Error-correcting Codes for Short Tandem Duplication and Edit 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 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 edit errors, where an edit error may be a substitution, deletion, or insertion. We focus on tandem repeats of length at most 3 and design codes for correcting an arbitrary number of duplication errors and one edit error. Because an edited symbol can be duplicated many times (as part of substrings of various lengths), a single edit 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. Compared to optimal codes correcting only duplication errors, numerical results show that the asymptotic cost of protecting against an additional edit 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 [2] have made DNA a promising candidate for rising data storage needs. Compared to traditional storage media, DNA storage has several advantages, including higher data density, longevity, and ease of generating copies [2]. 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 [2]–[16]. The current paper focuses on correcting short duplication and edit errors, where an edit is a substitution, insertion, or deletion.

A (tandem) duplication error generates a copy of a substring of the DNA sequence and inserts it after the original substring [3]. For example, from ACGT we may obtain ACGCGT. 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 [3]–[6] and bounded-length duplication, where the duplication length is bounded from above [3], [17]–[19] have been studied. For duplications whose length is at most

This work was supported in part by NSF grants under grant nos. 1816409 and 1755773. This paper was presented in part at the 2020 IEEE Symposium of Information Theory (ISIT) in 2020 [1].

Yuanyuan Tang is with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, 22903, USA, (email: yt5tz@virginia.edu).

Farzad Farnoud (Hassanzadeh) is with the Department of Electrical and Computer Engineering and the Department of Computer Science, University of Virginia, Charlottesville, VA, 22903, USA, (email: farzad@virginia.edu).

3, the case most relevant to this paper, Jain et al. [3] proposed error-correcting codes that were shown to have an asymptotically optimal rate by Kovačević [18].

In an edit event, a symbol in the sequence is substituted or deleted, or a new symbol is inserted. Among these, substitution errors have been studied in the literature in conjunction with *fixed-length* duplication errors, including substitution errors that are restricted to the inserted copies, reflecting the noisiness of the copying mechanism during the duplication process [20]–[22], and substitution errors that may occur anywhere in the string [6].

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 edit error, which may occur in any position in the string. Considering a single edit error reveals important insights into the interactions between edit and duplication errors and will be of use for studying the general case of t edit errors. As a simple example of this channel, the input ACG may become ACG $\to$ AC$\underline{\text{CC}}$G $\to$ ACTCG $\to$ AC$\underline{\text{TACTAC}}$TCG $\to$ ACT$\underline{\text{CT}}$ACTACTCG , where the duplication copies are marked with underlines, and the occurrences of the symbol T result from copies of the substitution C $\to$ T. Given that an arbitrary number of duplications are possible, an unbounded segment of the output word may be affected by the errors, and, for example, the substituted symbol may appear many times. However, relying on the fact that short tandem duplications lead to regular languages, we show that with an appropriate construction and preprocessing of the output of the channel, the deleterious effects of the errors may be localized. We leverage constrained coding and maximum distance separable codes to design codes for correcting the resulting errors, 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 than the code that corrects short duplications but no edits.

We will first consider only substitution edits and construct error-correcting codes capable of correcting many short duplications and a substitution. We will then prove that the same code can correct any number of duplications and an edit error by transforming insertion and deletion errors to substitution errors.

The paper is organized as follows. In Section II, we provide the notation and relevant background. Section III analyzes the error patterns that result from passing through duplication and substitution channels. After that, the code construction as well as the code size are presented in Section IV. In particular, in Subsection IV-A, we construct codes that can correct any number of duplications and one substitution and in Subsection IV-B, we show that the same codes can correct duplication and edit errors. Finally, Section V presents our concluding remarks.

II. Notation and Preliminaries

Let $\Sigma_q = \{0, 1, \ldots, q-1\}$ denote a finite alphabet of size q. To avoid trivial cases, we assume $q \geq 3$, which in particular includes the case of $q = 4$, relevant to DNA data storage. 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 $\{1, \ldots, n\}$.

Strings over Σ_q are denoted by bold symbols, such as $\boldsymbol{x}$ and $\boldsymbol{y}_j$, or by capital letters. The elements of strings are shown with plain typeface, e.g., $\boldsymbol{x} = x_1x_2\cdots x_n$ 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 their concatenation and $\boldsymbol{x}^m$ denotes the concatenation of m copies of $\boldsymbol{x}$. We use $|\boldsymbol{x}|$ to 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 $\boldsymbol{x}$. Furthermore, $\boldsymbol{u}$ is a prefix of $\boldsymbol{x}$ and $\boldsymbol{w}$ is a suffix of $\boldsymbol{x}$. For a string $\boldsymbol{x}$, two substrings $\boldsymbol{u}$ and $\boldsymbol{v}$ are said to *overlap* if we can write $\boldsymbol{x} = \boldsymbol{abcde}$, where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}$ are nonempty, $\boldsymbol{u} = \boldsymbol{bc}$, and $\boldsymbol{v} = \boldsymbol{cd}$.

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, given $\boldsymbol{x} = 1201210$ we may obtain via $\leq$3-TDs

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

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 $\mathrm{Irr}_{\leq k}(n) \subseteq \Sigma_q^n$ 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 use $\mathrm{Irr}_{\leq k}(*)$ denotes $\leq k$-irreducible strings of arbitrary lengths. Furthermore, let $D^*_{\leq k}(\boldsymbol{x})$ denote the *descendant cone* of $\boldsymbol{x}$, containing all the descendants of $\boldsymbol{x}$ after an arbitrary number of $\leq k$-TDs.

Given a string $\boldsymbol{x}'$, let

$$R_{\leq k}(\boldsymbol{x}') = \{\boldsymbol{x} \in \mathrm{Irr}_{\leq k}(*) | \boldsymbol{x}' \in D^*_{\leq k}(\boldsymbol{x})\}$$

denote the set of *duplication root*s of $\boldsymbol{x}'$, i.e., irreducible sequences of which $\boldsymbol{x}'$ is a descendant. Note that $\boldsymbol{x}$ is the root of itself if $\boldsymbol{x} \in \mathrm{Irr}_{\leq k}(*)$. 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 [3], i.e., $|R_{\leq 3}(\cdot)| = 1$. If $\boldsymbol{x}'$ is a descendant of $\boldsymbol{x}$, we have $R_{\leq 3}(\boldsymbol{x}) = R_{\leq 3}(\boldsymbol{x}')$. For $k = 3$, we may drop the ≤ 3 subscript from the notation and write $D^*(\cdot), R(\cdot), \mathrm{Irr}(\cdot)$.

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 $\boldsymbol{x}''$:

$$\boldsymbol{x}' = 1201202201210 \rightarrow \boldsymbol{x}'' = 1201202\underline{1}01210, \tag{2}$$

where the substituted symbol is underlined.

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*. We note that substitutions are *unrestricted* in the sense that they may occur in any position in the string, unlike the *noisy duplication* setting, where they are restricted to the inserted copies [6], [20]. 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 given in (1) and (2), we have $\boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x})$, denoting that $\boldsymbol{x}''$ is a descendant generated from $\boldsymbol{x}$ after an arbitrary number of $\leq$3-TDs and a substitution error.

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 time 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.

A code C is able to correct an arbitrary number of $\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 this condition, it is sufficient to have

$$R(D^{*,\leq 1}(\boldsymbol{c}_1)) \cap R(D^{*,\leq 1}(\boldsymbol{c}_2)) = \varnothing. \tag{3}$$

Condition (3) implies that for distinct codewords $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, $R(\boldsymbol{c}_1) \neq R(\boldsymbol{c}_2)$. Since both $R(\boldsymbol{c}_1)$ and $R(\boldsymbol{c}_2)$ are singleton, 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 [3], [18]. 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. Specifically, for strings $\boldsymbol{x}$ and $\boldsymbol{x}'' \in D^{*,\leq 1}(\boldsymbol{x})$, it is of interest to determine how $R(\boldsymbol{x}'')$ differs from $R(\boldsymbol{x})$. 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 only $\boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x})$. Note that 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 providing a useful definition and an auxiliary lemma and then a lemma that considers a simple case.

Let $\boldsymbol{s}$ and $\bar{\boldsymbol{s}}$ be strings of length n, and let A be the set of symbols in $\boldsymbol{s}$ and $\bar{A}$ the set of symbols in $\bar{\boldsymbol{s}}$. We say that $\boldsymbol{s}$ *dominates* $\bar{\boldsymbol{s}}$ if there exists a function $f : A \to \bar{A}$ such that $\bar{\boldsymbol{s}} = f(\boldsymbol{s})$, where $f(\boldsymbol{s}) = f(s_1) \cdots f(s_n)$. For example, 0102 dominates 1212 (using the mapping $f(0) = 1, f(1) = 2, f(2) = 2$) but 0102 does not dominate 0010. The string $012 \cdots k$ dominates any string of length $k + 1$.

The following lemma helps reduce the number of cases we need to consider using the dominance relationship defined above.

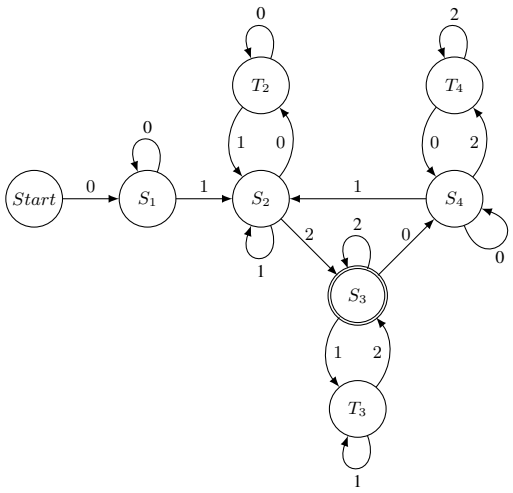


Figure 1: Finite automaton for the regular language $D^*(012)$ based on [17].

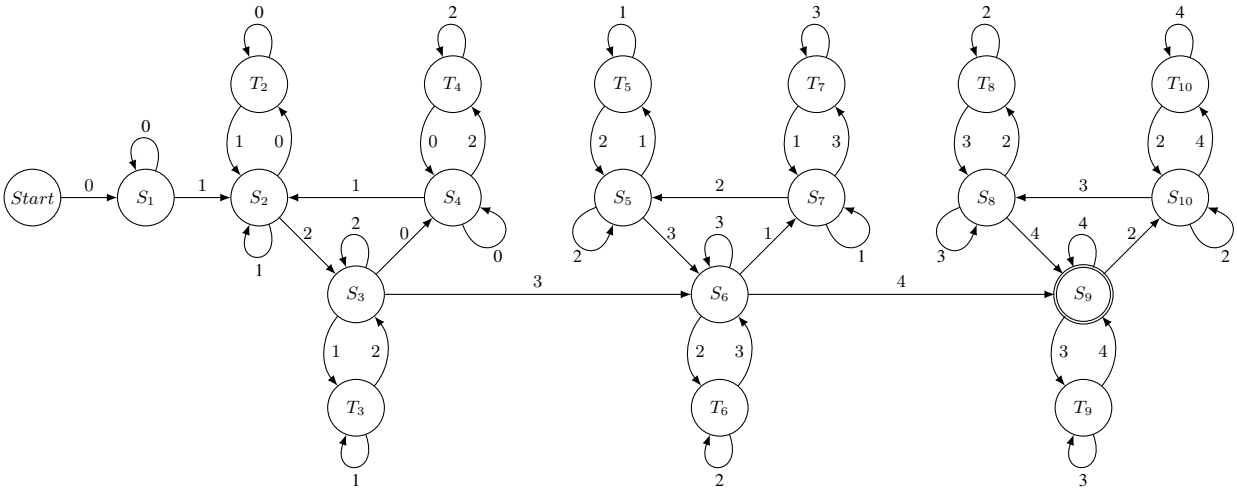


Figure 2: Finite automaton for the regular language $D^*(01234)$ based on [17].

Lemma 1. *Suppose $\boldsymbol{s}$ dominates $\bar{\boldsymbol{s}}$. The following hold:*

i) Suppose we apply the same duplication in both $\boldsymbol{s}$ and $\bar{\boldsymbol{s}}$ (that is, in the same position and with the same length). Let the resulting strings be $\boldsymbol{s}'$ and $\bar{\boldsymbol{s}}'$, respectively. Then $\boldsymbol{s}'$ dominates $\bar{\boldsymbol{s}}'$.

ii) If a deduplication is possible in $\boldsymbol{s}$, a deduplication in the same position and with the same length is possible in $\bar{\boldsymbol{s}}$. Let the result of applying this deduplication to $\boldsymbol{s}$ and $\bar{\boldsymbol{s}}$ be denoted by $\boldsymbol{s}'$ and $\bar{\boldsymbol{s}}'$, respectively. Then $\boldsymbol{s}'$ dominates $\bar{\boldsymbol{s}}'$.

iii) Let $\bar{\boldsymbol{s}}'$ be obtained from $\bar{\boldsymbol{s}}$ via a substitution in position i and let $\boldsymbol{s}'$ be obtained from $\boldsymbol{s}$ by substituting the symbol in position i with a symbol x not present in $\boldsymbol{s}$. Then, $\boldsymbol{s}'$ dominates $\bar{\boldsymbol{s}}'$.

iv) We have $|R(\bar{\boldsymbol{s}})| \leq |R(\boldsymbol{s})|$.

Before proving the lemma, we provide an example for each statement below, where duplicated, deduplicated, and substituted symbols are underlined. For i) consider

$$\boldsymbol{s} = 0102 \rightarrow \boldsymbol{s}' = 010\underline{10}2$$

$$\bar{\boldsymbol{s}} = 0101 \rightarrow \bar{\boldsymbol{s}}' = 010\underline{10}1,$$

for ii) consider

$$\boldsymbol{s} = 0102\underline{10}21 \rightarrow \boldsymbol{s}' = 01021$$

$$\bar{\boldsymbol{s}} = 0101\underline{101}1 \rightarrow \bar{\boldsymbol{s}}' = 01011,$$

for iii) consider

$$\boldsymbol{s} = 01021021 \rightarrow \boldsymbol{s}' = 0102\underline{3}021$$

$$\bar{\boldsymbol{s}} = 01011011 \rightarrow \bar{\boldsymbol{s}}' = 0101\underline{0}011,$$

and for iv) consider

$$\boldsymbol{s} = 0102\underline{10}21 \rightarrow 01021 = R(\boldsymbol{s})$$

$$\bar{\boldsymbol{s}} = 0101\underline{101}1 \rightarrow 01\underline{01}1 \rightarrow 011\underline{} \rightarrow 01 = R(\bar{\boldsymbol{s}}).$$

Proof: Let f be a function that can show $\boldsymbol{s}$ dominates $\bar{\boldsymbol{s}}$. i) For the first statement, the same mapping f also shows that $\boldsymbol{s}'$ dominates $\bar{\boldsymbol{s}}'$. ii) For the second statement, consider a repeat $\boldsymbol{aa}$ in $\boldsymbol{s}$. Then the repeat $f(\boldsymbol{a})f(\boldsymbol{a})$ is present in $\bar{\boldsymbol{s}}$ in the same position. So the deduplication is possible in $\bar{\boldsymbol{s}}$. The same mapping f proves that $\boldsymbol{s}'$ dominates $\bar{\boldsymbol{s}}'$. iii) For the third statement, let the substitution in $\bar{\boldsymbol{s}}$ alter the symbol in position i to some symbol a. If we extend f by mapping x to a, then f proves that $\boldsymbol{s}'$ dominates $\bar{\boldsymbol{s}}'$. iv) From ii), any sequence of deduplications applied to $\boldsymbol{s}$ can also be applied to $\bar{\boldsymbol{s}} = f(\boldsymbol{s})$. In particular, the sequence of deduplications that takes $\boldsymbol{s}$ to its root $R(\boldsymbol{s})$ takes $\bar{\boldsymbol{s}} = f(\boldsymbol{s})$ to $f(R(\boldsymbol{s}))$. The root of $\bar{\boldsymbol{s}}$ can be obtained by removing any remaining repeats in $f(R(\boldsymbol{s}))$ (recall that the root is unique so all sequences of deduplications must lead to the same sequence). Hence $|R(\bar{\boldsymbol{s}})| \leq |f(R(\boldsymbol{s}))|$. Noting $|f(R(\boldsymbol{s}))| = |R(\boldsymbol{s})|$ completes the proof. ■

Lemma 2. *For any alphabet* Σ_q,

$$\max_{\boldsymbol{x}\in\Sigma_q^3} \max_{\boldsymbol{x}''\in D^{*,1}(\boldsymbol{x})} |R(\boldsymbol{x}'')| = 13, \tag{4}$$

$$\max_{\boldsymbol{x}\in\Sigma_q^5} \max_{\boldsymbol{x}''\in D^{*,1}(\boldsymbol{x})} |R(\boldsymbol{x}'')| \leq 17. \tag{5}$$

Proof: For the first statement, it suffices to consider only $\boldsymbol{x} = 012$ and assume that the substitution that leads to $\boldsymbol{x}''$ replaces a symbol in $\boldsymbol{x}$ with some symbol other than 0, 1, and 2, e.g., 3. To see this, consider any string $\bar{\boldsymbol{x}}$ of length 3 over any alphabet. The string $\bar{\boldsymbol{x}}$ is dominated by $\boldsymbol{x}$. Now consider any $\bar{\boldsymbol{x}}'' \in D^{*,1}(\bar{\boldsymbol{x}})$. There is a sequence of "errors" consisting of duplications, a substitution, and more duplications that transforms $\bar{\boldsymbol{x}}$ to $\bar{\boldsymbol{x}}''$. By Lemma 1, i) and iii), there is a corresponding sequence of errors, consisting of duplications, a substitution, and duplications, that when applied to $\boldsymbol{x}$ will result in $\boldsymbol{x}''$, where $\boldsymbol{x}''$ dominates $\bar{\boldsymbol{x}}''$ (the substitution in the sequence of errors for $\boldsymbol{x}$ substitutes the existing symbol with a symbol not in the set $\{0, 1, 2\}$). Then by Lemma 1, iv), we have $|R(\bar{\boldsymbol{x}}'')| \leq |R(\boldsymbol{x}'')|$. Since this is true for any choice of $\bar{\boldsymbol{x}}$ and any $\bar{\boldsymbol{x}}'' \in D^{*,1}(\bar{\boldsymbol{x}})$,

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

State	Irreducible paths from 'Start' to state	Irreducible 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

it suffices to find

$$\max_{\boldsymbol{x}'' \in D^{*,1}(012)} |R(\boldsymbol{x}'')|,$$

where the substitution resulting in $\boldsymbol{x}''$ replaces the existing symbol with a symbol not present in $\boldsymbol{x} = 012$. Henceforth, we assume $\boldsymbol{x} = 012$.

As shown in [17], $D^*(\boldsymbol{x})$ 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 associated with each path is the sequence of the 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 $\hat{w} \notin \{0, 1, 2\}$ 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 $|R(\boldsymbol{x}'')| \leq |R(\boldsymbol{u})| + 1 + |R(\boldsymbol{z})|$ (recall that $R(\boldsymbol{s})$ is a singleton for a string $\boldsymbol{s}$). 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 (henceforth, called an *irreducible path*). 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 irreducible path from Z to S_3; all such possibilities are listed in the third column of Table I. An inspection of Table I 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 $\boldsymbol{x} \to 012\,\underline{012}\,\underline{012} \to 012\,01\underline{012}\,012 \to 012\,0101\underline{212}\,012 \to 0120103212012 = \boldsymbol{x}''$ with a substitution $1 \to 3$ in the last step, where we have combined non-overlapping duplications into a single step.

Let us now prove the second statement. Again we need only consider $\boldsymbol{x} = 01234$, for which $D^*(\boldsymbol{x})$ is the regular language whose automaton is shown in Figure 2. In a similar manner to the proof of the previous part, we can show that the length of the longest irreducible path from 'Start' to any state in the automaton is at most 8 and the length of the longest irreducible path from any state to S_9 is also at most 8. Hence, $|R(\boldsymbol{x}'')| \leq 8 + 1 + 8 = 17$, completing the proof. ■

We now consider changes to the roots of arbitrary strings when passed through a channel with arbitrarily many $\leq$3-TDs and one substitution. The next lemma is used in the main result of this section, Theorem 5, which shows that even though a substituted symbol may be duplicated many times, the effect of a substitution on the root is bounded.

Lemma 3. *Let $\boldsymbol{x}$ be any string of length at least 5 and $\boldsymbol{x}' \in D^*(\boldsymbol{x})$. For any decomposition of $\boldsymbol{x}$ as*

$$\boldsymbol{x} = \ \boldsymbol{r}\ ab\ \boldsymbol{t}\ de\ \boldsymbol{s},$$

for $a, b, d, e \in \Sigma_q$ and $\boldsymbol{r}, \boldsymbol{t}, \boldsymbol{s} \in \Sigma_q^*$, with $\boldsymbol{t}$ nonempty, there is a decomposition of $\boldsymbol{x}'$ as

$$\boldsymbol{x}' = \boldsymbol{u}\ ab\ \boldsymbol{w}\ de\ \boldsymbol{v}$$

such that $\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{v} \in \Sigma_q^*$, $\boldsymbol{u}ab \in D^*(\boldsymbol{r}ab)$, $ab\boldsymbol{w}de \in D^*(ab\boldsymbol{t}de)$, and $de\boldsymbol{v} \in D^*(de\boldsymbol{s})$.

Proof: If $\boldsymbol{x} = \boldsymbol{x}'$, the claim is true since we may choose $\boldsymbol{u} = \boldsymbol{r}, \boldsymbol{w} = \boldsymbol{t}, \boldsymbol{v} = \boldsymbol{s}$. It suffices to consider the case in which $\boldsymbol{x}'$ is obtained from $\boldsymbol{x}$ via a single duplication. The case of more duplications can be proved inductively.

First suppose the length of the duplication transforming $\boldsymbol{x}$ to $\boldsymbol{x}'$ is 1. If this duplication occurs in $\boldsymbol{r}$, we choose $\boldsymbol{u}$ to be the descendant of $\boldsymbol{r}$ and let $\boldsymbol{w} = \boldsymbol{t}$ and $\boldsymbol{v} = \boldsymbol{s}$, satisfying the claim. Duplication of a single symbol in $\boldsymbol{t}$ or $\boldsymbol{s}$ is handled similarly. If a is duplicated, we let $\boldsymbol{u} = \boldsymbol{r}a$, $\boldsymbol{w} = \boldsymbol{t}$, $\boldsymbol{v} = \boldsymbol{s}$. If b is duplicated, we let $\boldsymbol{u} = \boldsymbol{r}$, $\boldsymbol{w} = b\boldsymbol{t}$, $\boldsymbol{v} = \boldsymbol{s}$. The cases for d and e are similar.

Second, consider a duplication of length 2 or 3. Such a duplication is fully contained in $\boldsymbol{r}ab$, $ab\boldsymbol{t}de$, or $de\boldsymbol{s}$. A duplication of length 2 or 3 applied to a string $\boldsymbol{z}$ does not alter the first two and the last two symbols of $\boldsymbol{z}$. So, for example, if the duplication occurs in $\boldsymbol{r}ab$, then we can choose $\boldsymbol{u}$ such that $\boldsymbol{u}ab \in D^1(\boldsymbol{r}ab)$ and let $\boldsymbol{w} = \boldsymbol{t}$ and $\boldsymbol{v} = \boldsymbol{s}$. The cases of duplications contained in the other strings are similar. ■

We now provide an example in which we illustrate how the root of a string can be altered by several duplications and one substitution.

Example 4. *Fix $\Sigma_4 = \{0, 1, 2, 3\}$ as the alphabet. In the following examples, $\boldsymbol{x}$ is an irreducible string, $\boldsymbol{x}' \in D^*(\boldsymbol{x})$, and $\boldsymbol{x}'' \in D^{0,1}(\boldsymbol{x}')$. We compare $R(\boldsymbol{x}) = \boldsymbol{x}$ with $R(\boldsymbol{x}'')$. In particular, we will decompose $R(\boldsymbol{x})$ and $R(\boldsymbol{x}'')$ as $R(\boldsymbol{x}) = \boldsymbol{\alpha\beta\gamma}$ and $R(\boldsymbol{x}'') = \boldsymbol{\alpha\beta'\gamma}$. In other words, $R(\boldsymbol{x}'')$ can be obtained from $R(\boldsymbol{x})$ by deleting $\boldsymbol{\beta}$ and inserting $\boldsymbol{\beta}'$.*

- *Let $\boldsymbol{x} = 012302$, $\boldsymbol{x}' = 011\underline{2}01\underline{2012}30\underline{20}\underline{2}$, and $\boldsymbol{x}'' = 011\underline{2}01\mathbf{3}01\underline{2}30\underline{20}\underline{2}$, where the underlined symbols result from duplication and the bold symbol from substitution. Then $R(\boldsymbol{x}'') = 012013012302$ and the change from $R(\boldsymbol{x})$ to $R(\boldsymbol{x}'')$ can be viewed as*

$$R(\boldsymbol{x}) = \underbrace{012}_{\boldsymbol{\alpha}}\underbrace{302}_{\boldsymbol{\gamma}} \to R(\boldsymbol{x}'') = \underbrace{012}_{\boldsymbol{\alpha}}\underbrace{013012}_{\boldsymbol{\beta'}}\underbrace{302}_{\boldsymbol{\gamma}},$$

  with $\boldsymbol{\beta} = \Lambda$.
- *Let $\boldsymbol{x} = 13203103$, $\boldsymbol{x}' = 13\underline{13}2\underline{13}203\underline{10}3\underline{10}3$, and $\boldsymbol{x}'' = 13\underline{13}2\underline{13}\mathbf{1}03\underline{10}3\underline{10}3$. Then $R(\boldsymbol{x}'') = 13213103$ and the change from $R(\boldsymbol{x})$ to $R(\boldsymbol{x}'')$ can be viewed as*

$$R(\boldsymbol{x}) = \underbrace{132}_{\boldsymbol{\alpha}}\underbrace{0}_{\boldsymbol{\beta}}\underbrace{3103}_{\boldsymbol{\gamma}} \to R(\boldsymbol{x}'') = \underbrace{132}_{\boldsymbol{\alpha}}\underbrace{1}_{\boldsymbol{\beta'}}\underbrace{3103}_{\boldsymbol{\gamma}}.$$

- *Let $\boldsymbol{x} = 012010321201230$, $\boldsymbol{x}' = 012\underline{012}0103212\underline{0}2\underline{012}\underline{012}30$, and $\boldsymbol{x}'' = 012\underline{012}0101\mathbf{1}2120\underline{2}\underline{012}\underline{012}30$. Then $R(\boldsymbol{x}'') =$*

01230 *and the change from* $R(\boldsymbol{x})$ *to* $R(\boldsymbol{x}'')$ *can be viewed as*

$$R(\boldsymbol{x}) = \underbrace{012}_{\boldsymbol{\alpha}}\underbrace{0103212012}_{\boldsymbol{\beta}}\underbrace{30}_{\boldsymbol{\gamma}} \to R(\boldsymbol{x}'') = \underbrace{012}_{\boldsymbol{\alpha}}\underbrace{30}_{\boldsymbol{\gamma}}, \tag{6}$$

with $\boldsymbol{\beta}' = \Lambda$.

Let $\mathcal{L}$ be the smallest integer, if it exists, such that for any alphabet Σ_q, any $\boldsymbol{x} \in \Sigma_q^*$, and any $\boldsymbol{x}'' \in D^{*,1}(\boldsymbol{x})$, we can obtain $R(\boldsymbol{x}'')$ from $R(\boldsymbol{x})$ by deleting a substring of length at most $\mathcal{L}$ and inserting a substring of length at most $\mathcal{L}$ in the same position. The example given in (6) shows that $\mathcal{L}$, if it exists, satisfies $\mathcal{L} \geq 10$. We note however that the definition does not guarantee that $\mathcal{L}$ exists as we may be able to produce examples in which the length of the deleted or the inserted substring is arbitrarily long. The next theorem shows that such examples cannot be constructed by providing an explicit upper bound on $\mathcal{L}$.

Theorem 5. *$\mathcal{L}$ exists (i.e., it is finite). Moreover, $\mathcal{L} \leq 17$.*

Proof: 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 $R(\boldsymbol{x}) = R(\boldsymbol{x}_0)$.

We will find $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\beta}', \boldsymbol{\gamma} \in \Sigma_q^*$ with $R(\boldsymbol{x}) = \boldsymbol{\alpha}\boldsymbol{\beta}\boldsymbol{\gamma}$ and $R(\boldsymbol{x}'') = \boldsymbol{\alpha}\boldsymbol{\beta}'\boldsymbol{\gamma}$ such that $|\boldsymbol{\beta}'| \leq 17$. By symmetry, it suffices to prove $|\boldsymbol{\beta}'| \leq 17$ for all irreducible $\boldsymbol{x}$. To see this symmetry, note that $\boldsymbol{\alpha}\boldsymbol{\beta}'\boldsymbol{\gamma}$ is obtained from $\boldsymbol{\alpha}\boldsymbol{\beta}\boldsymbol{\gamma}$ by applying, in order, duplications, a single substitution, more duplications, and finally removing all repeats (performing all possible deduplications). Recall that for $\leq$3-TDs, the root is unique and regardless of the order in which deduplications are applied, we will arrive at the same root. In other words, applying a sequence of duplications to a string $\boldsymbol{s}$ and then removing all repeats is equivalent to removing all repeats from $\boldsymbol{s}$. Hence, we may instead assume that the process transforming $\boldsymbol{\alpha}\boldsymbol{\beta}\boldsymbol{\gamma}$ to $\boldsymbol{\alpha}\boldsymbol{\beta}'\boldsymbol{\gamma}$ is as follows: duplications, substitution, deduplications. Since this process is reversible, general statements that hold for $\boldsymbol{\beta}'$ also hold for $\boldsymbol{\beta}$.

Let $\boldsymbol{x}' \in D^*(\boldsymbol{x})$ be obtained from $\boldsymbol{x}$ through duplications and $\boldsymbol{x}''$ be obtained from $\boldsymbol{x}'$ through a substitution. We assume that $\boldsymbol{x} = \boldsymbol{r}abcde\boldsymbol{s}$, where $\boldsymbol{r}, \boldsymbol{s} \in \Sigma_q^*$ and $a, b, c, d, e \in \Sigma_q$, such that the substituted symbol in $\boldsymbol{x}'$ is a copy of c. Note that if $|\boldsymbol{x}| < 5$ or if a copy of one of its first two symbols or its last two symbols are substituted, then we can no longer write $\boldsymbol{x}$ as described. To avoid considering these cases separately, we may append two dummy symbols to the beginning of $\boldsymbol{x}$ and two dummy symbols to the end of $\boldsymbol{x}$, where the four dummy symbols are distinct and do not belong to Σ_q, and prove the result for this new string. Since these dummy symbols do not participate in any duplication, substitution, or deduplication events, the proof is also valid for the original $\boldsymbol{x}$.

With the above assumption and based on Lemma 3, we can write

$$\begin{aligned} \boldsymbol{x} &= \ \boldsymbol{r}\ ab\ c\ de\ \boldsymbol{s} \\ \boldsymbol{x}' &= \boldsymbol{u}\ ab\ \boldsymbol{w}\ de\ \boldsymbol{v} \in D^*(\boldsymbol{x}), \\ \boldsymbol{x}'' &= \boldsymbol{u}\ ab\ \boldsymbol{z}\ de\ \boldsymbol{v} \in D^{0,1}(\boldsymbol{x}'), \end{aligned} \tag{7}$$

where $\boldsymbol{u}ab \in D^*(\boldsymbol{r}ab)$, $ab\boldsymbol{w}de \in D^*(abcde)$, $de\boldsymbol{v} \in D^*(de\boldsymbol{s})$, and $\boldsymbol{z}$ is obtained from $\boldsymbol{w}$ by substituting an occurrence of c. From (7), $R(\boldsymbol{x}'') = R(\boldsymbol{r}R(ab\boldsymbol{z}de)\boldsymbol{s})$, where $R(ab\boldsymbol{z}de)$ starts with ab and ends with de (which may fully or partially

overlap). The outer R in $R(\boldsymbol{r}R(ab\boldsymbol{z}de)\boldsymbol{s})$ may remove some symbols at the end of $\boldsymbol{r}$, beginning and end of $R(ab\boldsymbol{z}de)$, and the beginning of $\boldsymbol{s}$, leading to $\boldsymbol{\alpha}\boldsymbol{\beta}'\boldsymbol{\gamma}$, where $\boldsymbol{\alpha}$ is a prefix of $\boldsymbol{r}$, $\boldsymbol{\beta}'$ is a substring of $R(ab\boldsymbol{z}de)$, and $\boldsymbol{\gamma}$ is a suffix of $\boldsymbol{s}$. Hence, $|\boldsymbol{\beta}'| \leq |R(ab\boldsymbol{z}de)|$. But $ab\boldsymbol{z}de \in D^{*,1}(abcde)$ and thus by Lemma 2, $|R(ab\boldsymbol{z}de)| \leq 17$, completing the proof. ■

IV. Error-correcting codes

Having studied how duplication roots are affected by tandem duplication and substitution errors, in Subsection IV-A, we construct codes that can correct such errors. In Subsection IV-B, we show that the same codes can correct duplication and edit errors. We will also determine the rate of these codes and compare it with the rate of codes that only correct duplications, which provides an upper bound.

A. Code constructions for correcting duplications and a substitution error

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 6. *Let l, m, N be positive integers with $m > l$ and $\boldsymbol{\sigma} \in \mathrm{Irr}(l)$. The code $\mathcal{C}_{\boldsymbol{\sigma}}$ of length $n = N(m+l) - l$ over Σ_q consists of irreducible 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 $\boldsymbol{x} \in \mathrm{Irr}(N(m+l)-l)$, $B_i \in \mathrm{Irr}(m) \subseteq \Sigma_q^m, i \in [N]$, and 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, we can relax the condition on $\boldsymbol{\sigma}B_i\boldsymbol{\sigma}$ for $i = 1, N$. Specifically, it suffices to have exactly one occurrence of $\boldsymbol{\sigma}$ in $B_1\boldsymbol{\sigma}$ and one occurrence of $\boldsymbol{\sigma}$ in $\boldsymbol{\sigma}B_N$. For simplicity however, we do not use these relaxed conditions.

Example 7. *Let $m = 6$, $N = 5$, and $\boldsymbol{\sigma} = 01231$ with $l = 5$. Then the code $\mathcal{C}_{\boldsymbol{\sigma}}$ in Construction 6 will contain the codeword*

$$\boldsymbol{x} = 012013012310301210123120231201231303203012312032023,$$

where the message blocks $B_1 = 012013$, $B_2 = 030121$, $B_3 = 202312$, $B_4 = 303203$, $B_5 = 203023$ are marked in red. Furthermore, $\boldsymbol{\sigma}B_i\boldsymbol{\sigma} \in \mathrm{Irr}(16)$ for $i \in [5]$, which, as will be shown in Lemma 9 below, implies that the codeword $\boldsymbol{x} \in \Sigma_4^{50}$ is irreducible.

Given an input $\boldsymbol{x}$ with N message blocks, let $\boldsymbol{y}$ be the root of the output after duplications and at most one substitution. We define a *block* in $\boldsymbol{y}$ as a maximal substring that does not overlap with any separator sequence $\boldsymbol{\sigma}$. Note that a block in $\boldsymbol{y}$ may or may not be an error-free message block from $\boldsymbol{x}$.

With Construction 6 in hand, in the next theorem, we show that the effect of one substitution and many tandem duplications is limited to a small number of message blocks. We note that $\mathcal{L}$ appearing in the theorem below was defined before Theorem 5 and satisfies $\mathcal{L} \leq 17$.

Theorem 8. *Let $\mathcal{C}_{\boldsymbol{\sigma}}$ be the code defined in Construction 6. If $m > \mathcal{L}$, then there exists a decoder $\mathcal{D}_{\boldsymbol{\sigma}}$ that, for any $\boldsymbol{x} \in \mathcal{C}_{\boldsymbol{\sigma}}$ and $\boldsymbol{y} \in R(D^{*,\leq 1}(\boldsymbol{x}))$, outputs $\boldsymbol{z} = \mathcal{D}_{\boldsymbol{\sigma}}(\boldsymbol{y})$ such that, relative to $\boldsymbol{x}$, either two of the message blocks B_i are substituted in $\boldsymbol{z}$ or four of them are erased.*

Proof: Let $\boldsymbol{x} = \boldsymbol{\alpha}\boldsymbol{\beta}\boldsymbol{\gamma}$ and $\boldsymbol{y} = \boldsymbol{\alpha}\boldsymbol{\beta}'\boldsymbol{\gamma}$, where by Theorem 5, $|\boldsymbol{\beta}|, |\boldsymbol{\beta}'| \leq \mathcal{L}$. To avoid a separate treatment for blocks B_1 and B_N, the decoder appends $\boldsymbol{\sigma}$ to the beginning and the end of $\boldsymbol{y}$ and assumes that the codewords are of the form $\boldsymbol{\sigma}B_1\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_N\boldsymbol{\sigma}$. The decoder considers two cases depending on whether the marker sequences $\boldsymbol{\sigma}$ are in the same positions in $\boldsymbol{y}$ as in the codewords in $\mathcal{C}_{\boldsymbol{\sigma}}$. If the markers are in the same positions, as shown in Figure 3, then $|\boldsymbol{x}| = |\boldsymbol{y}|$, and consequently, $|\boldsymbol{\beta}| = |\boldsymbol{\beta}'| \leq \mathcal{L}$. 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 block.

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 message blocks that may be affected and erases them, as described below. By the definition of blocks in $\boldsymbol{y}$, since the markers are in different positions in $\boldsymbol{x}$ and $\boldsymbol{y}$, 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 $\boldsymbol{\sigma}$. Two examples where such a situation may arise are shown in Figure 4. On the left, $|\boldsymbol{\beta}| = |\boldsymbol{\beta}'|$ and a marker is absent from $\boldsymbol{y}$ due to substituting $\boldsymbol{\beta}$ with $\boldsymbol{\beta}$. On the right, $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ have different lengths, and this causes the markers to move.

Let $\delta = |\boldsymbol{x}| - |\boldsymbol{y}| = |\boldsymbol{\beta}| - |\boldsymbol{\beta}'|$ and $\delta^+ = \max(0, \delta)$. Note that $|\boldsymbol{\beta}'| = |\boldsymbol{\beta}| - \delta \leq \mathcal{L} - \delta$. Furthermore, $|\boldsymbol{\beta}'| \leq \mathcal{L}$ and so $|\boldsymbol{\beta}'| \leq \min(\mathcal{L}, \mathcal{L} - \delta) = \mathcal{L} - \delta^+$. Let $\boldsymbol{y}'$ be obtained by removing $\boldsymbol{u}$ along with $\mathcal{L} - \delta^+ - 1$ elements from each of its sides from $\boldsymbol{y}$. This removes $\boldsymbol{\beta}'$ from $\boldsymbol{y}$. More formally, we claim $\boldsymbol{y}'$ can be obtained via a deletion from $\boldsymbol{x}$. First, suppose $|\boldsymbol{\beta}'| = 0$. Then $\boldsymbol{y} = \boldsymbol{\alpha}\boldsymbol{\gamma}$. Note that $\boldsymbol{u}$ is not a substring of $\boldsymbol{x}$ since every substring of $\boldsymbol{x}$ that has length $m + 2l$ and starts with $\boldsymbol{\sigma}$ also ends with $\boldsymbol{\sigma}$. Hence, it must overlap with both $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$. After deleting $\boldsymbol{u}$ from $\boldsymbol{y}$, we obtain a string $\boldsymbol{y}' = \boldsymbol{\alpha}'\boldsymbol{\gamma}'$ such that $\boldsymbol{\alpha}'$ is a prefix of $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}'$ is a suffix of $\boldsymbol{\gamma}$, proving the claim. Next, suppose that $|\boldsymbol{\beta}'| > 0$ and recall that $\boldsymbol{y} = \boldsymbol{\alpha}\boldsymbol{\beta}'\boldsymbol{\gamma}$. Since $\boldsymbol{u}$ is not a substring of $\boldsymbol{x}$, at least one of the following holds: i) $\boldsymbol{u}$ overlaps with both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}'$ or ii) $\boldsymbol{u}$ overlaps with both $\boldsymbol{\beta}'$ and $\boldsymbol{\gamma}$. Case i) is shown in Figure 4. In either case, the substring of $\boldsymbol{y}$ consisting of $\boldsymbol{u}$ and the $\mathcal{L} - \delta^+ - 1$ elements on each side of $\boldsymbol{y}$ contains $\boldsymbol{\beta}'$, proving the claim. Hence, $\boldsymbol{y}'$ is 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 message blocks and since its location is known, the decoder can mark these message blocks as erased. ■

In Construction 6, the constraint that $\boldsymbol{x}$ must be irreducible creates interdependence between the message blocks, making the code more complex. The following lemma allows us to treat each message block independently provided that $\boldsymbol{\sigma}$ is sufficiently long.

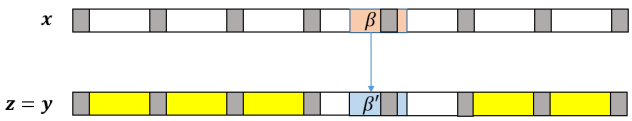


Figure 3: If marker sequences, shown as gray, are in the same positions in the codeword $\boldsymbol{x}$ and the retrieved string $\boldsymbol{y}$, then $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ have the same length and at most two of the message blocks are affected by the errors, as discussed in the proof of Theorem 8.

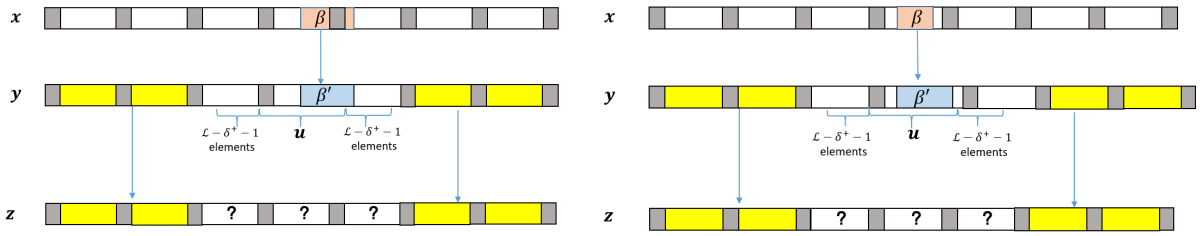


Figure 4: If marker sequences, shown as gray, are in different positions in the codeword $\boldsymbol{x}$ and the retrieved string $\boldsymbol{y}$, then a substring $\boldsymbol{u}$ is identified and then expanded to ensure it contains $\boldsymbol{\beta}'$. Those blocks in $\boldsymbol{y}$ that intersect with this expanded substring are marked as erasures while other blocks are error-free message blocks, as described in the proof of Theorem 8).

Lemma 9. *Let $\boldsymbol{x}$ be as defined in Construction 6 and assume $l \geq 5$. The condition $\boldsymbol{x} \in \mathrm{Irr}(N(m+l)-l)$ is satisfied if*

$$\boldsymbol{\sigma} B_i \boldsymbol{\sigma} \in \mathrm{Irr}(m+2l), \quad \textit{for all } i \in [N]. \tag{8}$$

Proof: Suppose that $\boldsymbol{x}$ has a repeat $\boldsymbol{a}\boldsymbol{a}$, with $|\boldsymbol{a}| \leq 3$. Since $|\boldsymbol{a}\boldsymbol{a}| \leq 6$ 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 assumption (8). ∎

We now present a code based on Construction 6 and prove that it can correct any number of tandem duplications and one substitution error.

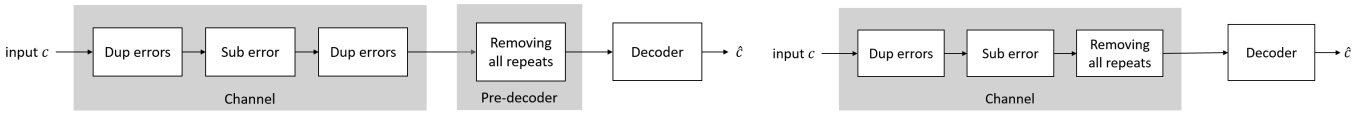
Construction 10. *Let l, m be positive integers with $m > l \geq 5$, and $\boldsymbol{\sigma} \in \mathrm{Irr}(l)$. Furthermore, let $\mathcal{B}^m_{\boldsymbol{\sigma}}$ denote the set of sequences B such that $\boldsymbol{\sigma}B\boldsymbol{\sigma} \in \mathrm{Irr}(m+2l)$ has exactly two occurrences of $\boldsymbol{\sigma}$, and $M = M^{(m)}_{\boldsymbol{\sigma}} = |\mathcal{B}^m_{\boldsymbol{\sigma}}|$. Finally, let t be a positive integer such that $2^t \leq M$ and $\zeta : \mathbb{F}_{2^t} \to \mathcal{B}^m_{\boldsymbol{\sigma}}$ be an injective 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 $d_H = 5$.

Note that the mapping ζ exists because $|\mathbb{F}_{2^t}| \leq |\mathcal{B}^m_{\boldsymbol{\sigma}}|$ by the choice of t. For example, we can sort the elements of $\mathbb{F}_{2^t}$ and $\mathcal{B}^m_{\boldsymbol{\sigma}}$ lexicographically and map the ith element of $\mathbb{F}_{2^t}$ to the ith element of $\mathcal{B}^m_{\boldsymbol{\sigma}}$.

Theorem 11. *If $m > \mathcal{L}$, then the error-correcting code $\mathcal{C}_{\mathrm{MDS}}$ in Construction 10 can correct any number of $\leq$3-TDs and at most one substitution error.*



(a) (b)

Figure 5: The duplication-substitution channel along with the decoder (a) and an equivalent representation of the end-to-end system (b).

Proof: Let the stored codeword be $\boldsymbol{x} = B_1\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_N \in \mathcal{C}_{MDS}$, where $B_i = \zeta(c_i)$ for $i \in [N]$ and $\boldsymbol{c} \in C$, with C denoting an $\mathrm{MDS}(N, N-4, 5)$ code. Based on the definitions of ζ and the set $B^m_{\boldsymbol{\sigma}}$ in Construction 10, the ith message block $B_i = \zeta(c_i)$ satisfies $\boldsymbol{\sigma}B_i\boldsymbol{\sigma} \in \mathrm{Irr}(m+2l)$. Then, by Lemma 9, $\boldsymbol{x} \in \mathrm{Irr}(N(m+l)-l)$ and so $\boldsymbol{x} \in \mathcal{C}_{\boldsymbol{\sigma}}$. Therefore, $\mathcal{C}_{MDS} \subseteq \mathcal{C}_{\boldsymbol{\sigma}}$. Suppose the retrieved word is $\boldsymbol{y}$. By Theorem 8, $\mathcal{D}_{\boldsymbol{\sigma}}(\boldsymbol{y})$ suffers either at most two substitutions or at most four erasures of message blocks. Suppose the message block B_i is substituted by another string $\boldsymbol{v}$ of length m. If $\zeta^{-1}(\boldsymbol{v})$ exists, this translates to a substitution of c_i. If not, we define $\zeta^{-1}(B_i)$ as an arbitrary element of $\mathbb{F}_{2^t}$, again leading to a possible substitution of c_i with another symbol. To decode, we can use the MDS decoder on $\zeta^{-1}(\mathcal{D}_{\boldsymbol{\sigma}}(\boldsymbol{y}))$, which relative to $\boldsymbol{c}$ suffers either ≤ 2 substitutions or ≤ 4 erasures. Given that the minimum Hamming distance of the MDS code is 5, the decoder can successfully recover $\boldsymbol{c}$. ■

B. Extension to edit errors

In this subsection, we extend Theorem 11 to include insertion and deletion errors in addition to substitution errors. We do so by showing an insertion can be viewed as a duplication plus a substitution and a deletion as a substitution plus a deduplication. The duplication-substitution channel discussed so far can be viewed as shown in Figure 5a, where in a pre-decoding step, the root of the retrieved string is found and then passed to the decoder. Recall that for $\leq$3-TDs, applying a sequence of duplications to a string $\boldsymbol{s}$ and then removing all repeats is equivalent to removing all repeats from $\boldsymbol{s}$. Hence, The process shown in Figure 5a is equivalent to the one shown in Figure 5b. The same equivalence holds if we replace the block representing a substitution error with a block representing an edit error. We can now prove the following corollary to Theorem 11.

Corollary 12. *If $m > \mathcal{L}$, then the error-correcting code $\mathcal{C}_{\mathrm{MDS}}$ in Construction 10 can correct any number of $\leq$3-TDs and at most one edit error.*

Proof: Suppose that $\boldsymbol{c} \in \mathcal{C}_{\mathrm{MDS}}$ suffers a sequence of errors consisting of duplications, an edit, and more duplications. Then all repeats are removed from the resulting string. This process is equivalent to applying the first set of duplications and the edit and then removing all the repeats. Denote this sequence of operations applied to $\boldsymbol{c}$ by S and the final result by $S(\boldsymbol{c})$. We show that we can find a sequence S' consisting of duplications, at most one substitution, and removal of all repeats, such that $S(\boldsymbol{c}) = S'(\boldsymbol{c})$. If the edit in S is a substitution, then we let $S' = S$. If it is an insertion or a deletion, we again start by setting $S' = S$ and then modify S' as follows: i) If the edit is an insertion, we replace it in S' by a duplication and, if needed, a substitution. Namely, we duplicate the symbol before the insertion and then substitute the copy as needed. For example, if in S we have $abc \xrightarrow{ins} abxc$, we replace this step by $abc \xrightarrow{dup} abbc \xrightarrow{sub} abxc$, where ins stands for insertion, dup for duplication, and sub for substitution. The substitution is not necessary if $x = b$. ii) If the edit is a deletion, we replace it by a deduplication

that is preceded by a substitution if needed. Namely, we substitute the symbol that is deleted in S to be equal to the previous symbol and then deduplicate it. For example, if in S we have $abc \xrightarrow{del} ac$, we replace this step by $abc \xrightarrow{sub} aac \xrightarrow{dedup} ac$ in S', where del stands for deletion and $dedup$ for deduplication. If $b = a$, the substitution is not needed. Now, S' is a sequence consisting of duplications, at most one substitution, and then removal of all repeats, and furthermore $S(\boldsymbol{c}) = S'(\boldsymbol{c})$. From Theorem 11 and the above discussion about the equivalence of Figures 5a and 5a, we find that the decoder can produce $\boldsymbol{c}$ given $S'(\boldsymbol{c})$. ■

C. Construction of message blocks

In this subsection, we study the set $\mathcal{B}^m_{\boldsymbol{\sigma}}$ of valid message blocks of length m with $\boldsymbol{\sigma}$ as the marker. Since in Construction 10, the markers $\boldsymbol{\sigma}$ do not contribute to the size of the code, to maximize the code rate, we set $l = |\boldsymbol{\sigma}| = 5$, i.e., $\boldsymbol{\sigma} \in \mathrm{Irr}(5)$.

For a given $\boldsymbol{\sigma}$, we need to find the set $\mathcal{B}^m_{\boldsymbol{\sigma}}$. 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 [3], the set of $\leq$3-irreducible strings over an alphabet of size q is a regular language whose graph $G_q = (V_q, \xi_q)$ is a subgraph 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 vertex concatenated with the labels of the subsequent edges. In this way, the label of a path in this graph is an irreducible sequence and each irreducible sequence is the label of a unique path in the graph. The graph G_q, when $q = 3$, can be found in [3, Fig. 1].

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

Theorem 13. *Over an alphabet of size q and for $\boldsymbol{\sigma} \in \mathrm{Irr}(5)$, there is a one-to-one correspondence between $B \in \mathcal{B}^m_{\boldsymbol{\sigma}}$ and paths of length $m + 5$ in G_q that start and end in $\boldsymbol{\sigma}$ but do not visit $\boldsymbol{\sigma}$ in their interiors. Specifically, each sequence $B \in \mathcal{B}^m_{\boldsymbol{\sigma}}$ corresponds to the path with the label $\boldsymbol{\sigma}B\boldsymbol{\sigma}$.*

Proof: Consider a path $p = \boldsymbol{v}_1\boldsymbol{v}_2 \cdots \boldsymbol{v}_{k+1}$ where $\boldsymbol{v}_i$ are vertices of G_q and k is the length of the path. Denote the label of this path by $\boldsymbol{s} = s_1s_2 \cdots s_{k+5}$. It can be shown by induction on k that $\boldsymbol{v}_i = s_is_{i+1}s_{i+2}s_{i+3}s_{i+4}$. Hence, the label of a path of length $m + 5$ that starts and ends in $\boldsymbol{\sigma}$ but does not visit $\boldsymbol{\sigma}$ otherwise is an irreducible sequence with exactly two occurrences of $\boldsymbol{\sigma}$ and is of the form $\boldsymbol{\sigma}B\boldsymbol{\sigma}$ where $B \in \mathcal{B}^m_{\boldsymbol{\sigma}}$. Conversely, suppose $B \in \mathcal{B}^m_{\boldsymbol{\sigma}}$. Then $\boldsymbol{\sigma}B\boldsymbol{\sigma}$ is an irreducible string of length $m + 10$ and thus the label of a unique path of length $m + 5$ in G_q. This path starts and ends in $\boldsymbol{\sigma}$. But it does not visit $\boldsymbol{\sigma}$ in its interior since that would imply there are more than two occurrences of $\boldsymbol{\sigma}$ in $\boldsymbol{\sigma}B\boldsymbol{\sigma}$. ■

D. 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 10,

$$R(\mathcal{C}_{\mathrm{MDS}}) = \frac{N-4}{Nm + (N-1)l}\log(N+1), \tag{9}$$

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

$$\begin{aligned} R(\mathcal{C}_{\mathrm{MDS}}) &\geq \frac{1 - 4/N}{m+l} \log(N+1) \\ &\geq \frac{1}{m+l} \left(1 - \frac{8}{M_{\boldsymbol{\sigma}}^{(m)} - 1}\right) (\log M_{\boldsymbol{\sigma}}^{(m)} - 1). \end{aligned} \tag{10}$$

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

$$R(\mathcal{C}_{\mathrm{MDS}}) = \frac{1}{m} \log M_{\boldsymbol{\sigma}}^{(m)} (1 + o(1)). \tag{11}$$

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 $M_{\boldsymbol{\sigma}}^{(m)} = |\mathcal{B}_{\boldsymbol{\sigma}}^m|$ and $l = 5$. From Theorem 13, $M_{\boldsymbol{\sigma}}^{(m)}$ equals the number of paths of length $m + l$ in G_q that start and end with $\boldsymbol{\sigma}$ but do not visit $\boldsymbol{\sigma}$ in their interiors. The number of paths of length $m + l - 2$ from vertex $\boldsymbol{u}$ to vertex $\boldsymbol{v}$ in G_q is given by the $(\boldsymbol{u}, \boldsymbol{v})$ element of $\left(A_{(\boldsymbol{\sigma})}\right)^{m+l-2}$. Noting that the paths start and end $\boldsymbol{\sigma}$, we have

$$M_{\boldsymbol{\sigma}}^{(m)} = r_{(\boldsymbol{\sigma})}^T \left(A_{(\boldsymbol{\sigma})}\right)^{m+l-2} c_{(\boldsymbol{\sigma})}, \tag{12}$$

where $(\cdot)^T$ denotes matrix transpose. Here, multiplying by $r_{(\boldsymbol{\sigma})}^T$ from the left and $c_{(\boldsymbol{\sigma})}$ from the right allow us to sum over elements $(\boldsymbol{u}, \boldsymbol{v})$ of $\left(A_{(\boldsymbol{\sigma})}\right)^{m+l-2}$ such that there is an edge from $\boldsymbol{\sigma}$ to $\boldsymbol{u}$ and an edge from $\boldsymbol{v}$ to $\boldsymbol{\sigma}$. As $m \to \infty$, if $A_{(\boldsymbol{\sigma})}$ is primitive [23], we have

$$\frac{1}{m+l} \log M_{\boldsymbol{\sigma}}^{(m)} \to \log(\lambda_{\boldsymbol{\sigma}}), \tag{13}$$

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 (12) and (13), 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 [3] that the set of ≤ 3-irreducible strings of length n is a code correcting any number of $\leq$3-TDs. In [18], 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 edit 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. An important quantity affecting the rate of the code is the number of outgoing edges from each vertex in G_q that do not lead to $\boldsymbol{\sigma}$. The asymptotic rate of the code is bounded from below by the number of such edges. The next lemma, which establishes the number of outgoing edges for each vertex, 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 14. *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. ■

Since $\boldsymbol{\sigma}$ must also be excluded, it may seem that the number of outgoing edges may be as low as $q-3$. But we show in the next theorem that with an appropriate choice of $\boldsymbol{\sigma}$, we can have $q-2$ as the lower bound.

Theorem 15. *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 14, 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 14, 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 14, 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, based on (11) and the proceeding theorem:

Corollary 16. *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 (13), which was 1.4078 bits/symbol.

V. Conclusion

This paper considered constructing error-correcting codes for channels with many short duplications and one edit error. Because the channel allows an arbitrary number of duplications, a single edit may affect an unbounded segment of the output. For example, an inserted symbol may appear many times in different positions. However, with an appropriate construction of message blocks and processing of the output strings, the edit error leads to the erasure of at most 4 message blocks or substitution of at most 2. Therefore, a maximum distance separable (MDS) code with minimum Hamming distance 5 over message blocks can correct these errors. However, there is an additional requirement. Namely, the codewords must be irreducible. Separating the message blocks with a marker sequence $\boldsymbol{\sigma}$ of length at least 5 allows us to ensure that the codewords are repeat-free by guaranteeing that each message block is irreducible. The rate of the code is determined by the number of such blocks, which in turn depends on the marker sequence $\boldsymbol{\sigma}$. We showed that permitted message blocks are paths in a modified De Bruijn graph and that choosing $\boldsymbol{\sigma}$ appropriately allows each vertex to have at least $q-2$ outgoing edges, thus guaranteeing an asymptotic rate of at least $\log(q-2)$. When $q=4$, the case corresponding to DNA storage, a computational bound for the code rate shows that the asymptotic rate is only 0.003 bits/symbol smaller than that of the code that corrects short duplications but no edits.

The problem of correcting more edit errors is also of interest but left to future work. Another, possibly more challenging problem is correcting edits and duplications of length bounded by an arbitrary constant k. If k is larger than 3, the duplication root is no longer unique [3], which complicates the code design. Furthermore, a key feature of duplications of length at most 3 is that such duplications lead to regular languages. We used this fact to characterize the effect of the channel on the roots of sequences. However, if $k \geq 4$, then the language is not regular [24], leading to challenges in characterizing the channel. To guarantee a unique root under $\leq k$ duplications with $k \geq 4$, the work [25] constructed error-correcting codes by selecting a set of $(k+1)$-distinct strings with $q \geq (k+1)$ and achieved an asymptotic code rate of $\frac{1}{k}\sum_{i=1}^{k}\log(q-i)$. This means that, despite the interesting approach, to correct duplications of length ≥ 4, [25] requires $q \geq 5$, which is not suitable for the important application of data storage in DNA with $q=4$.

Acknowledgement: The authors thank the anonymous reviewers of the paper for their insightful comments and helpful suggestions.

References

[1] Y. Tang and F. Farnoud, "Error-correcting codes for short tandem duplication and substitution errors," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 734–739.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] Y. Tang, Y. Yehezkeally, M. Schwartz, and F. Farnoud, "Single-error detection and correction for duplication and substitution channels," *IEEE Transactions on Information Theory*, vol. 66, no. 11, pp. 6908–6919, 2020.

[7] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2331–2351, 2020.

[8] 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.

[9] 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.

[10] A. Lenz, Y. Liu, C. Rashtchian, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding for efficient DNA synthesis," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 2885–2890.

[11] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Mass error-correction codes for polymer-based data storage," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 25–30.

[12] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Coding for optimized writing rate in DNA storage," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 711–716.

[13] H. M. Kiah, T. Thanh Nguyen, and E. Yaakobi, "Coding for sequence reconstruction for single edits," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 676–681.

[14] Y. Yehezkeally and M. Schwartz, "Uncertainty of reconstructing multiple messages from uniform-tandem-duplication noise," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 126–131.

[15] T. T. Nguyen, K. Cai, K. A. S. Immink, and H. M. Kiah, "Constrained coding with error control for DNA-based data storage," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 694–699.

[16] J. Sima, N. Raviv, and J. Bruck, "Robust indexing-optimal codes for DNA storage," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 717–722.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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

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

[24] P. Leupold, V. Mitrana, and J. M. Sempere, "Formal languages arising from gene repeated duplication," in *Aspects of Molecular Computing*. Springer, 2003, pp. 297–308.

[25] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Efficient encoding/decoding of gc-balanced codes correcting tandem duplications," *IEEE Transactions on Information Theory*, vol. 66, no. 8, pp. 4892–4903, 2020.