# Error-correcting codes for short tandem duplications and at most $p$ substitutions

Yuanyuan Tang, Hao Lou, and Farzad Farnoud

Electrical & Computer Engineering, University of Virginia, U.S.A., {yt5tz,hl2nu,farzad}@virginia.edu

***Abstract*—Compared to conventional data storage media, DNA has several advantages, including high data density, energy efficiency, longevity, and ease of generating copies. However, challenges arising from the prevalence and variety of errors in the DNA data storage pipeline, which include substitutions, duplications, insertions, and deletions, must be addressed. This paper focuses on simultaneously correcting an arbitrary number of short tandem duplications and at most $p$ substitutions, where a short tandem duplication error consists of inserting a copy of a substring of length at most 3 immediately after it. Interacting with tandem duplications, the substitutions may affect segments of unbounded lengths in the stored sequence. However, if the codewords are irreducible, i.e., they do not have any short tandem repeats, the problem can be cast as correcting edits in at most $p$ substrings of bounded lengths. We construct irreducible codes with a structure that allows identifying where edit errors have occurred, which are then corrected using an MDS code. The rate of the proposed code correcting duplications and at most $p$ substitutions, when $\log p = o(\log n)$, is shown to be at least $\log(q-2)(1-o(1))$, where $q$ is the alphabet size and $n$ is the length of the code.**

## I. Introduction

The exponentially increasing amount of data [1] poses significant challenges for traditional data storage media. Due to its high density, energy-efficiency, longevity, and ease of generating copies [2], [3], DNA serves as an attractive alternative for addressing some of these challenges. For instance, the amount of DNA in a single human cell can ideally hold 12.8 Gb of information, and furthermore DNA may last thousands of years [4]. However, DNA suffers from various types of errors arising from different stages of data storage and retrieval, including insertions, deletions, substitutions, and duplications [2], [5]. Many recent works, including [3], [6]–[19], design error-correcting codes to fight against these errors. This paper focuses on constructing codes for simultaneously correcting short tandem duplication and substitution errors.

A (tandem) duplication in a DNA sequence generates a copy of a substring and inserts it after the original substring [3], where the length of the duplication is the length of the copy. For instance, $\mathsf{CATCAG} \rightarrow \mathsf{CATC\underline{ATC}AG}$ is a tandem duplication of length 3, where the copy is marked with the underline. In recent years, many works have studied both fixed-length duplications [3], [7]–[9] and bounded-length duplications [3], [5], [20]–[23], where the length of fixed-length duplications is a constant and the length of bounded-length duplications is upper bounded. In particular, Jain et al. [3] presented a code for correcting an arbitrary number of duplications of length at most 3, also referred to as *short duplications*, and Kovačević [20] showed that the code is asymptotically optimal.

A substitution in a string changes a symbol to another one over the same alphabet. Previous works have studied error-correcting codes for both *restricted* substitutions occurring in duplication copies [9], [24] and *unrestricted* substitutions [9], [22], [24], which may occur anywhere. In particular, Tang et al. [22] constructed error-correcting codes to correct an arbitrary number of short duplications and at most one unrestricted substitution.

This paper focuses on constructing error-correcting codes that can correct an arbitrary number of short duplications and at most $p$ unrestricted substitutions, extending [22]. For example, given an input $\boldsymbol{x} = \mathsf{TGAC}$, several short duplications and 2 substitutions may produce an output $\boldsymbol{y}$ from $\boldsymbol{x}$ via the following sequence of errors: $\boldsymbol{x} = \mathsf{TGAC} \rightarrow \mathsf{TGA\underline{GA}C} \rightarrow \mathsf{TGA\underline{TGA}GAC} \rightarrow \mathsf{TGAT\mathbf{C}AGAC} \rightarrow \mathsf{TGAT\mathbf{C}\underline{T\mathbf{C}}AGAC} \rightarrow \mathsf{TGAT\mathbf{C}\mathbf{A}\mathbf{C}AGAC} \rightarrow \mathsf{TGAT\mathbf{C}\mathbf{A}\underline{T\mathbf{C}\mathbf{A}}\mathbf{C}AGAC} = \boldsymbol{y}$, where inserted copies are underlined and the substituted symbols and their copies are in bold and different colors. A challenge in correcting duplication and substitution errors simultaneously arises from the fact that a single substitution may be duplicated many times and affect an arbitrarily long segment of the output. However, it was shown in [22] that the effect of a substitution and any number of duplications on the duplication root of the sequence is equivalent to replacing a substring of bounded length, where the duplication root is obtained by replacing all repeats of the form $\boldsymbol{vv}$, $|\boldsymbol{v}| \leq 3$ with $\boldsymbol{v}$. Using this fact, a code is constructed in [22] with repeat-free codewords in which data blocks and "marker" sequences are interleaved. If the marker sequences in the retrieved word are in their original positions, only a small number of block substitutions could have occurred. Otherwise, the markers can be used to uniquely identify the position of the substring edit. In both cases, an MDS code is used to correct the errors.

We show in this paper that multiple substitutions can be viewed as replacing multiple substrings in the duplication roots. However, unlike the case of a single substitution, even if the markers are in their original positions, all data blocks may

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

have been substituted. To address this problem, we impose additional structures on the codewords that will enable us to localize the errors and characterize their effect on the data blocks. We show that the rate of the proposed code correcting any number of short tandem duplications and at most $p$ substitutions, when $\log p = o(\log n)$, is at least $\log(q-2)(1-o(1))$, where $q$ is the alphabet size and $n$ is the length of the code.

The paper is organized as follows. Section II introduces the notation and relevant prior results. Section III analyzes error patterns resulting from an arbitrary number of short duplications and at most $p$ substitutions. Finally, code constructions as well as the code size are provided in Section IV.

## II. Notation and Preliminaries

For integers $a, b$ with $a \leq b$, let $[a, b]$ denote the set of integers $\{a, a+1, \ldots, b\}$. If $a = 1$, $[a, b]$ is simplified as $[b]$. Given an integer $b \geq 1$, let $a \bmod' b$ be the integer in $[b]$ whose remainder when divided by $b$ is the same as that of $a$.

Let $\Sigma_q$ be a finite alphabet of size $q$. Without loss of generality, we let $\Sigma_q = \{0, 1, 2, \ldots, q-1\}$. The set of all strings of length $n$ and all finite strings over $\Sigma_q$ are denoted as $\Sigma_q^n$ and $\Sigma_q^*$, respectively. The empty string is denoted by $\Lambda$ and is a member of $\Sigma_q^*$.

In this paper, strings in $\Sigma_q^*$ are denoted by bold symbols or capital letters, such as $\boldsymbol{x}$, $\boldsymbol{y}_j$, and $B$. The plain typeface is used to denote entries of strings, e.g., $x_i \in \Sigma_q$ denotes the $i$th entry of the string $\boldsymbol{x}$. Let $\boldsymbol{xy}$ denote the concatenation of two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ for $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, and $\boldsymbol{x}^m$ denote the concatenation of $m$ copies of $\boldsymbol{x} \in \Sigma_q^*$. For four strings $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \Sigma_q^*$, $\boldsymbol{v}$ is called a substring of $\boldsymbol{x}$ if $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \boldsymbol{uvw}$. Furthermore, let $|\boldsymbol{x}|$ denote the length of $\boldsymbol{x}$.

A *tandem duplication* (TD) of length $k$ ($k$-TD) generates a copy of a substring with length $k$ and inserts it after the original substring, resulting in a *(tandem) repeat*. For example, given $\boldsymbol{x} = \boldsymbol{uvw}$ with $|\boldsymbol{v}| = k$, a $k$-TD may generate $\boldsymbol{uvvw}$ with the repeat $\boldsymbol{vv}$. A *deduplication* of length $k$ replaces a repeat of the form $\boldsymbol{vv}$ with $\boldsymbol{v}$, where $|\boldsymbol{v}| = k$.

Let $\leq k$-TDs denote TDs of length upper bounded by $k$. This paper focuses on $\leq k$-TDs with $k = 3$, also called *short tandem duplications*. For instance, given a string $\boldsymbol{x} = 3101203 \in \Sigma_4^*$, the output after several $\leq$3-TDs may be

$$\begin{aligned}\boldsymbol{x} =& 3101203 \to 3101\underline{1}203 \to 310112\underline{1}203 \\ \to& 31011\underline{01}\underline{1}21203 = \boldsymbol{x}',\end{aligned} \tag{1}$$

where the inserted copies with length $\leq 3$ are marked with underlines. The output $\boldsymbol{x}'$ is called a *descendant* of $\boldsymbol{x}$, i.e., a sequence derived from $\boldsymbol{x}$ after a number of $\leq$3-TDs. For $\boldsymbol{x} \in \Sigma_q^*$, we let $D^*_{\leq k}(\boldsymbol{x})$ represent the *descendant cone* of $\boldsymbol{x}$, i.e., the set of all descendants of $\boldsymbol{x}$ after an arbitrary number of $\leq k$-TDs.

The set of $\leq k$*-irreducible strings* of length $n$, denoted $\mathrm{Irr}_{\leq k}(n)$, consists of strings without repeats of the form $\boldsymbol{vv}$, where $|\boldsymbol{v}| \leq k$. Furthermore, $\mathrm{Irr}_{\leq k}(*)$ represents all irreducible strings of finite length. A *duplication root* of $\boldsymbol{x}$ is a $\leq k$-irreducible string $\boldsymbol{r}$ such that $\boldsymbol{x}$ is a descendant of $\boldsymbol{r}$. Equivalently, $\boldsymbol{r}$ can be obtained from $\boldsymbol{x}$ by performing all possible deduplications of length at most $k$. The set of duplication roots of $\boldsymbol{x}$ is denoted $R_{\leq k}(\boldsymbol{x})$. Note that $R_{\leq k}(\boldsymbol{x}) \subseteq \mathrm{Irr}_{\leq k}(*)$. For $\leq$ 3-TDs, the work [3] showed that $R_{\leq 3}(\boldsymbol{x})$ has a single element. When $R_{\leq 3}(\boldsymbol{x})$ is a singleton, we may treat it as a string rather than a set. The uniqueness of the root for $k = 3$ implies that if $\boldsymbol{x}'$ is a descendant of $\boldsymbol{x}$, then $R_{\leq 3}(\boldsymbol{x}) = R_{\leq 3}(\boldsymbol{x}')$.

In addition to short TDs, this paper considers *substitution* errors. Let $D^{*,\leq p}_{\leq k}(\boldsymbol{x})$ denote the set of sequences obtained from $\boldsymbol{x}$ through an arbitrary number of $\leq k$-TDs and at most $p$ substitutions. Following (1), two substitutions occurring in $\boldsymbol{x}'$ and an additional duplication may lead to $\boldsymbol{x}'' \in D^{*,\leq 2}_{\leq k}(\boldsymbol{x})$:

$$\begin{aligned}\boldsymbol{x}' = 3101101121203 &\to 3101201321203, \\ &\to \boldsymbol{x}'' = 3101201201321203,\end{aligned}$$

where the changes resulting from the two substituted symbols are marked red.

For simplicity, when $k = 3$, we drop the $\leq$3 subscript and write $D^*(\cdot), R(\cdot), \mathrm{Irr}(\cdot)$, and $D^{*,\leq p}(\boldsymbol{x})$.

As we will see in the next section, the effect of duplications and substitutions on the duplication roots can be viewed as substring edits. Formally, a *substring edit* in a string $\boldsymbol{w} \in \Sigma_q^*$ is the operation of replacing a substring $\boldsymbol{u}$ with a string $\boldsymbol{v}$, where at least one of $\boldsymbol{u}, \boldsymbol{v}$ is nonempty. The length of the substring edit is $\max\{|\boldsymbol{u}|, |\boldsymbol{v}|\}$. An *$L$-substring edit* is one whose length is at most $L$. We note that since $\boldsymbol{u}$ and $\boldsymbol{v}$ can be empty strings (although not simultaneously), a substring edit can represent three types of error: deletion of a substring, insertion of a substring, or substituting a substring with another substring, possibly with different lengths.

## III. Channels with short duplication and substitution errors

In this section, we study channels with an arbitrary number of $\leq$3-TDs and at most $p$ substitutions and motivate our error-correction approach.

First, we consider channels with only short duplication errors (i.e., $\leq$ 3-TDs). Let $\boldsymbol{x}$ and $\boldsymbol{y} \in D^*(\boldsymbol{x})$ denote the input and output of the channel. Note that the duplication root of $\boldsymbol{x}$ is also the duplication root of $\boldsymbol{y}$. This fact, along with the uniqueness of duplication roots for short duplications, implies that the channel does not alter the duplication root. This observation was used in [3] to propose using the set of irreducible strings of length $n$ as a code for correcting an arbitrary number of duplications. This code was shown to be asymptotically optimal by [20].

The problem of correcting short duplications and an additional substitution was studied in [22], [25]. There, motivated by the use of roots for correcting duplication errors, the effect of the substitution on the duplication roots was studied.

**Theorem 1.** *[25, Theorem 3] There exists a (minimal) positive integer $\mathcal{L}$ such that for any $\boldsymbol{x}$ and $\boldsymbol{y} \in D^{*,\leq 1}(\boldsymbol{x})$, $R(\boldsymbol{y})$ can be obtained from $R(\boldsymbol{x})$ through an $\mathcal{L}$-substring edit.*

It was shown in [25] that $\mathcal{L} \leq 17$. Note that a priori, it is not clear that the effect of the substitution can be limited to a short substring since the substituted symbol may be duplicated as parts of substrings of different lengths an arbitrary number of times.

For the channel with many short duplications and at most $p$ substitutions, we can prove the following result.

**Theorem 2.** *For any $\boldsymbol{x} \in \Sigma^*$ and $\boldsymbol{y} \in D^{*,\leq p}(\boldsymbol{x})$, $R(\boldsymbol{y})$ can be obtained from $R(\boldsymbol{x})$ by at most $p$ $\mathcal{L}$-substring edit errors.*

*Proof:* Consider the sequence of substitution and duplication errors that transform $\boldsymbol{x}$ into $\boldsymbol{y}$. (Note that the errors may occur in any order.) For $i \in [p]$, let $\boldsymbol{y}_i$ be the sequence obtained just after the $i$th substitution error. Furthermore, let $\boldsymbol{y}_0 = \boldsymbol{x}$ and $\boldsymbol{y}_{p+1} = \boldsymbol{y}$. By Theorem 1, for $i \in [p]$, $R(\boldsymbol{y}_i)$ can be obtained from $R(\boldsymbol{y}_{i-1})$ via an $\mathcal{L}$-substring edit. Also, $R(\boldsymbol{y}_p) = R(\boldsymbol{y}_{p+1})$. Hence, $R(\boldsymbol{y}_0)$ can be transformed into $R(\boldsymbol{y}_{p+1})$ via $p$ $\mathcal{L}$-substring edits. ■

We next provide an example, demonstrating Theorem 2.

**Example 3.** *Let the alphabet be $\Sigma_4 = \{0, 1, 2, 3\}$ and $p = 2$. We take the input $\boldsymbol{x}$ to be irreducible, i.e., $R(\boldsymbol{x}) = \boldsymbol{x}$. By passing through the channel, $\boldsymbol{x}$ suffers multiple $\leq$3-TDs and 2 symbol substitutions, resulting in $\boldsymbol{y} \in D^{*,2}(\boldsymbol{x})$. We show the difference between $R(\boldsymbol{x})$ and $R(\boldsymbol{y})$ for two possible input-output pairs. Below, substrings added via duplication are marked with underlines, while substituted symbols are red and bold.*

*First, we provide an example where $R(\boldsymbol{y})$ can be obtained from $R(\boldsymbol{x})$ via non-overlapping substring edits:*

$$\begin{aligned}
\boldsymbol{x} &= 3210313230121321,\\
\boldsymbol{y} &= 321\underline{32\mathbf{0}321}031\underline{31}32\underline{132}3\mathbf{2}121321\underline{132}1,\\
R(\boldsymbol{x}) &= \underbrace{321}_{\boldsymbol{\alpha}_0}\underbrace{\phantom{000}}_{\boldsymbol{\beta}_1}\underbrace{031}_{\boldsymbol{\alpha}_1}\underbrace{3230121}_{\boldsymbol{\beta}_2}\underbrace{321}_{\boldsymbol{\alpha}_2},\\
R(\boldsymbol{y}) &= \underbrace{321}_{\boldsymbol{\alpha}_0}\underbrace{32\mathbf{0}321}_{\boldsymbol{\beta}_1'}\underbrace{031}_{\boldsymbol{\alpha}_1}\underbrace{\phantom{000}}_{\boldsymbol{\beta}_2'}\underbrace{321}_{\boldsymbol{\alpha}_2},
\end{aligned}$$

*where the errors are $\boldsymbol{\beta}_1 = \Lambda \to \boldsymbol{\beta}_1'$ and $\boldsymbol{\beta}_2 \to \boldsymbol{\beta}_2' = \Lambda$.*

*In the second case, the two edits overlap, leading to a single substring substitution:*

$$\begin{aligned}
\boldsymbol{x} &= 132031230,\\
\boldsymbol{y} &= 132\underline{32}03\underline{\mathbf{2}}1\underline{32\mathbf{0}321}230\underline{230230},\\
R(\boldsymbol{x}) &= \underbrace{13203}_{\boldsymbol{\alpha}_0}\underbrace{\phantom{000}}_{\boldsymbol{\beta}_1}\underbrace{1230}_{\boldsymbol{\alpha}_1}\\
R(\boldsymbol{y}) &= \underbrace{13203}_{\boldsymbol{\alpha}_0}\underbrace{\mathbf{2}132\mathbf{0}32}_{\boldsymbol{\beta}_1'}\underbrace{1230}_{\boldsymbol{\alpha}_1}.
\end{aligned}$$

*Generally, $t$ overlapping $\mathcal{L}$-substring edits result in a $(t\mathcal{L})$-substring edit.*

## IV. Error-correcting codes

We showed in Theorem 2 that short TDs and substitutions applied to strings manifest as $\mathcal{L}$-substring edit errors on their duplication roots. Using this fact, in this section, we construct error-correcting codes that can correct an arbitrary number of short TDs and at most $p$ substitutions by correcting $\leq p$ $\mathcal{L}$-substring edits. We will also present the rate of the proposed codes.

### A. Code construction

To correct an arbitrary number of TDs and at most $p$ symbol substitutions, it suffices to construct error-correcting codes over $\leq$3-irreducible strings that correct at most $p$ $\mathcal{L}$-substring edits in the duplication roots of the codewords. Given that the substring edits are bounded in length, similar to [22], we divide the codewords into message blocks, separated by markers, such that an $\mathcal{L}$-substring edit only affects a limited number of message blocks. In the case of $p = 1$ studied in [22], it was shown that if the markers appear in the correct positions in the retrieved word, then at most two of the message blocks are substituted. For $p > 1$ however, even if all markers are in the correct positions, all message blocks may be substituted, making it challenging to correct more than one error.

We start by recalling an auxiliary construction from [22].

**Construction 4.** *[22, Construction 3] Let $l, m, N$ be positive integers with $m > l \geq 5$ and $\boldsymbol{\sigma} \in \mathrm{Irr}(l)$. Also, let $\mathcal{B}_{\boldsymbol{\sigma}}^m$ denote the set of sequences $B$ of length $m$ such that $\boldsymbol{\sigma}B\boldsymbol{\sigma}$ is irreducible and has exactly two occurrences of $\boldsymbol{\sigma}$. Define*

$$\mathcal{C}_{\boldsymbol{\sigma}} = \{B_1\boldsymbol{\sigma}B_2\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_N : B_i \in \mathcal{B}_{\boldsymbol{\sigma}}^m\}.$$

The irreducibility of $\boldsymbol{\sigma}B_i\boldsymbol{\sigma}$ ensures that the codewords are irreducible.

We denote the output of the channel by $\boldsymbol{y}$. Define a *block* in $\boldsymbol{y}$ as a maximal substring that does not overlap with any $\boldsymbol{\sigma}$. Furthermore, define an $m$*-block* in $\boldsymbol{y}$ as a block with length $m$. Note that $m$-blocks can be either message blocks in $\boldsymbol{x}$ or new blocks created by substring edits.

Having divided each codeword into $N$ message blocks and $N-1$ separators, we study in the next lemma how message blocks are affected by the errors.

**Lemma 5.** *Let $\boldsymbol{x} \in \mathcal{C}_{\boldsymbol{\sigma}}$, $m > \mathcal{L}$, and $\boldsymbol{y}$ be generated from $\boldsymbol{x}$ through at most $p$ $\mathcal{L}$-substring edits. Then there are less than $(N + p)$ $m$-blocks in $\boldsymbol{y}$. Furthermore, there are at least $N-2p$ error-free $m$-blocks in $\boldsymbol{y}$ which appear in $\boldsymbol{x}$ in the same order. More precisely, there are blocks $B_{i_1}, B_{i_2}, \ldots, B_{i_k}$ in $\boldsymbol{y}$, where $k \geq N - 2p$, each $B_{i_j}$ is a message block in $\boldsymbol{x}$, and any two blocks $B_{i_j}$ and $B_{i_{j'}}$ have the same relative order of appearance in $\boldsymbol{x}$ and in $\boldsymbol{y}$.*

*Proof:* First suppose $\boldsymbol{y}$ has $\geq (N + p)$ message blocks. This implies that the length of $\boldsymbol{y}$ is at least $(N+p)m+(N+p-1)l$, which is larger than the length of $\boldsymbol{x}$ by $pm+(p-1)l$. But this is not possible as $m > \mathcal{L}$ and the total length of inserted substrings is at most $p\mathcal{L}$.

Furthermore, if $m > \mathcal{L}$, each $\mathcal{L}$-substring edit alters i) a message block in $\boldsymbol{x}$, ii) a message block and a marker $\boldsymbol{\sigma}$, or iii) two message blocks and the marker between them. Hence at least $N-2p$ message blocks of $\boldsymbol{x}$ appear in $\boldsymbol{y}$ without being changed. ■

If the positions of the error-free $m$-blocks described in Lemma 5 in $\boldsymbol{y}$ were known, a Reed-Solomon (RS) code of length $N$ and dimension $N-2p$ could be used to recover codewords in $\mathcal{C}_{\boldsymbol{\sigma}}$. This however is not the case since the blocks can be shifted by substring edits. In order to determine the positions of the error-free $m$-blocks, we introduce another auxiliary construction based on Construction 4 by combining message blocks as *message groups*, where the message blocks in each group have different "colors".

**Construction 6.** *For an integer $T$, we partition $\mathcal{B}_{\boldsymbol{\sigma}}^m$ into $T$ parts $\mathcal{B}_{\boldsymbol{\sigma}}^m(j), j\in[T]$. The elements of $\mathcal{B}_{\boldsymbol{\sigma}}^m(j)$ are said to have* color $j$. *Let $\hat{N}$, $N$, and $T$ be integers such that $N=T\hat{N}$. We define the code*

$$\mathcal{C}_{(\boldsymbol{\sigma},T)}=\left\{B_1\boldsymbol{\sigma}B_2\boldsymbol{\sigma}\cdots B_N\in\mathcal{C}_{\boldsymbol{\sigma}}:B_i\in\mathcal{B}_{\boldsymbol{\sigma}}^m(i \bmod' T)\right\},$$

*where $\mathcal{C}_{\boldsymbol{\sigma}}$ has parameters $m,l$ with $m>\mathcal{L}>l\geq 5$. For $\boldsymbol{x}\in\mathcal{C}_{(\boldsymbol{\sigma},T)}$, we define the $k$-th* message group *$S_k$ as $S_k=(B_{(k-1)T+1},\ldots,B_{kT-1},B_{kT})$, $k\in[\hat{N}]$. Note that the message blocks in each message group have colors $1,2,\ldots,T$ in order.*

For example, if $N=12, T=3, \hat{N}=4$, then in a codeword

$$\boldsymbol{x}=B_1\boldsymbol{\sigma}B_2\boldsymbol{\sigma}B_3\boldsymbol{\sigma}B_4\boldsymbol{\sigma}B_5\boldsymbol{\sigma}B_6\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_{10}\boldsymbol{\sigma}B_{11}\boldsymbol{\sigma}B_{12},$$

the first group is $(B_1,B_2,B_3)$ and the second group is $(B_4,B_5,B_6)$. Furthermore, message blocks in both groups have colors $(1,2,3)$. The colors in the message group will help us identify the true position of the message blocks.

**Definition 7.** *For $\boldsymbol{x}\in\mathcal{C}_{(\boldsymbol{\sigma},T)}$ and $\boldsymbol{y}$ derived from $\boldsymbol{x}$ through at most $p$ $\mathcal{L}$-substring edits, let the $i$-th $m$-block in $\boldsymbol{y}$ be denoted by $B'_i$. A $T$-*group *in $\boldsymbol{y}$ is a substring $B'_{k+1}\boldsymbol{\sigma}B'_{k+2}\cdots\boldsymbol{\sigma}B'_{k+T}$ such that the $m$-block $B'_{k+j}$ has color $j$.*

The next lemma characterizes how error-free message groups (those that do not suffer any substring edits but may be shifted) appear in $\boldsymbol{y}$.

**Lemma 8.** *Suppose $\boldsymbol{x}\in\mathcal{C}_{(\boldsymbol{\sigma},T)}$ and let $\boldsymbol{y}$ be obtained from $\boldsymbol{x}$ through at most $p$ $\mathcal{L}$-substring edits. For $r\in[\hat{N}]$, if the $r$-th message group in $\boldsymbol{x}$ is not affected by any substring edit errors, then it will appear as a $T$-group after $b$ $m$-blocks in $\boldsymbol{y}$, where $b\in[(r-1)T-2p,(r-1)T+p-1]$.*

*Proof:* Since $m>\mathcal{L}$, each $\mathcal{L}$-substring edit can affect at most two message blocks and thus at most two message groups. Hence, there are at least $\hat{N}-2p$ message groups that do not suffer any substring edits.

Let the $r$-th message group $S_r$ in $\boldsymbol{x}$ be free of substring edits. Given that the colors of its message blocks are not altered, it will appear as a $T$-group in $\boldsymbol{y}$. Since each substring edit alters at most two message blocks, among the $(r-1)T$ message blocks appearing before $S_r$ in $\boldsymbol{x}$, at most $2p$ do not appear in $\boldsymbol{y}$. Furthermore, the substring edits add at most $p\mathcal{L}$ to the length of $\boldsymbol{x}$. Since $m>\mathcal{L}$, this means that at most $p-1$ new $m$-blocks are created in $\boldsymbol{y}$. Hence, $b\in[(r-1)T-2p,(r-1)T+p-1]$. ∎

The previous lemma guarantees the presence of error-free, but possibly shifted, $T$-groups, and provides bounds on their position in $\boldsymbol{y}$. In the following theorem, we use these facts to show that these $T$-groups can be synchronized and the errors can be localized.

**Theorem 9.** *Let $\mathcal{C}_{(\boldsymbol{\sigma},T)}$ be a code in Construction 6 and suppose $T\geq 3p$ and $\hat{N}\geq 4p+1$. There is a decoder $\mathcal{D}$ such that, for any $\boldsymbol{x}\in\mathcal{C}_{(\boldsymbol{\sigma},T)}$ and $\boldsymbol{y}$ derived from $\boldsymbol{x}$ through at most $p$ $\mathcal{L}$-substring edits, $\boldsymbol{v}=\mathcal{D}(\boldsymbol{y})$ suffers at most $t$ substitutions and $e$ erasures of message groups, where $t+e\leq 2p$.*

*Proof:* We start by identifying all $T$-groups in $\boldsymbol{y}$. Note that no two $T$-groups can overlap. Let $\boldsymbol{v}=(S'_1,\ldots,S'_{\hat{N}})$ be the decoded vector, where $S'_r$ is the decoded version of the message group $S_r$, determined as follows.

For $r=1,\ldots,\hat{N}$:

1) If there exists a $T$-group $\mathcal{T}$ appearing after $b$ message blocks such that $b\in[(r-1)T-2p,(r-1)T+p-1]$, then let $S'_r=\mathcal{T}$.
2) If such a $T$-group does not exist, let $S'_r=\Lambda$, denoting an erasure.

We note that for each $r$, at most one $T$-group may satisfy the condition in 1). If two such $T$-groups exist appearing after $b$ and $b'$ message blocks, we must have $|b-b'|\geq T$ and $b,b'\in[(r-1)T-2p,(r-1)T+p-1]$, implying $3p-1\geq T$, which contradicts the assumption on $T$.

If a message group $S_r$ is not subject to a substring edit, then by Lemma 8, we have $S'_r=S_r$. Otherwise, we may have a substitution of that message group, i.e., $S'_r\neq S_r$, or an erasure, $S'_r=\Lambda$. Since each substring edit may affect at most 2 message groups, the total number of substitutions and erasures is no more than $2p$. ∎

We now construct an MDS code that can correct the output of the decoder of Theorem 9.

**Construction 10.** *Let $\mathcal{C}_{(\boldsymbol{\sigma},T)}$ be the code in Construction 6 with parameters $l,m,T,\hat{N}$ satisfying $m>\mathcal{L}>l\geq 5, T\geq 3p$, and $\hat{N}\geq 4p+1$. Furthermore, assume $|\mathcal{B}_{\boldsymbol{\sigma}}^m(j)|\geq\hat{N}+1$ for $j\in[T]$. Finally, let $\gamma$ be a positive integer such that $2^\gamma\leq\hat{N}+1<2^{\gamma+1}$ and $\zeta_j:\mathbb{F}_{2^\gamma}\to\mathcal{B}_{\boldsymbol{\sigma}}^m(j)$ be a one-to-one mapping for $j\in[T]$. We define $\mathcal{C}_{MDS}$ as*

$$\begin{aligned}\mathcal{C}_{MDS}=&\{\zeta_1(c_1^1)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_j(c_1^j)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_T(c_1^T)\boldsymbol{\sigma}\\&\boldsymbol{\sigma}\zeta_1(c_2^1)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_j(c_2^j)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_T(c_2^T)\boldsymbol{\sigma}\cdots\\&\boldsymbol{\sigma}\zeta_1(c_{\hat{N}}^1)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_j(c_{\hat{N}}^j)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_T(c_{\hat{N}}^T):\\&\{\boldsymbol{c}^j,j\in[T]\}\subseteq\mathrm{MDS}(\hat{N},\hat{N}-4p,4p+1)\},\end{aligned}$$

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

For each $j$, we also define an inverse $\zeta_j^{-1}$ for $\zeta_j$. For $B\in\mathcal{B}_{\boldsymbol{\sigma}}^m(j)$, if $\beta\in\mathbb{F}_{2^\gamma}$ such that $\zeta_j(\beta)=B$ exists, then let $\zeta_j^{-1}(B)=\beta$. Otherwise, let $\zeta_j^{-1}(B)=0$.

**Theorem 11.** *The error-correcting codes $\mathcal{C}_{MDS}$ in Construction 10 can correct any number of $\leq$3-TDs and at most $p$ symbol substitutions.*

*Proof:* Given a codeword $\boldsymbol{x} \in \mathcal{C}_{MDS}$, let $\boldsymbol{x}'' \in D^{*,\leq p}(\boldsymbol{x})$ be obtained from $\boldsymbol{x}$ via any number of $\leq$3-TDs and at most $p$ symbol substitutions, and let $\boldsymbol{y} = R(\boldsymbol{x}'')$. Note that by construction, $\boldsymbol{x}$ is irreducible. Thus, by Theorem 2, $\boldsymbol{y}$ can be obtained from $\boldsymbol{x}$ through at most $p$ $\mathcal{L}$-substring edits. As $\mathcal{C}_{MDS} \subseteq \mathcal{C}_{(\boldsymbol{\sigma},T)}$, based on Theorem 9, $\boldsymbol{v} = \mathcal{D}(\boldsymbol{y})$ such that at most $t$ substitutions and $e$ erasures of message groups occur, where $t + e \leq 2p$. Hence, for $j \in [T]$, the blocks $(\zeta_j(c_1^j), \zeta_j(c_2^j), \ldots, \zeta_j(c_{\hat{N}}^j))$ suffer at most $2p$ erasures or substitutions. Consequently, if we apply $\zeta_j^{-1}$ to the corresponding retrieved blocks in $\boldsymbol{v}$, the codeword $(c_1^j, c_2^j, \ldots, c_{\hat{N}}^j)$ also suffers at most $2p$ substitutions or erasures, which can be corrected using the MDS code. ■

We note that the decoding algorithm needs to identify the color of sequences $B \in \mathcal{B}_{\boldsymbol{\sigma}}^m$. To do this efficiently, we consider each part in the partition of $\mathcal{B}_{\boldsymbol{\sigma}}^m$ to be a contiguous block in the lexicographically sorted list of the elements of $\mathcal{B}_{\boldsymbol{\sigma}}^m$. Then, with the knowledge of the first element of each part, we can identify the color of a given $B \in \mathcal{B}_{\boldsymbol{\sigma}}^m$ in $\log T$ time.

## *B. Code rate*

In this subsection, we present choices for the parameters of Construction 10 and discuss the rate of the resulting code.

Among the $n$ symbols of each codeword in Construction 10, $4pTm + (\hat{N}T - 1)l$ symbols belong to MDS parities or markers. We choose for $T$ and $l$ to be their smallest possible values and set $T = 3p$ and $l = 5$.

The construction requires that $|\mathcal{B}_{\boldsymbol{\sigma}}^m(j)| \geq \hat{N}+1$ for all $j$. Let $M_{\boldsymbol{\sigma}}^{(m)} = |\mathcal{B}_{\boldsymbol{\sigma}}^m|$. Dividing $\mathcal{B}_{\boldsymbol{\sigma}}^m$ into parts of nearly equal size, we find that each part $\mathcal{B}_{\boldsymbol{\sigma}}^m(j)$ has size at least $M_{\boldsymbol{\sigma}}^{(m)}/T - 1$. We then choose $\hat{N} + 1$ as the largest power of two not larger than $M_{\boldsymbol{\sigma}}^{(m)}/T - 1$, ensuring that $\hat{N}+1 \geq M_{\boldsymbol{\sigma}}^{(m)}/(2T) - (1/2)$. Assume

$$M_{\boldsymbol{\sigma}}^{(m)} \geq 24p^2 + 15p. \tag{2}$$

Then $\hat{N} + 1 \geq M_{\boldsymbol{\sigma}}^{(m)}/(2T) - (1/2) \geq 4p + 2$.

Furthermore, note that $\hat{N}T(m+5) - 5 = n$ and thus $\hat{N} = \frac{n+5}{(m+5)(3p)}$. The size of the code then becomes

$$|\mathcal{C}_{MDS}| = (\hat{N} + 1)^{(\hat{N}-4p)(3p)},$$

and

$$\begin{aligned}
\log|\mathcal{C}_{MDS}| &\geq \left(\frac{n}{m+5} - 12p^2\right) \log\left(\frac{M_{\boldsymbol{\sigma}}^{(m)}}{6p} - \frac{1}{2}\right) \\
&\geq \left(\frac{n}{m+5} - 12p^2\right) \left(\log M_{\boldsymbol{\sigma}}^{(m)} + \log\left(\frac{1}{6p} - \frac{1}{2M_{\boldsymbol{\sigma}}^{(m)}}\right)\right) \\
&\geq \left(\frac{n}{m+5} - 12p^2\right) \left(\log M_{\boldsymbol{\sigma}}^{(m)} - \log(6p+1)\right), \qquad (3)
\end{aligned}$$

where in the last step we have used the fact that $M_{\boldsymbol{\sigma}}^{(m)} \geq 24p^2 + 15p$.

It was shown in [22] that $M_{\boldsymbol{\sigma}}^{(m)} \geq (q-2)^{m-c_q}$ for some $\boldsymbol{\sigma}$, where $c_q$ is a constant independent of $m$. In particular, $c_3 \leq 13, c_4 \leq 6$, and $c_q \leq 5$ for $q \geq 5$. To satisfy (2), we need

$$m \geq \max\{\log_{q-2}(24p^2 + 15p) + c_q, \mathcal{L} + 1\}. \tag{4}$$

Given a code $\mathcal{C}(n)$ with length $n$ and the code size $|\mathcal{C}(n)|$, the code rate is defined as $R(\mathcal{C}(n)) = \frac{1}{n}\log|\mathcal{C}(n)|$. From (3), for the rate of $\mathcal{C}_{MDS}$,

$$\begin{aligned}
R(\mathcal{C}_{MDS}) &\geq \left(\frac{m - c_q}{m+5} - \frac{12p^2 m}{n}\right)\log(q-2) - \frac{\log(6p+1)}{m+5} \\
&\geq \left(1 - \frac{c_q + 5}{m+5} - \frac{12p^2 m}{n}\right)\log(q-2) - \frac{\log(6p+1)}{m+5},
\end{aligned}$$

where $m$ satisfies (4). For $\log p = o(\log n)$, letting $m = \Theta(\log n)$, we find that the rate asymptotically satisfies

$$R(\mathcal{C}_{MDS}) \geq \log(q-2)(1 - o(1))$$

We note that the rate of the code that only corrects duplications is bounded above by $\log(q-1)$.

## References

[1] D. Reinsel, J. Rydning, and J. Gantz, "Worldwide global datasphere forecast, 2020–2024: The covid-19 data bump and the future of data growth," *Int. Data Corp.(IDC), Framingham, MA, USA, Tech. Rep. US44797920*, 2020.

[2] H. H. Lee, R. Kalhor, N. Goela, J. Bolot, and G. M. Church, "Terminator-free template-independent enzymatic DNA synthesis for digital information storage," *Nature communications*, vol. 10, no. 1, pp. 1–12, 2019.

[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] 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.

[5] 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.

[6] S. L. Shipman, J. Nivala, J. D. Macklis, and G. M. Church, "CRISPR–Cas encoding of a digital movie into the genomes of a population of living bacteria," *Nature*, vol. 547, no. 7663, pp. 345–349, Jul. 2017.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

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

[25] ——, "Error-correcting codes for short tandem duplication and substitution errors," *arXiv preprint arXiv:2011.05896*, 2020.