σ_4^n. We evaluate the probability that the first k-length window $\boldsymbol{W}$ of $\boldsymbol{x}$ does not satisfy the sum-balanced constraint. Applying Hoeffding's inequality we obtain:

$$P\left(\left|\sum_{x_i \in \mathbf{W}} x_i - \frac{3k}{2}\right| \geq \frac{k}{2}\right) = 2\,P\left(\left(\sum_{x_i \in \mathbf{W}} x_i - \frac{3k}{2}\right) \geq \frac{k}{2}\right)$$
$$\leq 2\,e^{-\frac{2k^2/4}{9k}} = 2e^{-k/18}.$$

Since $f(k) = e^{-k/18}$ is decreasing in k, the probability that a k'-length window violates the sum-balanced constraint is at most $f(k)$ for $k' \geqslant k$. Also, since there are at most n^2 windows, applying the union bound and setting $k = 36\log n$ yields

$$P(\boldsymbol{x} \notin \mathrm{Bal}_k(n)) \leq n^2 2e^{-\frac{k}{18}} = 2n^2\,e^{-2\log n} = 2\,n^{2-2\log e}.$$

Therefore, the size of $\mathrm{Bal}_k(n)$ is at least

$$|\mathrm{Bal}_k(n)| \geq 4^n(1 - 2\,n^{2-2\log e}).$$

Since $n \geq 4$, we have that $1 - 2\,n^{2-2\log e} \geq 1/4$. Therefore, $|\mathrm{Bal}_k(n)| \geq 4^{n-1}$. ■

Next, we recall that the signature of $\boldsymbol{x}$, denoted by $\alpha(\boldsymbol{x})$, is a binary vector of length $n-1$, where $\alpha(\boldsymbol{x})_i = 1$ if $x_{i+1} \geq x_i$, and 0 otherwise, for $i \in [n-1]$. It is well-known that if a single deletion occurs in $\boldsymbol{x}$, resulting in $\boldsymbol{y}$, then $\alpha(\boldsymbol{y})$ can be obtained by deleting a single symbol from $\alpha(\boldsymbol{x})$. The following lemma provides an upper bound on the length of a run of a k-balanced word and its signature.

Lemma 3: Let $\boldsymbol{x} \in \mathrm{Bal}_k(n)$. Then the length of a run in $\boldsymbol{x}$ is at most $(k-1)$ while the length of a run in $\alpha(\boldsymbol{x})$ is strictly less than $2(k-1)$.

Proof: Let $\boldsymbol{x} \in \mathrm{Bal}_k(n)$. We first show that the length of a run in $\boldsymbol{x}$ is at most $(k-1)$. Suppose otherwise that the run in $\boldsymbol{x}$ is $(x_{i+1}, \ldots, x_{i+t})$ where $t \geq k$. Since $\boldsymbol{x} \in \mathrm{Bal}_k(n)$, we have $x_{i+1} + x_{i+2} + \cdots + x_{i+t} = t x_{i+1} \in (t, 2t)$. We have a contradiction since $x_{i+1} \in \{0,1,2,3\}$.

Let $\boldsymbol{W}_{(0,1)}$ be a window in $\boldsymbol{x}$ that contains only 0 and 1 symbols. We claim that the length of $\boldsymbol{W}_{(0,1)}$ is at most $(k-1)$. Otherwise, assume that the size is s where $s \geq k$. Then the sum of symbols in this window is at most s. We then get a contradiction since $\boldsymbol{x} \in \mathrm{Bal}_k(n)$ and the sum of such symbols is strictly more than s. Similarly, let $\boldsymbol{W}_{(2,3)}$ be a window in $\boldsymbol{x}$ that contains only 2 and 3 symbols. Then the length of $\boldsymbol{W}_{(2,3)}$ is at most $(k-1)$.

We look at runs in the signature $\alpha(\boldsymbol{x})$. First, the run of zeroes in $\alpha(\boldsymbol{x})$ is at most three and this happens when the corresponding window is $(3,2,1,0)$. Next, we look at a run of ones in $\alpha(\boldsymbol{x})$. Now, such a run is obtained when there is a subsequence $\boldsymbol{y}$ in $\boldsymbol{x}$ of the form $\boldsymbol{y} = 0^{t_1}1^{t_2}2^{t_3}3^{t_4}$. As shown earlier, the window $0^{t_1}1^{t_2}$ has length at most $(k-1)$ and the window $2^{t_3}3^{t_4}$ has length at most $(k-1)$. Hence, the length of $\boldsymbol{y}$ is at most $(2k-2)$, which the corresponding run of ones in $\alpha(\boldsymbol{x})$ is at most $2(k-1)$. ■

We are now ready to present our main code construction for this subsection.

Construction 3: Given $k < n$, set $P = 5k$. For $a \in \mathbb{Z}_{4n+1}$, $b \in \mathbb{Z}_P$, $c \in \mathbb{Z}_2$ and $d \in \mathbb{Z}_7$, set $\mathcal{C}^B(n; a, b, c, d)$ as follows.

$$\begin{aligned}&\mathcal{C}^B(n; k, a, b, c, d)\\ &\quad = \Big\{\boldsymbol{x} \in \mathrm{Bal}_k(n) : \mathrm{Syn}(\boldsymbol{x}) = a \pmod{4n+1},\\ &\qquad \alpha(\boldsymbol{x}) \in \mathrm{SVT}_{b,c,P}(n-1), \text{ and } \sum_{i=1}^{n} x_i = d \pmod 7\Big\}.\end{aligned}$$

In what follows, we first prove the correctness of Construction 3 by providing an efficient decoder that can correct a single edit in linear time. Subsequently, in Theorem 6, we show that $\mathcal{C}$ is order-optimal with a suitable choice of k.

The decoding operates as follows.

- First, the decoder decides whether a deletion, insertion or substitution has occurred. Note that this information can be recovered by simply observing the length of the received word.
- If the length of the output word is equal to n, then we conclude that at most a single substitution error has occurred and Lemma 4 provides the procedure to correct the substitution error (if it exists).
- Otherwise, the output vector has length $n-1$ (or $n+1$). We then conclude that a single deletion (or insertion) has occurred and we proceed according to Lemma 6.

Lemma 4: The code $\mathcal{C}^B(n; a, b, c, d)$ corrects a single substitution in linear time.

Proof: Suppose that the original codeword is $\boldsymbol{x}$ and we receive a vector $\boldsymbol{y}$ of length n. The decoder proceeds as follows.

- **Step 1. Error detection**. Compute $d' \triangleq \sum_{i=1}^{n} y_i \pmod 7$. If $d' = d$, then we conclude that there is no error and $\boldsymbol{y}$ is the original codeword $\boldsymbol{x}$. Otherwise, a substitution error occurs. Henceforth, we assume that it occurs in position j.
- **Step 2.** Next, we compute $y_j - x_j = d' - d \pmod 7$. Since, $x_j, y_j \in \{0,1,2,3\}$, we can determine the value of $y_j - x_j$ as integers.
- **Step 3. Error location**. Compute $a' = \mathrm{Syn}(\boldsymbol{y}) \pmod{4n+1}$ and clearly, we have $a' - a = j(y_j - x_j) \pmod{4n+1}$. Now, since $1 \leqslant j \leqslant n$, we uniquely determine the index j with the value of $y_j - x_j$ from Step 2.
- **Step 4. Recovering the symbol**. Finally, given the error location j, we recover x_j using y_j and $y_j - x_j$.

It is easy to see that all the decoding steps run in $O(n)$ time. Hence, $\mathcal{C}^B(n; a, b, c, d)$ corrects a single substitution in linear time. ■

It remains to describe the decoding procedure for correcting a single deletion. To this end, we have the following technical lemma that that characterizes words whose deletion balls intersect and whose syndromes are the same.

Lemma 5: Let $\boldsymbol{x}$ and $\boldsymbol{z}$ be two words such that the following hold.

(B1) $\boldsymbol{x}$ and $\boldsymbol{z}$ belongs to $\mathrm{Bal}_k(n)$.
(B2) $\mathrm{Syn}(\boldsymbol{x}) = \mathrm{Syn}(\boldsymbol{z}) = a \pmod{4n+1}$.
(B3) $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} z_i = d \pmod 7$.
(B4) $\mathcal{B}^{\mathrm{indel}}(\boldsymbol{x}) \cap \mathcal{B}^{\mathrm{indel}}(\boldsymbol{z})$ is non-empty.

Suppose that $\boldsymbol{y} \in \mathcal{B}^{\mathrm{indel}}(\boldsymbol{x}) \cap \mathcal{B}^{\mathrm{indel}}(\boldsymbol{z})$. If $\boldsymbol{y}$ is obtained from $\boldsymbol{x}$ by deleting x_i and obtained from $\boldsymbol{z}$ by deleting z_j, then we have that $|i-j| < k$.

Proof: From (B3) and the fact that x_i and z_j belongs to Σ_4, we have that the deleted symbols are the same. In other words, $x_i = z_j$ and we set this value to be m.

Without loss of generality, assume that $i > j$. We compute the syndrome of $\boldsymbol{y}$

$$\mathrm{Syn}(\boldsymbol{y}) = \sum_{t=1}^{n-1} t y_t = a' \pmod{4n+1},$$

and consider the quantity

$$(a-a') \pmod{4n+1} = \sum_{t=1}^{n} t x_t - \sum_{t=1}^{n-1} t y_t = im + \sum_{t=i}^{n-1} y_t. \quad (8)$$

On the other hand, if we compute the same quantity using $\boldsymbol{z}$, we have that

$$(a-a') \pmod{4n+1} = jm + \sum_{t=j}^{n-1} y_t. \quad (9)$$

Now, we know that $y_t = z_{t+1}$ for $t \geqslant j$. Subtracting (9) from (8) and setting $i - j = b$, we have that

$$bm = \sum_{t=j+1}^{i} z_t \pmod{4n+1}.$$

Since $bm < 4n+1$ and $0 \le z_t \le 3$ for all t, we have that $bm = \sum_{t=j+1}^{i} z_t$. Suppose to the contrary that $i - j = b \ge k$. Then since $\boldsymbol{z} \in \mathrm{Bal}_k(n)$, we have that $b < \sum_{t=j+1}^{i} z_t < 2b$. Hence, we have that $b < bm < 2b$, contradicting the fact that m is an integer. Therefore, $i - j < k$. ■

Finally, we present the deletion-correcting procedure.

Lemma 6: The code $\mathcal{C}^B(n; a, b, c, d)$ corrects a single deletion or single insertion in linear time.

Proof: Since the decoding process for correcting an insertion is similar to correcting a deletion, we only present the case of a deletion. Now, let $\boldsymbol{y}$ be the result of a deletion occurring to $\boldsymbol{x} \in \mathcal{C}^B(n; a, b, c, d)$ at position i. To recover $\boldsymbol{x}$, the decoder proceeds as follows.

- **Step 1. Identifying the deleted symbol**. From the constraint $\sum_{t=1}^{n} x_t \equiv d \pmod 7$, we compute the deleted symbol $m \triangleq d - \sum_{t=1}^{n-1} y_t \pmod 7$.

- **Step 2. Localizing the deletion**.
  - **Localizing the deletion in $\boldsymbol{x}$**. Using (8) or (9), we compute the possible deleted positions. Specifically, set $a' = \text{Syn}(\boldsymbol{y}) \pmod{4n+1}$ and we compute $\mathbb{J} = \{1 \le j \le n : a' + jm + \sum_{t=j}^{n-1} y_t = a \pmod{4n+1}\}$. According to Lemma 5, we have $|i-j| < k$ for all $i, j \in \mathbb{J}$.
  - **Localizing the deletion in $\alpha(\boldsymbol{x})$**. For $j \in \mathbb{J}$, set $\boldsymbol{y}_j^+$ to be word obtained by inserting m at index j. Suppose that $\alpha(\boldsymbol{y})$ can be obtained by deleting a single symbol from $\alpha(\boldsymbol{y}_j^+)$ at position j'. Then we add j' to the set $\mathbb{J}'_j$. We set $\mathbb{J}' \triangleq \bigcup_{j\in\mathbb{J}} \mathbb{J}'_j$. We claim that $\mathbb{J}' \subseteq [j'_{min}, j'_{min}+P-1]$ where j'_{min} is the smallest index. Indeed, Lemma 3 states that the longest run in $\boldsymbol{x}$ is at most $(k-1)$ and the longest run in $\alpha(\boldsymbol{x})$ is less than $2(k-1)$. Therefore, the interval containing $\mathbb{J}'$ is of length at most $2(k-1)+k+2(k-1) < 5k = P$.
- **Step 3. Recovering** $\alpha(\boldsymbol{x})$. Since $\alpha(\boldsymbol{x}) \in SVT_{b,c,P}(n-1)$, we apply $\text{Dec}_{b,c,P}^{SVT}$ (from Theorem 4) to $\alpha(\boldsymbol{y})$ and $\mathbb{J}'$ to recover $\alpha(\boldsymbol{x})$.
- **Step 4. Recovering $\boldsymbol{x}$**. From recovered $\alpha(\boldsymbol{x})$ and symbol m, we can determine $\boldsymbol{x}$ uniquely.

It is easy to see that all the decoding steps run in $O(n)$. Hence, $\mathcal{C}(n; a, b, c, d)$ can correct a single deletion or single insertion in linear time. ■

Combining the results of Lemma 4 and Lemma 6 we have the following theorem.

Theorem 6: The code $\mathcal{C}^B(n; a, b, c, d)$ corrects a single edit in linear-time. There exist a, b, c, d such that the size of $\mathcal{C}^B(n; a, b, c, d)$ is at least

$$|\mathcal{C}(n; a, b, c, d)| \ge \frac{|\text{Bal}_k(n)|}{35(4n+1)k}.$$

When $k = 36 \log_4 n$, we have that the redundancy is at most $\log_4 n + O(\log\log n)$ bits.

Proof: It remains to demonstrate the property of the code size. The lower bound is obtained using pigeonhole principle. Setting $k = 36\log_4 n$, we have that $|\text{Bal}_k(n)| \geqslant 4^{n-1}$ and hence the redundancy is $\log_4 n + O(\log\log n)$. ■

Remark 3: Observe that $\mathcal{C}^B(n; a, b, c, d)$ requires at least $1 + \log(35(4n+1)k) \geqslant 13 + \log n$ bits of redundancy. In contrast, the encoder $\text{Enc}_{\mathbb{E}}^A$ from Proposition 4 requires at most $4 + 2\log n$ bits of redundancy. Therefore, even though $\mathcal{C}^B(n; a, b, c, d)$ is order-optimal, the encoder $\text{Enc}_{\mathbb{E}}^A$ incurs less redundant bits whenever $n \leqslant 2^9 = 512$.

Finally, we provide an efficient encoder that encodes binary messages into a quaternary codebook that corrects a single edit. While the codebook in general is not the same as $\mathcal{C}^B(n; a, b, c, d)$, the decoding procedure is similar and the number of redundant bits is similar to that of $\mathcal{C}^B(n; a, b, c, d)$.

A high-level description of the encoding procedure is as follows.

(I) **Enforcing the k-sum-balanced constraint**. Given an arbitrary message $\boldsymbol{x}$ over Σ_4 of length $m-1$, we encode $\boldsymbol{x}$ to a word $\boldsymbol{z} \in \text{Bal}_k(m)$. However, it is not straightforward to efficiently code for this constraint. Hence, we instead impose a tighter constraint on the sum, but we impose the constraint only on windows of length exactly k. We provide the formal definition of *restricted-sum-balanced* in Definition 4.

(II) **Appending the syndromes**. Using $\boldsymbol{z}$, we then compute its VT syndrome $a \triangleq \text{Syn}(\boldsymbol{z}) \pmod{4n+1}$, SVT syndrome $b \triangleq \text{Syn}(\alpha(\boldsymbol{z})) \pmod{P}$ and $c \triangleq \sum_{i=1}^n \alpha(\boldsymbol{z})_i \pmod 2$, and the check $d \triangleq \sum_{i=1}^n z_i \pmod 7$. Finally, we append the quaternary representations of a, b, c and d to the end of $\boldsymbol{z}$ with a *marker* between $\boldsymbol{z}$ and these syndromes. It turns out we can modify the procedures given in Lemma 4 and Lemma 6 and correct a single edit in linear time. We do so in Theorem 7.

Enforcing the k-sum-balanced constraint. As mentioned earlier, instead of encoding into k-sum-balanced words, we require the words to have the following property.

Definition 4: Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \Sigma_4^n$. A window $\boldsymbol{W}$ of length k of $\boldsymbol{x}$, i.e. $\boldsymbol{W} = (x_{i+1}, \ldots, x_{i+k})$ is called *restricted-sum-balanced* if $5k/4 < \sum_{x_j\in\boldsymbol{W}} x_j < 7k/4$. A word $\boldsymbol{x}$ is *k-restricted-sum-balanced* if every window $\boldsymbol{W}$ of length exactly k of the word $\boldsymbol{x}$ is restricted-sum-balanced.

The following lemma states that a k-restricted-sum-balanced is also $(4k)$-sum-balanced.

Lemma 7: Let $\text{Bal}_k^*(n) = \{\boldsymbol{x} \in \Sigma_4^n : \boldsymbol{x}$ is k-restricted-sum-balanced$\}$. We have that $\text{Bal}_k^*(n) \subseteq \text{Bal}_{4k}(n)$.

Proof: Let $\boldsymbol{x} \in \text{Bal}_k^*(n)$ and $\boldsymbol{W}$ be a window of $\boldsymbol{x}$ whose length is at least $4k$.

Suppose that the length of $\boldsymbol{W}$ is $Mk+N$ for some $M \ge 4$ and $0 \le N < k$. Hence, we write $\boldsymbol{W} \triangleq \boldsymbol{z}_1\boldsymbol{z}_2\ldots\boldsymbol{z}_M\boldsymbol{z}_{M+1}$ where each $\boldsymbol{z}_i$ is a window of length k for $1 \le i \le M$ and the window $\boldsymbol{z}_{M+1}$ is of length N.

Since $\boldsymbol{x} \in \text{Bal}_k^*(n)$, we then have $\boldsymbol{z}_i$ is k-restricted-sum-balanced for $1 \le i \le M$. Therefore, $5k/4 < \sum_{x_j\in\boldsymbol{z}_i} x_j < 7k/4$ for $1 \le i \le M$. Hence, for $M \ge 4$ and $0 \le N < k$, we have

$$\begin{aligned}\sum_{x_j\in\boldsymbol{W}} x_j &> \sum_{i=1}^{M}\left(\sum_{x_t\in\boldsymbol{z}_i} x_t\right)\\ &> 5Mk/4 = Mk + Mk/4\\ &\ge Mk + k > Mk + N, \text{ and}\\ \sum_{x_j\in\boldsymbol{W}} x_j &< \sum_{i=1}^{M}\left(\sum_{x_t\in\boldsymbol{z}_i} x_t\right) + 3N\\ &< 7Mk/4 + 3N = 2Mk + 2N + (N - Mk/4)\\ &< 2Mk + 2N.\end{aligned}$$

This shows that $\boldsymbol{W}$ is sum-balanced. Hence, $\boldsymbol{x} \in \text{Bal}_{4k}(n)$. ■

Now, we may modify the sequence replacement techniques [9], [30] to encode for the restricted-sum-balanced constraint.

Proposition 5: Suppose that $k \geqslant 72\log n$. Then there is a pair of efficient algorithms $\text{Enc}_{k-\text{rsb}} : \Sigma_4^{n-1} \to \Sigma_4^n$ and $\text{Dec}_{k-\text{rsb}} : \text{Im}(\text{Enc}_{k-\text{rsb}}) \to \Sigma_4^{n-1}$ such that $\text{Enc}_{k-\text{rsb}}(\boldsymbol{x}) \in \text{Bal}_k^*(n)$ and $\text{Dec}_{k-\text{rsb}} \circ \text{Enc}_{k-\text{rsb}}(\boldsymbol{x}) = \boldsymbol{x}$ for all $\boldsymbol{x} \in \Sigma_4^{n-1}$.

Appending the syndromes. Set $k = 72\log n$ in Proposition 5. Suppose that the message $\boldsymbol{x} \in \Sigma_4^{n-1}$ is encoded to $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in \mathrm{Bal}_k^*(n)$, or, $\boldsymbol{y} \triangleq \textsc{Enc}_{k-\mathrm{rsb}}(\boldsymbol{x})$.

Let $k' = 4k$ and $P = 5k'$. The encoder computes the following four sequences over Σ_4.

- Set the *VT syndrome* $a \triangleq \mathrm{Syn}(\boldsymbol{y}) \pmod{4n+1}$. Let $\boldsymbol{R}_1$ be the quaternary representations of a of length $\lceil \log_4(4n+1)\rceil$.
- Set the *SVT syndrome* $b \triangleq \mathrm{Syn}(\alpha(\boldsymbol{y})) \pmod{P}$. Let $\boldsymbol{R}_2$ be the quaternary representations of b of length $\lceil \log_4 \ P\rceil$.
- Set the *parity check* of $\alpha(\boldsymbol{y})$, i.e. $c \triangleq \sum_{i=1}^n \alpha(\boldsymbol{y})_i \pmod 2$. Let $\boldsymbol{R}_3 \equiv c$.
- Set the *check* $d \triangleq \sum_{i=1}^n y_i \pmod 7$. Let $\boldsymbol{R}_4$ be the quaternary representations of d of length 2.

Let r be the smallest symbol in $\Sigma_4\backslash\{y_n\}$ and let the marker $\boldsymbol{M} = (r, r)$. We append $\boldsymbol{M}, \boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3, \boldsymbol{R}_4$ to $\boldsymbol{y}$ and output the codeword $\boldsymbol{yMR}_1\boldsymbol{R}_2\boldsymbol{R}_3\boldsymbol{R}_4$ of length $N = (n+\lceil\log_4(4n+1)\rceil + \lceil\log_4 \ P\rceil + 5)$.

Finally, we summarize our encoding procedure and demonstrate its correctness.

Theorem 7: Given n, a, b, c, d, set $k = 72\log n$, $k' = 4k$, $P = 5k'$ and $N = (n + \lceil\log_4(4n+1)\rceil + \lceil\log_4 \ P\rceil + 5)$. We define $\textsc{Enc}_{\mathbb{E}}^B : \Sigma_4^{n-1} \to \Sigma_4^N$ as follows. Set $\boldsymbol{y} = \textsc{Enc}_{k-\mathrm{rsb}}(\boldsymbol{x})$ as in Theorem 5 and let $\boldsymbol{M}, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_4$ be as defined above. If we set $\textsc{Enc}_{\mathbb{E}}^B(\boldsymbol{x}) \triangleq \boldsymbol{yMR}_1\boldsymbol{R}_2\boldsymbol{R}_3\boldsymbol{R}_4$, then the code defined by $\textsc{Enc}_{\mathbb{E}}^B$ corrects a single edit in linear time. Therefore, the redundancy of $\textsc{Enc}_{\mathbb{E}}^B$ is $\log n + O(\log\log n)$ bits.

Proof: It remains to provide the corresponding decoder and show that it corrects a single edit in linear time. Suppose that we receive $\boldsymbol{z}'$. The idea is to recover $\boldsymbol{y}$ as the first n symbols in $\boldsymbol{z}'$ and then use the decoder in Proposition 5 to recover the information sequence $\boldsymbol{x}$. First, the decoder decides whether a deletion, insertion or substitution has occurred. Note that this information can be recovered by simply observing the length of the received word. The decoding operates as follows.

(i) **If the length of $\boldsymbol{z}'$ is** N, we conclude that at most a single substitution error has occurred. Let $\boldsymbol{z}' = (z_1', \ldots, z_N')$. The decoder sets $\boldsymbol{y}'$ as the first n symbols of $\boldsymbol{z}'$, the marker $\boldsymbol{M} = (z_{n+1}', z_{n+2}')$, $\boldsymbol{R}_1'$ as the next $\lceil\log_4(4n+1)\rceil$ symbols, $\boldsymbol{R}_2'$ as the next $\lceil\log_4 \ P\rceil$ symbols, $\boldsymbol{R}_3'$ as the next symbol and the last two symbols as $\boldsymbol{R}_4'$. The decoder proceeds as follows.

- **Checking the marker**. If z_{n+1}', z_{n+2}' are not identical, then the substitution occurred here. The decoder concludes that $\boldsymbol{y} \equiv \boldsymbol{y}'$. On the other hand, if z_{n+1}', z_{n+2}' are identical, the decoder concludes that there is no error in the marker. The decoder computes $a' \triangleq \mathrm{Syn}(\boldsymbol{y}') \pmod{4n+1}$, $b' \triangleq \mathrm{Syn}(\alpha(\boldsymbol{y}')) \pmod{P}$, $c' \triangleq \sum_{i=1}^n \alpha(\boldsymbol{y}')_i \pmod 2$ and $d' \triangleq \sum_{i=1}^n y_i' \pmod 7$, and proceeds to the next step.
- **Comparing the check**. If $\boldsymbol{R}_4'$ corresponds the quaternary representation of d', the decoder concludes that $\boldsymbol{y} \equiv \boldsymbol{y}'$. Otherwise, it proceeds to the next step.
- **Comparing the VT syndrome**. If $\boldsymbol{R}_1'$ corresponds to the quaternary representations of a', the decoder concludes that $\boldsymbol{y} \equiv \boldsymbol{y}'$. Otherwise, there must be a substitution in the $\boldsymbol{y}'$. Hence, there is no error in $\boldsymbol{R}_1', \boldsymbol{R}_2', \boldsymbol{R}_3'$, and $\boldsymbol{R}_4'$. The decoder follows the steps in Lemma 4 to recover $\boldsymbol{y}$ from $\boldsymbol{y}'$.

(ii) **If the length of $\boldsymbol{z}'$ is** $(N-1)$, we conclude that a single deletion has occurred. Suppose $\boldsymbol{z}' = (z_1', \ldots, z_{N-1}')$. The decoder proceeds as follows.

- **Localizing the deletion.** If z_n' and z_{n+1}' are different, the decoder concludes that there is no deletion in $\boldsymbol{y}$ and sets $\boldsymbol{y}$ as the first n symbols of $\boldsymbol{z}'$. Otherwise, there is a deletion in the first n symbols. Hence, there is no error in $\boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3$, and $\boldsymbol{R}_4$.
- **Recovering $\boldsymbol{y}$.** The decoder sets $\boldsymbol{y}'$ as the first $(n-1)$ symbols in $\boldsymbol{z}'$ and follows the steps in Lemma 6 to recover $\boldsymbol{y}$ from $\boldsymbol{y}'$.

(iii) **If the length of $\boldsymbol{z}'$ is** $(N+1)$, we conclude that a single insertion has occurred. Suppose $\boldsymbol{z}' = (z_1', \ldots, z_{N+1}')$. The decoder proceeds as follows.

- **Localizing the insertion.** If z_{n+1}' and z_{n+2}' are identical, the decoder sets $\boldsymbol{y}$ as the first n symbols of $\boldsymbol{z}'$. On the other hand, if z_{n+1}' and z_{n+2}' are different, the decoder sets $\boldsymbol{y}'$ as the first $(n+1)$ symbols of $\boldsymbol{z}'$ and there is no error in $\boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3$, and $\boldsymbol{R}_4$.
- **Recovering $\boldsymbol{y}$.** The decoder follows the steps in Lemma 6 to recover $\boldsymbol{y}$ from $\boldsymbol{y}'$.

It is easy to see that all the decoding steps run in $O(n)$ time. ■

Since the posting of our preprint on arXiv, a group at Stanford University written an implemention of the encoder described in this subsection. The open-source implementation is available here [27].

C. Edit Error in DNA Storage Channel

There are recent works that characterize the error probabilities by analyzing data from certain experiments [12], [22]. Specifically, Heckel *et al.* [12] studied the substitution errors and computed conditional error probabilities for mistaking a certain nucleotide for another. They also compared their data with experiments from other research groups and observed that the probabilities of mistaking T for a C ($\mathtt{T} \to \mathtt{C}$) and A for a G ($\mathtt{A} \to \mathtt{G}$) are significantly higher than substitution probabilities.

Motivated by the study, we consider an alternative error model where substitution errors only occur between certain nucleotides. We refer this error as a *nucleotide edit*. The edits that we define earlier is referred as *general edit*. In the nucleotide-edit model, besides a single deletion, insertion, a substitution happens only when

$$\mathtt{A} \to \{\mathtt{C},\mathtt{G}\}, \mathtt{T} \to \{\mathtt{C},\mathtt{G}\}, \mathtt{C} \to \{\mathtt{A},\mathtt{T}\}, \text{ and } \mathtt{G} \to \{\mathtt{A},\mathtt{T}\}.$$

Now, recall that the one-to-one correspondence between $\mathcal{D} = \{\mathtt{A},\mathtt{T},\mathtt{C},\mathtt{G}\}$ and two-bit sequences is:

$$\mathtt{A} \leftrightarrow 00, \quad \mathtt{T} \leftrightarrow 01, \quad \mathtt{C} \leftrightarrow 10, \quad \mathtt{G} \leftrightarrow 11.$$

Then a nucleotide-edit (substitution) in a symbol occurs if and only if the corresponding first bit is flipped.

Example 7: Suppose that $\boldsymbol{\sigma} = \mathtt{ACAGTG}$. Suppose a substitution error occurs at the third symbol, changing A to C, and we received $\boldsymbol{\sigma}_1 = \mathtt{ACCGTG}$. On the other hand, suppose a substitution error also occurs at the third symbol, changing A to G, and we received $\boldsymbol{\sigma}_2 = \texttt{ACGGTG}$. We then see that there is exactly one substitution in $\boldsymbol{U}_{\boldsymbol{\sigma}}$ which is also at the third symbol.

$\boldsymbol{\sigma}$	A	C	A	G	T	G
$\boldsymbol{U}_{\boldsymbol{\sigma}}$	0	1	0	1	0	1
$\boldsymbol{L}_{\boldsymbol{\sigma}}$	0	0	0	1	1	1

$\boldsymbol{\sigma}_1$	A	C	C	G	T	G
$\boldsymbol{U}_{\boldsymbol{\sigma}}$	0	1	1	1	0	1
$\boldsymbol{L}_{\boldsymbol{\sigma}}$	0	0	0	1	1	1

$\boldsymbol{\sigma}_2$	A	C	G	G	T	G
$\boldsymbol{U}_{\boldsymbol{\sigma}}$	0	1	1	1	0	1
$\boldsymbol{L}_{\boldsymbol{\sigma}}$	0	0	1	1	1	1

We now constructed order-optimal nucleotide-edit-correcting codes $\mathcal{C}^{\rm nt}(n; a, b, c)$ for DNA storage as follows.

Construction 4: For given $n, r, P > 0, P \geq r + 1$, $a \in \mathbb{Z}_n, b \in \mathbb{Z}_P, c \in \mathbb{Z}_2$, let $\mathcal{C}^{\rm nt}_{a,b,c}(n, r, P)$ be the set of all DNA strands $\boldsymbol{\sigma}$ of length n satisfying the following constraints.

- **Upper-array constraints**.
  - The upper-array $\boldsymbol{U}_{\boldsymbol{\sigma}}$ is a codeword in $\mathrm{L}_a(n)$.
  - The longest run of 0's or 1's in $\boldsymbol{U}_{\boldsymbol{\sigma}}$ is at most r.
- **Lower-array constraint.** The lower array $\boldsymbol{L}_{\boldsymbol{\sigma}}$ is a codeword in $\mathrm{SVT}_{b,c,P}(n)$.

Theorem 8: The code $\mathcal{C}^{\rm nt}_{a,b,c}(n, r, P)$ corrects a single nucleotide edit in linear time.

Proof: Suppose $\boldsymbol{\sigma}'$ is the received word. According to Proposition 1, $\boldsymbol{\sigma}' \in \mathcal{B}^{\rm edit}(\boldsymbol{\sigma})$ implies that $\boldsymbol{U}_{\boldsymbol{\sigma}'} \in \mathcal{B}^{\rm edit}(\boldsymbol{U}_{\boldsymbol{\sigma}})$ and $\boldsymbol{L}_{\boldsymbol{\sigma}'} \in \mathcal{B}^{\rm edit}(\boldsymbol{L}_{\boldsymbol{\sigma}})$. Since $\boldsymbol{U}_{\boldsymbol{\sigma}} \in \mathrm{L}_a(n)$, where $\mathrm{L}_a(n)$ can correct a single edit, we can recover $\boldsymbol{U}_{\boldsymbol{\sigma}}$ in linear time. For $\boldsymbol{L}_{\boldsymbol{\sigma}}$, we have two cases.

- If $\boldsymbol{\sigma}'$ is of length $n-1$ (or $n+1$), then a deletion or insertion has occurred. Now, we are able to identify the location of deletion (or insertion) in $\boldsymbol{U}_{\boldsymbol{\sigma}}$, which belongs to a run of length at most r. Hence, we can locate the error in $\boldsymbol{L}_{\boldsymbol{\sigma}}$ within $(r+1)$ positions. Since $\boldsymbol{L}_{\boldsymbol{\sigma}} \in \mathrm{SVT}_{b,c,P}(n)$ with $P \geq r+1$, we are able to recover uniquely $\boldsymbol{L}_{\boldsymbol{\sigma}}$.
- If $\boldsymbol{\sigma}'$ is of length n, then a nucleotide substitution has occurred. Therefore, it is necessary that a bit flip occur in the $\boldsymbol{U}_{\boldsymbol{\sigma}}$ and hence, we can locate the error. To correct the corresponding position in $\boldsymbol{L}_{\boldsymbol{\sigma}}$, we simply make use of the syndrome in $\mathrm{SVT}_{b,c,P}(n)$. ■

Next, we present an efficient encoder that maps messages into $\mathcal{C}^{\rm nt}_{a,b,c}(n, r, P)$.

Nucleotide-Edit-Encoder. Given $n, r = 2\lceil \log n \rceil + 4, P = r + 1, a \in \mathbb{Z}_n, b \in \mathbb{Z}_P, c \in \mathbb{Z}_2$, set $t \triangleq \lceil \log P \rceil + 1$ and $m \triangleq 2n - \lceil \log n \rceil - t - 2$.

INPUT: $\boldsymbol{x} \in \{0,1\}^m$

OUTPUT: $\boldsymbol{\sigma} \triangleq \mathrm{ENC}^{\rm nt}_{\mathbb{E}}(\boldsymbol{x}) \in \mathcal{C}^{\rm nt}_{a,b,c}(n, r, P)$

(I) Set $\boldsymbol{x}_1$ be the first $(n - \lceil \log n \rceil - 2)$ bits in $\boldsymbol{x}$ and $\boldsymbol{x}_2$ be the last $(n-t)$ bits in $\boldsymbol{x}$.

(II) Let $\boldsymbol{y}_1' = \mathrm{ENC}_{RLL}(\boldsymbol{x}_1)$ and use Encoder $\mathbb{L}$ as described in Subsection II-C to encode $\boldsymbol{y}_1 = \mathrm{ENC}_{\mathbb{L}}(\boldsymbol{y}_1') \in \mathrm{L}_a(n)$.

(III) Use SVT-Encoder ENC_{SVT} to encode $\boldsymbol{y}_2 = \mathrm{ENC}_{SVT}(\boldsymbol{x}_2) \in \mathrm{SVT}_{b,c,P}(n)$.

(IV) Finally, set $\boldsymbol{y} = \boldsymbol{y}_1 || \boldsymbol{y}_2$ and output $\boldsymbol{\sigma} \triangleq \Psi^{-1}(\boldsymbol{y})$.

Proposition 6: The Nucleotide-Edit-Encoder is correct. In other words, $\mathrm{ENC}^{\rm nt}_{\mathbb{E}}(\boldsymbol{x}) \in \mathcal{C}^{\rm nt}_{a,b,c}(n, r, P)$ for all $\boldsymbol{x} \in \{0,1\}^m$. The redundancy of our encoder is $\log n + \log\log n + O(1)$.

Proof: Let $\boldsymbol{\sigma} \triangleq \mathrm{ENC}^{\rm nt}_{\mathbb{E}}(\boldsymbol{x})$. Based on our encoder, $\boldsymbol{U}_{\boldsymbol{\sigma}} = \boldsymbol{y}_1$ and $\boldsymbol{y}_1 = \mathrm{ENC}_{\mathbb{L}}(\boldsymbol{y}_1') \in \mathrm{L}_a(n)$. In addition, $\boldsymbol{y}_1' = \mathrm{ENC}_{RLL}(\boldsymbol{x}_1)$, which implies the longest run of 0's or 1's in $\boldsymbol{y}_1'$ is at most $(\lceil \log n \rceil + 3)$ according to Theorem 3. Therefore, the maximum run in $\boldsymbol{U}_{\boldsymbol{\sigma}}$ after $\mathrm{ENC}_{\mathbb{L}}$ is at most $(\lceil \log n \rceil + 3) + (\lceil \log n \rceil + 1) = 2\lceil \log n \rceil + 4 = r$. Since $P = r + 1$, it satisfies the upper-array constraints in Construction 4. On the other hand, based on our encoder, $\boldsymbol{L}_{\boldsymbol{\sigma}} = \boldsymbol{y}_2$ and $\boldsymbol{y}_2 = \mathrm{ENC}_{SVT}(\boldsymbol{x}_2) \in \mathrm{SVT}_{b,c,P}(n)$. It implies $\boldsymbol{L}_{\boldsymbol{\sigma}}$ also satisfies the lower-array constraint in Construction 4. Therefore, $\mathrm{ENC}^{\rm nt}_{\mathbb{E}}(\boldsymbol{x}) \in \mathcal{C}^{\rm nt}_{a,b,c}(n, r, P)$ for all $\boldsymbol{x} \in \{0,1\}^m$. The redundancy of our encoder is $\lceil \log n \rceil + 2 + \lceil \log(2\lceil \log n \rceil + 5) \rceil + 1 = \log n + \log\log n + O(1)$. ■

For completeness, we state the corresponding Nucleotide-Edit-Decoder, $\mathrm{DEC}^{\rm nt}_{\mathbb{E}}(\boldsymbol{\sigma})$, for DNA codes that correct a single nucleotide edit.

Nucleotide-Edit-Decoder. For given $n, a, b, c, r = 2\lceil \log n \rceil + 4, P = r + 1, t = \lceil \log P \rceil + 1$, set $m = 2n - \lceil \log n \rceil - t - 2$.

INPUT: $\boldsymbol{\sigma} \in \mathcal{D}^{n*}$

OUTPUT: $\boldsymbol{x} = \mathrm{DEC}^{\rm nt}_{\mathbb{E}}(\boldsymbol{\sigma}) \in \{0,1\}^m$

(I) Compute $\hat{\boldsymbol{y}}_1 = \boldsymbol{U}_{\boldsymbol{\sigma}}$ and $\hat{\boldsymbol{y}}_2 = \boldsymbol{L}_{\boldsymbol{\sigma}}$.

(II) Compute $\boldsymbol{y}_1' \triangleq \mathrm{DEC}^L_a(\hat{\boldsymbol{y}}_1)$ to correct a single edit in $\hat{\boldsymbol{y}}_1$.

(III) Compute $\boldsymbol{y}_2 \triangleq \mathrm{DEC}^{SVT}_{b,c,P}(\hat{\boldsymbol{y}}_2)$ of length $(n-t)$.

(IV) Compute $\boldsymbol{y}_1 = \mathrm{DEC}_{RLL}(\boldsymbol{y}_1')$ of length $(n - \lceil \log n \rceil - 2)$.

(V) Output $\boldsymbol{y}_1\boldsymbol{y}_2$ of length $m = 2n - \lceil \log n \rceil - t - 2$.

V. GC-Balanced Encoder Correcting Single Edit

We modify the single-edit-correcting Encoder $\mathbb{L}$ in Section III to obtain a GC-balanced single-edit-correcting encoder. Our modification makes use of the celebrated Knuth's balancing technique [18].

Knuth's balancing technique is a linear-time algorithm that maps a binary message $\boldsymbol{x}$ to a balanced word $\boldsymbol{z}$ of the same length by flipping the first k bits of $\boldsymbol{x}$. The crucial observation demonstrated by Knuth is that such an index k always exists and k is commonly referred to as the *balancing index*. Formally, we have the following theorem.

Theorem 9 (Knuth [18]): There exists a pair of linear-time algorithms $\mathrm{ENC}^K : \{0,1\}^n \to \{0,1\}^n \times [n]$ and $\mathrm{DEC}^K : \{0,1\}^n \times [n] \to \{0,1\}^n$ such that the following holds. If $\mathrm{ENC}^K(\boldsymbol{x}) = (\boldsymbol{z}, k)$, then $\boldsymbol{z}$ is balanced and $\mathrm{DEC}^K(\boldsymbol{z}, k) = \boldsymbol{x}$.

To represent the balancing index, Knuth appends $\boldsymbol{z}$ with a short balanced suffix of length $\lceil \log n \rceil$ and so, a lookup table of size n is required. In constrast, since we only require the upper sequence $\boldsymbol{U}_{\boldsymbol{\sigma}}$ to be balanced, we simply store the balancing index k in the lower sequence $\boldsymbol{L}_{\boldsymbol{\sigma}}$. Consequently, we do not need a look up table for the balancing indices.

GC-Balanced Encoder. Given n, set $t = \lceil \log n \rceil$ and $m = 2n - 3t - 2$.

INPUT: $\boldsymbol{x} \in \{0,1\}^n$, $\boldsymbol{y} \in \{0,1\}^{n-3t-2}$ and so, $\boldsymbol{x}\boldsymbol{y} \in \{0,1\}^m$

OUTPUT: $\boldsymbol{\sigma} = \mathrm{ENC}_{\texttt{GC}}(\boldsymbol{x}\boldsymbol{y})$

(I) Apply Knuth's balancing technique to obtain $(\boldsymbol{z}, k) \triangleq \textsc{Enc}^K(\boldsymbol{x})$.
(II) Compute $d \triangleq \mathrm{Syn}(\boldsymbol{z}) \pmod{2n}$ and let $\boldsymbol{d}$ be the binary representation of d of length $t+1$. On the other hand, let $\boldsymbol{k}$ be the binary representation of the balancing index k of length t.
(III) Next, we append $\boldsymbol{d}$ and $\boldsymbol{k}$ to $\boldsymbol{y}$ to obtain a binary word of length $n-t-1$. Using Encoder $\mathbb{L}$, we compute $\boldsymbol{c} \triangleq \textsc{Enc}_{\mathbb{L}}(\boldsymbol{ydk})$.
(IV) Finally, we set $\boldsymbol{\sigma} \triangleq \Psi^{-1}(\boldsymbol{z}||\boldsymbol{c})$.

Example 8: Consider $n = 16$ and $a = 0$. So, $t = 4$ and $m = 2n - 3t - 2 = 18$. Let $\boldsymbol{x} = 1111111100001111$ and $\boldsymbol{y} = 01$, and so the message is 111111110000111101.
(I) Knuth's method yields $\boldsymbol{z} = 0000111100001111$ with index $k = 4$.
(II) So, $d = \mathrm{Syn}(\boldsymbol{z}) = 20 \pmod{2n}$. The binary representations of d and k are $\boldsymbol{d} = 10100$ and $\boldsymbol{k} = 0100$, respectively.
(III) Hence, we have $\boldsymbol{ydk} = 01101000100$ and use Encoder 1 to compute $\boldsymbol{c} = \textsc{Enc}_{\mathbb{L}}(\boldsymbol{ydk}) = 1101110110001001$.
(IV) So, the DNA codeword is $\boldsymbol{\sigma} =$ TTATGGCGTAAAGCCG.

To demonstrate that the map $\textsc{Enc}_{\texttt{GC}}$ corrects a single edit, we provide an explicit decoding algorithm.

GC-**Balanced Decoder**. For any n, set $m = 2n - 3\lceil \log n \rceil - 2$.
INPUT: $\boldsymbol{\sigma} \in \mathcal{D}^{n*}$
OUTPUT: $\boldsymbol{xy} = \textsc{Dec}_{\texttt{GC}}(\boldsymbol{\sigma}) \in \{0,1\}^m$
(I) Compute $\hat{\boldsymbol{z}} = \boldsymbol{U}_{\boldsymbol{\sigma}}$ and $\hat{\boldsymbol{c}} = \boldsymbol{L}_{\boldsymbol{\sigma}}$.
(II) Compute $\boldsymbol{c} \triangleq \textsc{Dec}_a^L(\hat{\boldsymbol{c}})$ to correct a single edit in $\hat{\boldsymbol{c}}$.
(III) Remove the suffices $\boldsymbol{d}$ and $\boldsymbol{k}$ to obtain $\boldsymbol{y}$ and figure out the syndrome d and balancing index k.
(IV) Using the syndrome d, compute $\boldsymbol{z} \triangleq \textsc{Dec}_d^L(\hat{\boldsymbol{z}})$ to correct a single edit in $\hat{\boldsymbol{z}}$.
(V) Using the index k, compute $\boldsymbol{x} \triangleq \textsc{Dec}^K(\boldsymbol{z}, k)$.

Theorem 10: The map $\textsc{Enc}_{\texttt{GC}}$ is a GC-balanced single-edit-correcting encoder with redundancy $3\lceil \log n \rceil + 2$.

Proof: The GC-Balanced Decoder shows that the GC-Balanced Encoder corrects a single edit. Hence, it remains to verify that any codeword $\boldsymbol{\sigma} = \textsc{Enc}_{\texttt{GC}}(\boldsymbol{xy})$ is GC-balanced. This follows from the fact that $\boldsymbol{U}_{\boldsymbol{\sigma}} = \boldsymbol{z}$ is the binary balanced word obtained from balancing $\boldsymbol{x}$. ■

VI. Conclusion

We designed order-optimal quaternary encoders for codes that correct either a single indel or a single edit. The encoders map binary messages into quaternary codes in linear-time and the redundancy is at most $\log n + O(\log \log n)$. Moreover, in the case for indel, our encoder uses only $\lceil \log n \rceil + 2$ redundant bits, improving the best known encoder of Tenengolts (1984) by at least four bits. We also present linear-time encoders for quaternary codes that correct a single nucleotide edit and GC-balanced codes that can correct a single edit.

To conclude, we discuss open problems and possible directions of research.

(i) *Provide a construction of nonbinary single-edit-correcting codes with $\log n + O(1)$ redundant bits.* While Theorem 6 provides a family of single-edit-correcting codes with $\log n + \Theta(\log \log n)$ redundant bits, the sphere-packing bound states that $\log n + \Omega(1)$ redundant bits are necessary. It remains open to determine whether there exists codes with $\log n + O(1)$ redundant bits.
(ii) *Extend the linear-time encoders for codes correcting multiple deletions.* Recently, Sima *et al.* [26] constructed q-ary t-deletions-correcting codes with $4t \log n + \Theta(\log \log n)$ redundant bits for $t \geqslant 2$ and fixed values of q. In the same work, the authors also provided an efficient encoder for these codes and the method essentially involves appending two syndromes of lengths $4t \log n + O(1)$ and $\Theta(\log \log n)$ to the message. However, these codes have encoding/decoding complexity $O(n^{2t})$ so that the constructions admit polynomial-time encoders/decoders only for the case where t is a constant with respect to n. In contrast, in our work, the encoders $\textsc{Enc}_{\mathbb{L}}$, $\textsc{Enc}_{\mathbb{I}}$ and $\textsc{Enc}_{\mathrm{SVT}}$ insert only one syndrome at *specially chosen* positions. A natural question is whether the linear-time encoders can be modified and adapted for the codes in [26] so that the encoder uses only $4t \log n + O(1)$ redundant bits.
(iii) *Extend the linear-time encoders for codes correcting multiple edits*. Particularly, in the segmented edit model, we have the additional assumption that the channel input is divided into segments and that at most one edit can occur within a segment [4], [21]. Our encoders, namely, $\textsc{Enc}_{\mathbb{I}}$ and $\textsc{Enc}_{\mathbb{E}}^A$, can be extended for this model and a detailed discussion of the construction is deferred this to our future research work. In addition, our proposed single edit correcting codes can also be used for recovering several edits in some modern storage devices that provide users with *multiple reads* (see [2], [7], [11]).

Acknowledgment

The authors would like to thank the editor and the reviewers for their constructive feedback and helpful suggestions.

References

[1] K. A. S. Abdel-Ghaffar and H. C. Ferreira, "Systematic encoding of the Varshamov-Tenengol'ts codes and the Constantin-Rao codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 340–345, Jan. 1998.
[2] M. Abroshan, R. Venkataramanan, L. Dolecek, and A. G. i Fàbregas, "Coding for deletion channels with multiple traces," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 1372–1376.
[3] M. Abroshan, R. Venkataramanan, and A. G. I. Fabregas, "Efficient systematic encoding of non-binary VT codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 91–95.
[4] M. Abroshan, R. Venkataramanan, and A. G. i Fabregas, "Coding for segmented edit channels," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3086–3098, Apr. 2018.
[5] Y. M. Chee, H. M. Kiah, and T. T. Nguyen, "Linear-time encoders for codes correcting a single edit for DNA-based data storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 772–776.
[6] K. Cai, X. He, H. M. Kiah, and T. T. Nguyen, "Efficient constrained encoders correcting a single nucleotide edit in DNA storage," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Barcelona, Spain, May 2020, pp. 8827–8830.
[7] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Trans. Inf. Theory*, vol. 66, no. 10, pp. 6084–6103, Oct. 2020.
[8] S. Clancy, "Genetic mutation," *Nature Edu.*, vol. 1, no. 1, p. 187, 2008.

[9] O. Elishco, R. Gabrys, M. Medard, and E. Yaakobi, "Repeat-free codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 932–936.

[10] R. Gabrys, E. Yaakobi, and O. Milenkovic, "Codes in the Damerau distance for deletion and adjacent transposition correction," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2550–2570, Apr. 2018.

[11] H. M. Kiah, T. T. Nguyen, and E. Yaakobi, "Coding for sequence reconstruction for single edits," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 676–681.

[12] R. Heckel, G. Mikutis, and R. N. Grass, "A characterization of the DNA data storage channel," *Sci. Rep.*, vol. 9, no. 1, pp. 1–12, Jul. 2019.

[13] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Stat. Assoc.*, vol. 58, no. 301, pp. 13–30, Mar. 1963.

[14] 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, 2020.

[15] K. S. Immink and J. Weber, "Very efficient balanced codes," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 188–192, Feb. 2010.

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

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

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

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

[20] V. I. Levenshtein, "Asymptotically optimum binary code with correction for losses of one or two adjacent bits," (in Russian), *Problemy Kibernetiki*, vol. 19, pp. 293–298, 1967.

[21] Z. Liu and M. Mitzenmacher, "Codes for deletion and insertion channels with segmented errors," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 224–232, Jan. 2010.

[22] L. Organick *et al.*, "Random access in large-scale DNA data storage," *Nature Biotechnol.*, vol. 36, no. 3, pp. 242–248, 2018.

[23] M. G. Ross *et al.*, "Characterizing and measuring bias in sequence data," *Genome Biol.*, vol. 14, no. 5, p. R51, 2013.

[24] K. Saowapa, H. Kaneko, and E. Fujiwara, "Systematic deletion/insertion error correcting codes with random error correction capability," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst. (EFT)*, Albuquerque, NM, USA, Nov. 1999, pp. 284–292.

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

[26] J. Sima, R. Gabrys, and J. Bruck, "Optimal codes for the q-ary deletion channel," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 740–745.

[27] D. C. H. Tan. *Single Edit Correcting Code*. Accessed: Aug. 13, 2020. [Online]. Available: https://github.com/dtch1997/single-edit-correcting-code

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

[29] J. H. Weber and K. A. S. Immink, "Knuth's balancing of codewords revisited," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1673–1679, Jul. 2010.

[30] A. Van Wijngaarden and K. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 200–207, Feb. 2010.

[31] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors," *Avtomatika Telemekhanika*, vol. 26, no. 2, pp. 288–292, 1965.

[32] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 1, pp. 1–6, Dec. 2017.

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

[34] P. Yakovchuk, "Base-stacking and base-pairing contributions into thermal stability of the DNA double helix," *Nucleic Acids Res.*, vol. 34, no. 2, pp. 564–574, Jan. 2006.

Kui Cai (Senior Member, IEEE) received the B.E. degree in information and control engineering from Shanghai Jiao Tong University, Shanghai, China, the M.Eng. degree in electrical engineering from the National University of Singapore, and the joint Ph.D. degree in electrical engineering from the Technical University of Eindhoven, The Netherlands, and the National University of Singapore. She is currently an Associate Professor with the Singapore University of Technology and Design (SUTD). Her main research interests include coding theory, information theory, and signal processing for various data storage systems and digital communications. She received the 2008 IEEE Communications Society Best Paper Award in Coding and Signal Processing for Data Storage. She has served as the Vice-Chair (Academia) for IEEE Communications Society and the Data Storage Technical Committee (DSTC) in 2015 and 2016.

Yeow Meng Chee (Senior Member, IEEE) received the B.Math. degree in computer science and combinatorics and optimization and the M.Math. and Ph.D. degrees in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1988, 1989, and 1996, respectively. He was a Professor with the School of Physical and Mathematical Sciences and the Interim Dean of science with Nanyang Technological University. He is currently a Professor with the Department of Industrial Systems Engineering and Management and the Associate Vice President of innovation and enterprise with the National University of Singapore. His research interests include the interplay between combinatorics and computer science/engineering, particularly combinatorial design theory, coding theory, extremal set systems, and electronic design automation.

Ryan Gabrys (Member, IEEE) is currently the Scientist of the Naval Information Warfare Center Pacific. His research interest includes theoretical computers science with applications to cyber security and information storage.

Han Mao Kiah (Member, IEEE) received the Ph.D. degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2014. From 2014 to 2015, he was a Post-Doctoral Research Associate with the Coordinated Science Laboratory, University of Illinois at Urbana–Champaign. From 2015 to 2018, he was a Lecturer with the School of Physical and Mathematical Sciences (SPMS), NTU, where he is currently an Assistant Professor. His research interests include DNA-based data storage, coding theory, enumerative combinatorics, and combinatorial design theory.

Tuan Thanh Nguyen (Member, IEEE) received the B.Sc. and Ph.D. degrees in mathematics from Nanyang Technological University (NTU), Singapore, in 2014 and 2018, respectively. He was a Research Fellow with the School of Physical and Mathematical Sciences, NTU, from August 2018 to September 2019. He is currently a Research Fellow with the Singapore University of Technology and Design. His research interests include error correction codes and constrained codes for communication systems and data storage systems, especially codes for DNA-based data storage.