Error-correcting Codes for Short Tandem Duplication and Substitution Errors

Yuanyuan Tang and Farzad Farnoud
Electrical & Computer Engineering, University of Virginia, {yt5tz, farzad}@virginia.edu

***Abstract*—Due to its high data density and longevity, DNA is considered a promising storage medium for satisfying ever-increasing data storage needs. However, the diversity of errors that occur in DNA sequences makes efficient error-correction a challenging task. This paper aims to address simultaneously correcting two types of errors, namely, short tandem duplication and substitution errors. We focus on tandem repeats of length at most 3 and design codes for correcting an arbitrary number of duplication errors and one substitution error. Because a substituted symbol can be duplicated many times (possibly as part of longer substrings), a single substitution can affect an unbounded substring of the retrieved word. However, we show that with appropriate preprocessing, the effect may be limited to a substring of finite length, thus making efficient error-correction possible. We construct a code for correcting the aforementioned errors and provide lower bounds for its rate. In particular, compared to optimal codes correcting only duplication errors, numerical results show that the asymptotic cost of protecting against an additional substitution is only 0.003 bits/symbol when the alphabet has size 4, an important case corresponding to data storage in DNA.**

I. Introduction

Recent advances in DNA synthesis and sequencing technologies [1] have made DNA a promising candidate for rising data storage needs. Compared to traditional storage media, DNA storage has many advantages, including higher data density, longevity, and ease of generating copies [1]. However, DNA is subject to a diverse set of errors that may occur during the various stages of data storage and retrieval, including substitutions, duplications, insertions, and deletions. This poses a challenge to the design of error-correcting codes and has led to many recent works studying the subject, including [1]–[8], among others. The current paper focuses on correcting short duplication and substitution errors.

A (tandem) duplication error generates a copy of a substring of the DNA sequence and inserts it after the original substring [2]. For example, from ACGT we may obtain ACG<u>CG</u>T. The length of the duplication is the length of the substring being copied, which is 2 in the preceding example. In the literature, both fixed-length duplication [2]–[5], [9] and variable-length duplication, where the duplication length is bounded from above [2], [10]–[12] have been studied. For duplications whose length is at most 3, [2] proposed error-correcting codes that were shown to have an asymptotically optimal rate by [11].

In a substitution event, a symbol in the sequence is changed to another alphabet symbol. Substitution errors may be restricted to the inserted copies, reflecting the noisiness of the copying that occur during the duplication process [13], [14] or be unrestricted. For fixed-length duplication, these settings have been studied in [5], [15].

This work was supported in part by NSF-CCF grants under grant nos. 1816409 and 1755773.

We focus on correcting errors that may arise from channels with many duplication errors of length at most 3, which we refer to as *short duplications*, and one unrestricted substitution error. We restrict ourselves to a single substitution error as a first step towards the general case of t substitution errors. As a simple example, in a channel with many short duplications and one substitution, the input ACG may become ACTCTACTACTCG, where the occurrences of the symbol T result from copies of a substitution of the form C $\rightarrow$ T. Given that an arbitrary number of duplications are possible, an unbounded segments of the output word may be affected by the errors and the incorrect substituted symbol may appear many times. We show, however, with an appropriate construction and preprocessing of the output of the channel, the deleterious effects of the errors may be localized. We then use a *maximum distance separation* (MDS) code to correct the errors. We establish a lower bound on the code rate and provide an asymptotic analysis that shows that the code has rate at least $\log(q-2)$, where q is the size of the alphabet and the $\log$ is in base 2. We note that the rate of the code correcting only short duplications is upper bounded by $\log(q-1)$. When $q=4$, the case corresponding to DNA storage, we provide a computational bound for the code rate, showing that, asymptotically, its rate is only 0.003 bits/symbol smaller compared to the code that corrects short duplications but no substitutions.

The paper is organized as follows. In Section II, we provide the notation and relevant background. Section III analyzes the errors patterns that result from passing through the duplication and substitution channels. Finally, code construction as well as the code size are presented in Section IV. Due to lack of space, some of the proofs are omitted or only sketched.

II. Notation and Preliminaries

Let $\Sigma_q = \{0, 1, \ldots, q-1\}$ denote a finite alphabet of size q. The set of all strings of finite length over Σ_q is denoted by Σ_q^*, while Σ_q^n represents the strings of length n. In particular, Σ_q^* contains the empty string Λ. Let $[n]$ denote the set $\{1, \ldots, n\}$.

Bold symbols, such as $\boldsymbol{x}$ and $\boldsymbol{y}_j$, are used to denote strings over Σ_q. The elements of strings are shown with plain typeface, e.g., $\boldsymbol{x} = x_1 x_2 \cdots x_n$ and $\boldsymbol{y}_j = y_{j1} y_{j2} \cdots y_{jm}$, where $x_i, y_{ji} \in \Sigma_q$. Given two strings $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, $\boldsymbol{xy}$ denotes the concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$, and $\boldsymbol{x}^m$ denotes the concatenation of m copies of $\boldsymbol{x}$. Let $|\boldsymbol{x}|$ denote the length of a word $\boldsymbol{x} \in \Sigma_q^*$. For four words $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \Sigma_q^*$, if $\boldsymbol{x}$ can be expressed as $\boldsymbol{x} = \boldsymbol{uvw}$, then $\boldsymbol{v}$ is a substring of $\boldsymbol{x}$.

Given a word $\boldsymbol{x} \in \Sigma_q^*$, a *tandem duplication* (TD) of length k copies a substring of length k and inserts it after the original. This is referred to as a k-TD. For example, a 2-TD may generate $abcbcde$ from $abcde$. Here, $bcbc$ is called a *(tandem) repeat* of length 2. Our focus in this paper is on TDs of length bounded by k, denoted $\leq k$-TD, for $k = 3$. For example, from $\boldsymbol{x} = 1201210$ we may obtain

$$\begin{aligned} \boldsymbol{x} =& 1201210 \to 1201\underline{201}210 \to \\ & 120120\underline{20}1210 \to 1201202\underline{2}01210 = \boldsymbol{x}', \end{aligned} \tag{1}$$

where the underlined substrings are the inserted copies. We say that $\boldsymbol{x}'$ is a *descendant* of $\boldsymbol{x}$, i.e., a sequence resulting from $\boldsymbol{x}$ through a sequence of duplications.

Let $D^*_{\leq k}(\boldsymbol{x})$ denote the *descendant cone* of $\boldsymbol{x}$ containing all descendants of $\boldsymbol{x}$ after an arbitrary number of $\leq k$-TDs. Given a string $\boldsymbol{x}$, let $R_{\leq k}(\boldsymbol{x})$ denotes the set of *duplication roots* of $\boldsymbol{x}$, i.e., sequences $\boldsymbol{r}$ containing no repeats of length $\leq k$ such that $\boldsymbol{x} \in D^*_{\leq k}(\boldsymbol{r})$. For a set S of strings, $R_{\leq k}(S)$ is the set of strings each of which is a root of at least one string in S. If $R_{\leq k}(\cdot)$ is a singleton, we may view it as a string rather than a set. A root can be obtained from $\boldsymbol{x}$ by repeatedly replacing all repeats of the form $\boldsymbol{aa}$ with $\boldsymbol{a}$, where $|\boldsymbol{a}| \leq k$ (each such operation is called a *deduplication*). For $\leq$ 3-TDs, the duplication root is unique [2]. If $\boldsymbol{x}'$ is a descendant of $\boldsymbol{x}$, we have $R_{\leq 3}(\boldsymbol{x}) = R_{\leq 3}(\boldsymbol{x}')$. Finally, let $\mathrm{Irr}_{\leq k}(n) \subseteq \Sigma_q^n$ denote the set of *irreducible strings* (more precisely, $\leq k$-irreducible strings) of length n, i.e., strings without repeats of length at most k. We observe that $R_{\leq 3}(\boldsymbol{x}) \in \mathrm{Irr}_{\leq 3}(*)$, where $\mathrm{Irr}_{\leq 3}(*)$ denotes $\leq$ 3-irreducible strings of arbitrary lengths. For $k = 3$, we may drop the ≤ 3 subscript and denote these entities as $D^*(\cdot), R(\cdot), \mathrm{Irr}(\cdot)$.

We also consider substitution errors, although our attention is limited to at most one error of this kind. Continuing the example given in (1), a substitution occurring in the descendant $\boldsymbol{x}'$ of $\boldsymbol{x}$ may result in $\boldsymbol{x}''$:

$$\boldsymbol{x}' = 1201202201210 \to \boldsymbol{x}'' = 1201202\underline{1}01210,$$

We denote by $D^{t,p}_{\leq k}(\boldsymbol{x})$ the set of strings that can be obtained from $\boldsymbol{x}$ through t TDs of length at most k and p substitutions, *in any order*. Replacing t with $*$ denotes any number of $\leq k$-TDs and replacing p with $\leq p$ denotes at most p substitutions. We again drop $\leq k$ from the notation when $k = 3$. In the example above, $\boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x})$.

III. Channels with many $\leq$ 3-TDs and one substitution error

In this section, we study channels that alter the input string by applying an arbitrary number of duplication errors and at most one substitution error, where the substitution may occur at any point in the sequence of errors. We will first study the conditions a code must satisfy to be able to correct such errors. Then, we will investigate the effect of such channels on the duplication root of sequences, which is an important aspect of designing our error-correcting codes.

Figure 1. Finite automaton for the regular language $D^*(012)$ [10].

Table I
Paths representing irreducible strings starting from and ending at specific states.

state	Paths from Start to state	Paths from state to S_3
S_1	0	012, 1012, 12, 12012,
S_2	01, 01201	012,1012, 12, 12012, 2, 2012, 212, 212012
S_3	012	012, 02012, 12, 12012, 2, 2012, 212, 212012
S_4	0120	012, 02012, 1012, 12, 12012, 2012
T_2	010, 012010	012, 1012,12, 12012
T_3	0121	12, 12012, 2, 2012, 212, 212012
T_4	01202	012, 02012, 2012

A code C is able to correct many $\leq$ 3-TDs and a substitution if and only if for any two distinct codewords $\boldsymbol{c}_1, \boldsymbol{c}_2 \in C$, we have $D^{*,\leq 1}(\boldsymbol{c}_1) \cap D^{*,\leq 1}(\boldsymbol{c}_2) = \varnothing$, to satisfy which it is sufficient to have

$$R(D^{*,\leq 1}(\boldsymbol{c}_1)) \cap R(D^{*,\leq 1}(\boldsymbol{c}_2)) = \varnothing. \tag{2}$$

Condition (2) implies that for distinct codewords $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, $R(\boldsymbol{c}_1) \neq R(\boldsymbol{c}_2)$. This latter condition is in fact sufficient for correcting only $\leq$ 3-TDs since this type of error does not alter the duplication root. For correcting only $\leq$ 3-TDs, defining the code as the set of irreducible strings of a given length leads to asymptotically optimal codes [2], [11]. The decoding process is simply finding the root of the received word.

We take a similar approach to correct many $\leq$ 3-TDs and a substitution. More specifically, the proposed code C is a subset of $\leq$ 3-irreducible strings, i.e., $R(\boldsymbol{c}) = \boldsymbol{c}$ for $\boldsymbol{c} \in C$. To recover $\boldsymbol{c}$ from the received word $\boldsymbol{y}$, we find $R(\boldsymbol{y})$ and from that recover $R(\boldsymbol{c}) = \boldsymbol{c}$, as will be discussed.

We start by studying the effect of $\leq$ 3-TDs and one substitution on the root of a string. Consider a string $\boldsymbol{x}$ and let $\boldsymbol{x}'' \in D^{*,\leq 1}(\boldsymbol{x})$. We either have $\boldsymbol{x}'' \in D^*(\boldsymbol{x})$, i.e., $\boldsymbol{x}''$ suffers only duplications or $\boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x})$. In the former case $R(\boldsymbol{x}'') = R(\boldsymbol{x})$. Hence, below we consider $\boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x})$. Furthermore, duplications that occur after the substitution do not affect the root and so in our analysis we may assume that the substitution is the last error. We start by a lemma which considers a simple case.

Lemma 1. *For a string $\boldsymbol{x} \in \mathrm{Irr}(3)$,*

$$L = \max\{|R(\boldsymbol{x}'')| : \boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x})\} \leq 13.$$

Proof: Up to relabeling, the choices for $\boldsymbol{x}$ are 010 and 012. After a given sequence of $\leq$ 3-TDs and a substitution, we will obtain $\boldsymbol{x}''$. We then deduplicate all repeats to obtain $R(\boldsymbol{x}'')$. For the same sequence of errors, since any deduplication that is possible when $\boldsymbol{x} = 012$ is also possible when $\boldsymbol{x}'' = 010$, the length of the root for $\boldsymbol{x} = 010$ is not larger than when $\boldsymbol{x} = 012$. Hence, from this point on, we assume $\boldsymbol{x} = 012$.

As shown in [10], $D^*(\boldsymbol{x})$ is a regular language whose words can be described as paths from Start to S_3 in the finite automaton given in Figure 1, where the word is equal to the sequence of edge labels. Let $\boldsymbol{x}' \in D^*(\boldsymbol{x})$ and $\boldsymbol{x}'' \in D^{0,1}(\boldsymbol{x}')$. Assume $\boldsymbol{x}' = \boldsymbol{u}w\boldsymbol{z}$ and $\boldsymbol{x}'' = \boldsymbol{u}\hat{w}\boldsymbol{z}$, where $\boldsymbol{u}, \boldsymbol{z}$ are strings and w and $\hat{w}$ are distinct symbols. The string $\boldsymbol{u}$ represents a path from Start to some state U and the string $\boldsymbol{z}$ represents a path from some state Z to S_3 in the automaton, where there is an edge with label w from U to Z.

Since $R(\boldsymbol{x}'') = R(R(\boldsymbol{u})\hat{w}R(\boldsymbol{z}))$, we have $|R(\boldsymbol{x}'')| \le |R(\boldsymbol{u})| + 1 + |R(\boldsymbol{z})|$. The maximum value for $|R(\boldsymbol{u})|$ is the length of some path from Start to U such that the corresponding sequence does not have any repeats. All such paths/sequences are listed in the second column of Table I for all choices of U. Similarly, the maximum value for $|R(\boldsymbol{z})|$ is the length of some path from Z to S_3 such that the corresponding sequence does not have any repeats, all of which are listed in the third column of Table I. An inspection of the Table shows that choosing $U = T_2$ and $Z = S_2$ leads to the largest value of $|R(\boldsymbol{u})|+1+|R(\boldsymbol{z})|$, namely $6+1+6 = 13$. We note that the specific sequence achieving this length is $\boldsymbol{x}'' = 0120103212012$ which can be obtained via the sequence $\boldsymbol{x} \to 012\,\underline{012}\,\underline{012} \to 012\,01\underline{01}2\,012 \to 012\,01012\underline{12}\,012 \to \boldsymbol{x}''$, where we have combined non-overlapping duplications into a single step. ■

We now consider the roots of arbitrary strings when passed through a channel with arbitrarily many $\le$ 3-TDs and one substitution. We show that even though a substituted symbol may be duplicated many times, the effect of a substitution on the root is bounded.

Theorem 2. *Let $\boldsymbol{x} \in \Sigma^*$ and $\boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x})$. There exist $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\beta}', \boldsymbol{\gamma} \in \Sigma^*$ and a (minimal) positive integer $\mathcal{L}$ such that $R(\boldsymbol{x}) = \boldsymbol{\alpha\beta\gamma}$ and $R(\boldsymbol{x}'') = \boldsymbol{\alpha\beta}'\boldsymbol{\gamma}$ where $|\boldsymbol{\beta}|, |\boldsymbol{\beta}'| \le \mathcal{L}$ for all choices of $\boldsymbol{x}$. Furthermore, $\mathcal{L} \le 39$.*

Proof Sketch: We may assume $\boldsymbol{x}$ is irreducible. If it is not, let $\boldsymbol{x}_0 = R(\boldsymbol{x})$ so that $\boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x}) \subseteq D^{*,1}(\boldsymbol{x}_0)$. If the statement of the theorem holds for $\boldsymbol{x}_0$, it also holds for $\boldsymbol{x}$ since $R(\boldsymbol{x}) = R(\boldsymbol{x}_0)$.

The upper bounds on the length of $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ are the same. To see this, note that $\boldsymbol{\alpha\beta}'\boldsymbol{\gamma}$ is obtained from $\boldsymbol{\alpha\beta\gamma}$ by applying, in order, duplications, a single substitution, more duplications, and finally removing all repeats (performing all possible deduplications). Since duplications that occur after the substitution do not make any difference, we may instead assume that the process is as follows: duplications, substitution, deduplications. Since this process is reversible, general statements that hold for $\boldsymbol{\beta}'$ also hold for $\boldsymbol{\beta}$.

It can be shown that we can write

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{\alpha}_1 abcde\boldsymbol{\gamma}_1 \\ \boldsymbol{x}' &= \boldsymbol{u}\boldsymbol{s}'\boldsymbol{v} \in D^*(\boldsymbol{x}) \\ \boldsymbol{x}'' &= \boldsymbol{u}\boldsymbol{s}''\boldsymbol{v} \in D^{0,1}(\boldsymbol{x}'), \end{aligned} \tag{3}$$

where $\boldsymbol{s}'$ belongs to $D^*(abcde)$ and starts with ab and ends with de; $\boldsymbol{s}''$ is obtained from $\boldsymbol{s}'$ by substituting an occurrence of c; $\boldsymbol{u}ab \in D^*(\boldsymbol{\alpha}_1 ab)$ and $de\boldsymbol{v} \in D^*(de\boldsymbol{\gamma}_1)$.

From (3), $R(\boldsymbol{x}'') = R(\boldsymbol{\alpha}_1 R(\boldsymbol{s}'')\boldsymbol{\gamma}_1)$, where $R(\boldsymbol{s}'')$ starts with ab and ends with de (which may fully or partially overlap). The outer R in $R(\boldsymbol{\alpha}_1 R(\boldsymbol{s}'')\boldsymbol{\gamma}_1)$ may remove some symbols at the end of $\boldsymbol{\alpha}_1$, beginning or end of $R(\boldsymbol{s}'')$, and the beginning of $\boldsymbol{\gamma}_1$, leading to $\boldsymbol{\alpha\beta}'\boldsymbol{\gamma}$, where $\boldsymbol{\alpha}$ is a prefix of $\boldsymbol{\alpha}_1$, $\boldsymbol{\beta}'$ is a substring of $R(\boldsymbol{s}'')$, and $\boldsymbol{\gamma}$ is a suffix of $\boldsymbol{\gamma}_1$. Hence, $|\boldsymbol{\beta}'| \le |R(\boldsymbol{s}'')|$.

It can be shown that $\boldsymbol{s}'$ is the concatenation with overlap[1] of a descendant of abc, a descendant of bcd, and a descendant of cde. In $\boldsymbol{s}''$ each of these descendants are altered in at most one position. And so $|\boldsymbol{\beta}'| \le |R(\boldsymbol{s}'')| \le 3L = 39$. ■

IV. Error-correcting codes

Having studied how duplication roots are affected by tandem duplication and substitution errors, in this section we construct codes that can correct such errors. We will also determine the rate of these codes and compare them with the rate of the codes that only correct duplications, which provides an upper bound.

A. Code constructions

As noted in the previous section, the effect of a substitution error on the root of the stored codeword is local in the sense that a substring of bounded length may be deleted and another substring of bounded length may be inserted in its position. A natural approach to correcting such errors is to divide the codewords into blocks such that this alteration can affect a limited number of blocks. In particular, we divide the string into *message blocks* that are separated by *marker blocks* known to the decoder. We start with an auxiliary construction.

Construction 3. *Let l, m, N be positive integers with $m > l$ and $\boldsymbol{\sigma} \in \mathrm{Irr}(l)$. The code $\mathcal{C}_{\boldsymbol{\sigma}}$ (where dependence on N, m is implicit) consists of strings $\boldsymbol{x}$ obtained by alternating between* (message) blocks *of length m and copies of the* marker *sequence $\boldsymbol{\sigma}$, i.e., $\boldsymbol{x} = B_1\boldsymbol{\sigma}B_2\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_N$, such that $\boldsymbol{x} \in \mathrm{Irr}(N(m+l)-l)$ and $|B_i| = m, i \in [N]$. Furthermore, there are exactly two occurrences of $\boldsymbol{\sigma}$ in $\boldsymbol{\sigma}B_i\boldsymbol{\sigma}$, for all $i \in [N]$. Thus, there are precisely $N-1$ occurrences of $\boldsymbol{\sigma}$ in $\boldsymbol{x}$.*

We remark that for our purposes, it is sufficient to relax the condition for B_1 to requiring exactly one occurrence of $\boldsymbol{\sigma}$ in $B_1\boldsymbol{\sigma}$ and similarly, the condition for B_N can be relaxed. For simplicity however, we do not use these relaxed conditions.

With this construction in hand, we show that the effect of a substitution error and many tandem duplications is limited to a small number of blocks.

Theorem 4. *Let $\mathcal{C}_{\boldsymbol{\sigma}}$ be the code defined in Construction 3. If $m > \mathcal{L}$, then there exists a decoder $\mathcal{D}_{\boldsymbol{\sigma}}$ that, for any $\boldsymbol{x} \in \mathcal{C}_{\boldsymbol{\sigma}}$ and $\boldsymbol{y} \in R(D^{*,\le 1}(\boldsymbol{x}))$, outputs $\boldsymbol{z} = \mathcal{D}_{\boldsymbol{\sigma}}(\boldsymbol{y})$ such that, relative to $\boldsymbol{x}$, either two of the blocks B_i are substituted in $\boldsymbol{z}$ or four of them are erased.*

[1]By concatenation with overlap, we mean identical symbols at the extremes can appear once. For example, concatenation with overlap of abc and bcd results in either $abcbcd$ or $abcd$.

Proof: Let $\boldsymbol{x} = \boldsymbol{\alpha\beta\gamma}$ and $\boldsymbol{y} = \boldsymbol{\alpha\beta'\gamma}$, where by Theorem 2, $|\boldsymbol{\beta}|, |\boldsymbol{\beta}'| \leq \mathcal{L}$. The decoder considers two cases depending whether the marker sequences $\boldsymbol{\sigma}$ are in the same positions in $\boldsymbol{y}$ as in the codewords in $\mathcal{C}_{\boldsymbol{\sigma}}$. If this is the case, then $|\boldsymbol{\beta}| = |\boldsymbol{\beta}'| \leq \mathcal{L}$. Since $\mathcal{L} < m = |B_i|$, at most two (adjacent) blocks B_i are affected by substituting $\boldsymbol{\beta}$ by $\boldsymbol{\beta}'$ and thus $\boldsymbol{z} = \boldsymbol{y}$ differs from $\boldsymbol{x}$ in at most two blocks.

On the other hand, if the markers are in different positions in $\boldsymbol{y}$ compared to the codewords in $\mathcal{C}_{\boldsymbol{\sigma}}$, the decoder uses the location of the markers to identify the position of the blocks that may be affected and erases them, as described below. To avoid a separate treatment for blocks B_1 and B_N, the decoder appends $\boldsymbol{\sigma}$ to the beginning and end of $\boldsymbol{y}$ and assumes that the codewords are of the form $\boldsymbol{\sigma}B_1\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_N\boldsymbol{\sigma}$. Define a block in $\boldsymbol{y}$ as a maximal substring that does not overlap with any $\boldsymbol{\sigma}$. By the assumption of this case, there is at least one block B in $\boldsymbol{y}$ whose length differs from m. Hence, $\boldsymbol{y}$ has a substring $\boldsymbol{u}$ of length $m + 2l$ that starts with $\boldsymbol{\sigma}$ and contains part or all of B but does not end with $\boldsymbol{\sigma}$.

Since $\boldsymbol{u}$ is not a substring of any codeword, it must overlap with $\boldsymbol{\beta}'$ (if $\boldsymbol{\beta}'$ is the empty string, then $\boldsymbol{u}$ surrounds the location from which $\boldsymbol{\beta}$ was deleted and we may still consider that $\boldsymbol{u}$ and $\boldsymbol{\beta}'$ overlap). Let $\delta = |\boldsymbol{x}| - |\boldsymbol{y}| = |\boldsymbol{\beta}| - |\boldsymbol{\beta}'|$ and $\delta^+ = \max(0, \delta)$. Since $|\boldsymbol{\beta}'| \leq \min(\mathcal{L}, \mathcal{L} - \delta) = \mathcal{L} - \delta^+$, removing $\boldsymbol{u}$ along with the $\mathcal{L} - \delta^+ - 1$ elements on each of its sides, with a total length of $m + 2l + 2\mathcal{L} - 2\delta^+ - 2$, will remove $\boldsymbol{\beta}'$ from $\boldsymbol{y}$. This results in a sequence that relative to $\boldsymbol{x}$ suffers a deletion of length at most $m + 2l + 2\mathcal{L} - 2\delta^+ - 2 + |\boldsymbol{\beta}| - |\boldsymbol{\beta}'| < 3m + 2l$ from a known position. The deletion affects at most 4 blocks and since its location is known, the decoder can mark these blocks as erased. ■

In Construction 3, the constraint that $\boldsymbol{x}$ must be irreducible creates inter-dependencies between the message blocks, making the code more complex. The following theorem allows us to treat each message block independently provided that $\boldsymbol{\sigma}$ is sufficiently long.

Theorem 5. *Let $\boldsymbol{x}$ be as defined in Construction 3 and assume $l \geq 5$. The condition $\boldsymbol{x} \in \mathrm{Irr}(N(m + l) - l)$ is satisfied if*

$$\boldsymbol{\sigma}B_i\boldsymbol{\sigma} \in \mathrm{Irr}(m + 2l), \quad \text{for all } i \in [N]. \tag{4}$$

Proof: Suppose that $\boldsymbol{x}$ has a repeat $\boldsymbol{aa}$, with $|\boldsymbol{a}| \leq 3$. Since $|\boldsymbol{aa}| \leq 6$ and $|\boldsymbol{\sigma}| \geq 5$, there is no i such that the repeat lies in $B_i\boldsymbol{\sigma}B_{i+1}$ and overlaps both B_i and B_{i+1}. So it must be fully contained in $B_1\boldsymbol{\sigma}$, $\boldsymbol{\sigma}B_N$, or $\boldsymbol{\sigma}B_i\boldsymbol{\sigma}$ for some $2 \leq i \leq N - 1$, contradicting the assumption (4). ■

We now present a code based on Construction 3 and prove that it can correct many tandem duplications and one substitution error.

Construction 6. *Let l, m be positive integers with $m > l \geq 5$, and $\boldsymbol{\sigma} \in \mathrm{Irr}(l)$. Furthermore, let $\mathcal{B}_{\boldsymbol{\sigma}}^m$ denote the set of sequences B such that $\boldsymbol{\sigma}B\boldsymbol{\sigma} \in \mathrm{Irr}(m + 2l)$ has exactly two occurrences of $\boldsymbol{\sigma}$, and $M = M_{\boldsymbol{\sigma}}^{(m)} = |\mathcal{B}_{\boldsymbol{\sigma}}^m|$. Finally, let t be a positive integer such that $2^t \leq M$ and $\zeta : \mathbb{F}_{2^t} \to \mathcal{B}_{\boldsymbol{\sigma}}^m$ be a one-to-one mapping. We define $\mathcal{C}_{MDS}$ as*

$$\mathcal{C}_{MDS} = \{\zeta(c_1)\boldsymbol{\sigma}\zeta(c_2)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta(c_N) : \boldsymbol{c} \in \mathrm{MDS}(N, N-4, 5)\},$$

where $\mathrm{MDS}(N, N - 4, 5)$ denotes an MDS code over $\mathbb{F}_{2^t}$ of length $N = 2^t - 1$, dimension $N - 4$, and Hamming distance $d_H = 5$.

Theorem 7. *If $m > \mathcal{L}$, the error-correcting code $\mathcal{C}_{\mathrm{MDS}}$ in Construction 6 can correct any number of ≤ 3-TD and at most one substitution errors.*

B. Construction of message blocks

In this subsection, we study the set $\mathcal{B}_{\boldsymbol{\sigma}}^m$ of valid message blocks of length m with $\boldsymbol{\sigma}$ as the marker. First, in Construction 6, the markers $\boldsymbol{\sigma}$ do not contribute to the size of the code and so to maximize the code rate, we set $l = |\boldsymbol{\sigma}| = 5$. Then $\boldsymbol{\sigma} \in \mathrm{Irr}(5)$.

For a given $\boldsymbol{\sigma}$, we need to find the set $\mathcal{B}_{\boldsymbol{\sigma}}^m$. The first step in this direction is finding all irreducible sequences of length $m + 2l = m + 10$. We will then identify those that start and end with $\boldsymbol{\sigma}$ but contain no other $\boldsymbol{\sigma}$s.

As shown in [2], the set of ≤ 3-irreducible strings over an alphabet of size q is a regular language whose graph $G_q = (V_q, \xi_q)$ is a subset of the De Bruijn graph. The vertex set V_q consists of 5-tuples $a_1a_2a_3a_4a_5$ that do not have any repeats (of length at most 2). There is an edge from $a_1a_2a_3a_4a_5 \to a_2a_3a_4a_5a_6$ if $a_1a_2a_3a_4a_5a_6$ belongs to $\mathrm{Irr}(6)$. The label for this edge is a_6. The label for a path is the 5-tuple representing its starting point concatenated with the labels of the subsequent edges. In this way, the labels of paths in this graph are the irreducible sequences. The graph G_q, when $q = 3$, can be found in [2, Fig. 1].

The following theorem characterizes the set $\mathcal{B}_{\boldsymbol{\sigma}}^m$ and will be used in the next section to find the size of the code.

Theorem 8. *Over an alphabet of size q and for $\boldsymbol{\sigma} \in \mathrm{Irr}(5)$, there is a one-to-one correspondence between $B \in \mathcal{B}_{\boldsymbol{\sigma}}^m$ and paths of length $m + 5$ in G_q that start and end in $\boldsymbol{\sigma}$ but do not visit $\boldsymbol{\sigma}$ otherwise. Specifically, each sequence $B \in \mathcal{B}_{\boldsymbol{\sigma}}^m$ corresponds to a path with the label $\boldsymbol{\sigma}B\boldsymbol{\sigma}$.*

C. Code rate

We now turn to find the rate of the code introduced in this section. For a code C of length n and size $|C|$, the rate is defined as $R(C) = \frac{1}{n}\log|C|$. For the code of Construction 6,

$$R(\mathcal{C}_{\mathrm{MDS}}) = \frac{N - 4}{Nm + (N - 1)l}\log(N + 1), \tag{5}$$

where N depends on the choice of $\boldsymbol{\sigma} \in \mathrm{Irr}(5)$. More specifically, $N \leq 2^{\lfloor \log M_{\boldsymbol{\sigma}}^{(m)} \rfloor} - 1$. Choosing the largest permissible value for N implies that $N \geq (M_{\boldsymbol{\sigma}}^{(m)} - 1)/2$ and

$$\begin{aligned} R(\mathcal{C}_{\mathrm{MDS}}) &\geq \frac{1 - 4/N}{m + l}\log(N + 1) \\ &\geq \frac{1}{m + l}\left(1 - \frac{8}{M_{\boldsymbol{\sigma}}^{(m)} - 1}\right)(\log M_{\boldsymbol{\sigma}}^{(m)} - 1). \end{aligned} \tag{6}$$

If we let m and $M_{\boldsymbol{\sigma}}^{(m)}$ grow large, the rate becomes

$$R(\mathcal{C}_{\mathrm{MDS}}) = \frac{1}{m}\log M_{\boldsymbol{\sigma}}^{(m)}(1+o(1)).$$

For a given alphabet Σ_q, let A denote the adjacency matrix of G_q, where the rows and columns of A are indexed by $\boldsymbol{v} \in V_q \subseteq \Sigma_q^5$. Furthermore, let $A_{(\boldsymbol{v})}$ be obtained by deleting the row and column corresponding to $\boldsymbol{v}$ from A and $c_{(\boldsymbol{v})}$ (resp. $r_{(\boldsymbol{v})}^T$) be the column (row) of A corresponding to $\boldsymbol{v}$ with the element corresponding to $\boldsymbol{v}$ removed. Recall that $M_{\boldsymbol{\sigma}}^{(m)} = |\mathcal{B}_{\boldsymbol{\sigma}}^m|$. From Theorem 8, we have

$$M_{\boldsymbol{\sigma}}^{(m)} = r_{(\boldsymbol{\sigma})}^T \left(A_{(\boldsymbol{\sigma})}\right)^{m+l-2} c_{(\boldsymbol{\sigma})} \quad (7)$$

where $(\cdot)^T$ denotes matrix transpose. As $m \to \infty$, if $A_{(\boldsymbol{\sigma})}$ is primitive [16], we have

$$\frac{1}{m+l}\log M_{\boldsymbol{\sigma}}^{(m)} \to \log(\lambda_{\boldsymbol{\sigma}}), \quad (8)$$

where $\lambda_{\boldsymbol{\sigma}}$ is the largest eigenvalue of $A_{(\boldsymbol{\sigma})}$. Maximizing over $\boldsymbol{\sigma} \in V_q$ yields the largest value for $M_{\boldsymbol{\sigma}}^{(m)}$ in (7) and (8), and thus the highest code rate. This is possible to do computationally for small values of q and, in particular, for $q = 4$, which corresponds to data storage in DNA. In this case, $A_{(\boldsymbol{\sigma})}$ is primitive for all choices of $\boldsymbol{\sigma} \in \mathrm{Irr}(5)$ and the largest eigenvalue is obtained for $\boldsymbol{\sigma} = 01201$ (and strings obtained from 01201 by relabeling the alphabet symbols). For this $\boldsymbol{\sigma}$, we find $\lambda_{\boldsymbol{\sigma}} = 2.6534$, leading to an asymptotic code rate of 1.4078 bits/symbol.

It was shown in [2] that the set of irreducible strings of length n is a code correcting any number of $\leq$ 3-TDs. In [11], it was shown that the rate of this code, $\frac{1}{n}\log|\mathrm{Irr}(n)|$, is asymptotically optimal. It is easy to see that $\frac{1}{n}\log|\mathrm{Irr}(n)| \leq \log(q-1)$ as no symbol can be repeated. For the case of $q = 4$, we have $\frac{1}{n}\log|\mathrm{Irr}(n)| = \log 2.6590 = 1.4109$ bits/symbol. Therefore, the cost of protection against a single substitution in our construction is only 0.003 bits/symbol. It should be noted, however, that here we have assumed m is large, thus ignoring the overhead from the MDS code and marker strings.

In addition to the computational rate obtained above for the important case of $q = 4$, we will provide analytical bounds on the code rate. The next lemma will be useful in identifying an appropriate choice of $\boldsymbol{\sigma}$, and the following theorem provides a lower bound for $M_{\boldsymbol{\sigma}}^{(m)}$ for such a choice.

Lemma 9. *For $q > 2$, a vertex $\boldsymbol{v} = a_1a_2a_3a_4a_5$ in G_q has $q-2$ outgoing edges if $a_3 = a_5$ or $a_1a_2 = a_4a_5$. Otherwise, it has $q-1$ outgoing edges.*

Proof: Consider $\boldsymbol{v} = a_1a_2a_3a_4a_5 \in \mathrm{Irr}(5)$, and $\boldsymbol{w} = a_2a_3a_4a_5a_6 \in \mathrm{Irr}(5)$. There is an edge from $\boldsymbol{v}$ to $\boldsymbol{w}$ if $a_1a_2a_3a_4a_5a_6 \in \mathrm{Irr}(6)$. The number of outgoing edges from $\boldsymbol{v}$ equals the number of possible values for a_6 such that this condition is satisfied. Clearly, $a_6 \neq a_5$. Furthermore, if $a_3 = a_5$, then $a_6 \neq a_4$ and if $a_1a_2 = a_4a_5$, then $a_6 \neq a_3$.

However, $a_3 = a_5$ and $a_1a_2 = a_4a_5$ cannot simultaneously hold, since that would imply $a_2 = a_3$, contradicting $\boldsymbol{v} \in \mathrm{Irr}(5)$. Hence, if either $a_3 = a_5$ or $a_1a_2 = a_4a_5$ holds, then there are $q-2$ outgoing edges and if neither holds, there are $q-1$ outgoing edges. ■

Theorem 10. *Over an alphabet of size $q > 2$, there exists $\boldsymbol{\sigma} \in \mathrm{Irr}(5)$ such that $M_{\boldsymbol{\sigma}}^{(m)} \geq (q-2)^{m-c_q}$, where c_q is a constant independent from m.*

Proof: Recall that $M_{\boldsymbol{\sigma}}^{(m)}$ is the number of paths of length $m+5$ in G_q that start and end in $\boldsymbol{\sigma}$ but do not visit $\boldsymbol{\sigma}$ otherwise. Since the path must return to $\boldsymbol{\sigma}$, we will show below that for an appropriate choice of $\boldsymbol{\sigma}$, there is a path in G_q from any vertex to $\boldsymbol{\sigma}$ and define c_q such that the length of this path is at most $c_q + 5$. Hence $M_{\boldsymbol{\sigma}}^{(m)}$ is at least the number of paths of length $m - c_q$ from $\boldsymbol{\sigma}$ to another vertex that do not pass through $\boldsymbol{\sigma}$.

As shown in Lemma 9, each vertex in G_q has at least $q-2$ outgoing edges. We select $\boldsymbol{\sigma}$ such that this still holds even if edges leading to $\boldsymbol{\sigma}$ are excluded. We do so by ensuring that each vertex $\boldsymbol{v}$ with an outgoing edge to $\boldsymbol{\sigma}$ has $q-1$ outgoing edges. Let $\boldsymbol{v} = a_1a_2a_3a_4a_5$ and $\boldsymbol{\sigma} = a_2a_3a_4a_5a_6$. Based on Lemma 9, if $a_2 \neq a_5$ and $a_3 \neq a_5$, then $\boldsymbol{v}$ has $q-1$ outgoing edges. In particular, we can choose $\boldsymbol{\sigma} = 01020$ since $q \geq 3$. With this choice, $M_{\boldsymbol{\sigma}}^{(m)} \geq (q-2)^{m-c_q}$.

To complete the proof, we need to show that there is a path in G_q from any vertex to $\boldsymbol{\sigma} = 01020$. For $q = 3, 4, 5$, we have checked this claim computationally by explicitly forming G_q. Let us then suppose $q \geq 6$, where the alphabet Σ_q contains $\{3, 4, 5\}$. Let $\boldsymbol{v} = a_1 \cdots a_5$ be some vertex in G_q. There is an edge from $\boldsymbol{v}$ to $a_2 \cdots a_6$ for some $a_6 \in \{3, 4, 5\}$ since, from Lemma 9, at most two elements of Σ_q are not permissible. Continuing in similar fashion, in 5 steps, we can go from $\boldsymbol{v}$ to some vertex $\boldsymbol{w} = b_1 \cdots b_5$ whose elements b_i belong to $\{3, 4, 5\}$. We can then reach $\boldsymbol{\sigma}$ in 5 additional steps via the path $\boldsymbol{w} \to b_2 \cdots b_4b_50 \to b_3b_4b_501 \to \cdots \to \boldsymbol{\sigma}$, proving the claim. In particular, for $q \geq 6$, we have $c_q \leq 5$. ■

We can now find a lower bound on the asymptotic rate:

Corollary 11. *For $q > 2$, as $m \to \infty$, $R(\mathcal{C}_{\mathrm{MDS}}) \geq \log(q-2)(1+o(1))$.*

We note that this gives the lower bound of 1 bit/symbol for $q = 4$, which we can compare to the upper bound of $\log(q-1) = 1.585$ for codes correcting only duplications and to the rate obtained computationally following (8), which was 1.4078 bits/symbol.

References

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

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

[3] M. Kovačević and V. Y. Tan, "Asymptotically optimal codes correcting fixed-length duplication errors in DNA storage systems," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2194–2197, 2018.

[4] Y. Yehezkeally and M. Schwartz, "Reconstruction codes for DNA sequences with uniform tandem-duplication errors," *IEEE Transactions on Information Theory*, vol. 66, no. 5, pp. 2658–2668, 2020.

[5] Y. Tang, Y. Yehezkeally, M. Schwartz, and F. Farnoud, "Single-error detection and correction for duplication and substitution channels," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019.

[6] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Transactions on Information Theory*, 2019.

[7] K. Cai, Y. M. Chee, R. Gabrys, H. M. Kiah, and T. T. Nguyen, "Optimal codes correcting a single indel/edit for DNA-based data storage," *arXiv preprint arXiv:1910.06501*, 2019.

[8] O. Elishco, R. Gabrys, and E. Yaakobi, "Bounds and constructions of codes over symbol-pair read channels," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1385–1395, 2020.

[9] H. Lou, M. Schwartz, J. Bruck, and F. Farnoud, "Evolution of k-mer frequencies and entropy in duplication and substitution mutation systems," *IEEE Transactions on Information Theory*, vol. 66, no. 5, pp. 3171–3186, 2020.

[10] S. Jain, F. Farnoud, and J. Bruck, "Capacity and expressiveness of genomic tandem duplication," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6129–6138, 2017.

[11] M. Kovačević, "Codes correcting all patterns of tandem-duplication errors of maximum length 3," *arXiv preprint arXiv:1911.06561*, 2019.

[12] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Deciding the confusability of words under tandem repeats in linear time," *ACM Transactions on Algorithms (TALG)*, vol. 15, no. 3, pp. 1–22, 2019.

[13] D. Pumpernik, B. Oblak, and B. Borštnik, "Replication slippage versus point mutation rates in short tandem repeats of the human genome," *Molecular Genetics and Genomics*, vol. 279, no. 1, pp. 53–61, 2008.

[14] F. Farnoud, M. Schwartz, and J. Bruck, "Estimation of Duplication History under a Stochastic Model for Tandem Repeats," *BMC Bioinformatics*, vol. 20, no. 1, 2019. [Online]. Available: https://doi.org/10.1186/s12859-019-2603-1

[15] Y. Tang and F. Farnoud, "Error-correcting codes for noisy duplication channels," in *57th Annual Allerton Conference on Communication, Control, and Computing*, 2019, pp. 140–146.

[16] D. Lind, B. Marcus, L. Douglas, and M. Brian, *An introduction to symbolic dynamics and coding*. Cambridge university press, 1995.