Correcting multiple short duplication and substitution errors

Yuanyuan Tang*, Shuche Wang†, Ryan Gabrys‡, and Farzad Farnoud*
* Electrical & Computer Engineering, University of Virginia, U.S.A., {yt5tz,farzad}@virginia.edu
† Institute of Operations Research and Analytics, National University of Singapore, shuche.wang@u.nus.edu
‡ Calit2, University of California-San Diego, U.S.A., rgabrys@eng.ucsd.edu

***Abstract*—Due to its higher data density, longevity, energy efficiency, and ease of generating copies, DNA is considered a promising storage technology for satisfying future needs. However, a diverse set of errors including deletions, insertions, duplications, and substitutions may arise in DNA at different stages of data storage and retrieval. The current paper constructs error-correcting codes for simultaneously correcting short (tandem) duplications and at most p substitutions, where a short duplication generates a copy of a substring with length ≤ 3 and inserts the copy following the original substring. Compared to the state-of-the-art codes for duplications only, the proposed codes correct up to p substitutions (in addition to duplications) at the additional cost of roughly $8p(\log_q n)(1+o(1))$ symbols of redundancy, thus achieving the same asymptotic rate, where $q \geq 4$ is the alphabet size. Furthermore, the time complexities of both the encoding and decoding processes are polynomial when p is a constant with respect to n.**

I. Introduction

With recent advances in sequencing and biological synthesis, deoxyribonucleic acid (DNA) is considered a promising candidate for satisfying future data storage needs [1], [2]. In particular, experiments in [1], [3]–[7] demonstrate that data can be stored on and subsequently retrieved from DNA. Compared to traditional data storage media, DNA has the advantages of higher data density, longevity, energy efficiency, and ease of generating copies [1], [7]. However, a diverse set of errors may occur at different stages of the data storage and retrieval process, such as deletions, insertions, duplications, and substitutions. Many recent works, such as [7]–[24], have been devoted to protecting the data against these errors. The current paper constructs error-correcting codes for duplication and substitution errors.

A *(tandem) duplication* in a DNA sequence generates a copy of a substring and then inserts it directly following the original substring [8], where the duplication length is the length of the copy. For example, given ACTG, a tandem duplication may generate ACTCTG, where CTCT is a *(tandem) repeat*. Bounded-length duplications are those whose length is at most a given constant. In particular, we refer to duplications of length at most 3 as *short duplications*. Correcting fixed-length duplications [8], [10]–[12], [25] and bounded-length duplications [8], [23], [26]–[29] have been both studied recently. In particular, the code in [8], which has a polynomial-time encoder, provides the highest known asymptotic rate for correcting any number of short duplications. For an alphabet of size q, the rate is $\log r$, where r is the largest positive real root of $x^3-(q-2)x^2-(q-3)x-(q-2)=0$. For $q=4$ this rate is $\log 2.6590$ and as q increases, the rate is approximately $\log(q-1)$ [23].

For channels with both duplication and substitution errors, *restricted* substitutions [12], [25], which occur only in duplicated copies, and *unrestricted* substitutions [12], [29]–[31] have been studied. The closest work to the current paper, [31], constructed error-correcting codes for short duplications and at most p unrestricted substitutions with an asymptotic code rate lower bounded by $\log(q-2)$ [31]. However, compared to the codes for only duplications [8], the codes in [31] incur an asymptotic rate loss in order to correct the additional $\leq p$ substitutions. The current paper focuses on constructing error-correcting codes for short duplications and at most p (unrestricted) substitutions with significantly less redundancy than the approach from [31]. We note that the short duplications and substitutions can occur in an arbitrary order.

One of the challenging aspects of correcting duplications and at most p substitutions simultaneously is that a single substitution may be duplicated many times and affect an unbounded segment. However, assuming that the input to the channel is irreducible, i.e., it has no repeats of length ≤ 3, after removing all tandem copies with length ≤ 3 from the output, the effects of short duplications and at most p substitutions can be localized in at most p substrings, each with length ≤ 17 [29], [31]. Therefore, similar to [31], we construct our error-correcting codes as a subset of irreducible strings, but leverage the *syndrome compression* technique to substantially reduce the redundancy. Syndrome compression has been recently used to provide explicit constructions for correcting a wide variety of errors with the redundancy as low as roughly twice the Gilbert-Varshamov bound [32]–[35]. More specifically, we protect the data by appending a vector with length around $8p(\log_q n)(1+o(1))$ to each input of length n, where the appended vector is used to distinguish all confusable inputs. We ensure that the appended vector is itself protected against errors and can be decoded correctly. We then use it to recover the data by eliminating incorrect confusable inputs. Compared to the explicit code for duplications only [8], the proposed code corrects $\leq p$ substitutions in addition to duplications at the extra cost of roughly $8p(\log_q n)(1+o(1))$ symbols of redundancy for $q \geq 4$, and achieves the same asymptotic code rate. This improves

This work was supported in part by NSF grants CIF 1816409 and CIF 1755773.

upon the approach from [31], which suffers an asymptotic rate loss. Both time complexities of the encoding and decoding processes are polynomial when p is a constant.

The paper is organized as follows. Section II presents the notation and preliminaries. In Section III, we derive an upper bound on the size of the confusable set for an irreducible string, which is a key step of the syndrome compression technique used to construct our error-correcting codes. Finally, Section IV presents the code construction, as well as a discussion of the redundancy and an analysis of the encoding and decoding complexities. Due to lack of space, some of the proofs are omitted or only sketched.

II. Notation and Preliminaries

Let $\Sigma_q = \{0, 1, 2, \cdots, q-1\}$ represent a finite alphabet of size q, Σ_q^n the set of all strings of length n over Σ_q, and Σ_q^* the set of all finite strings over Σ_q. Given two integers a, b with $a \leq b$, the set $\{a, a+1, \cdots, b\}$ is shown as $[a, b]$. We simplify $[1, b]$ as $[b]$. Unless otherwise stated, logarithms are to the base 2.

We use bold symbols to denote strings over Σ_q, i.e., $\boldsymbol{x}, \boldsymbol{y}_j \in \Sigma_q^*$. The entries of a string are represented by plain typeface, e.g., the ith elements of $\boldsymbol{x}, \boldsymbol{y}_j \in \Sigma_q^*$ are $x_i, y_{ji} \in \Sigma_q$ respectively. For two strings $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, let $\boldsymbol{xy}$ denote their concatenation. Given four strings $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \Sigma_q^*$, if $\boldsymbol{x} = \boldsymbol{uvw}$, then $\boldsymbol{v}$ is called a substring of $\boldsymbol{x}$. Furthermore, we let $|\boldsymbol{x}|$ represent the length of a string $\boldsymbol{x} \in \Sigma_q^n$, and let $\|S\|$ denote the size (the number of elements) of a set S.

A *(tandem) duplication* (TD) of length k (k-TD) is the operation of generating a copy of a substring and inserting it directly following the substring, where k is the length of the copy. For example, for $\boldsymbol{x} = \boldsymbol{uvw}$ with $|\boldsymbol{v}| = k$, a k-TD may generate $\boldsymbol{uvvw}$, where $\boldsymbol{vv}$ is called a *(tandem) repeat* with length $2k$. A duplication of length at most k is denoted as a $\leq k$-TD. We focus on $\leq k$-TDs with $k = 3$, which we call *short duplications*. For example, given $\boldsymbol{x} = 213012 \in \Sigma_4^*$, a sequence of $\leq$3-TDs may produce

$$\begin{aligned} \boldsymbol{x} &= 213012 \to 213\underline{213}012 \to 21321303\underline{0}12 \\ &\to 2132\underline{2}1303012 = \boldsymbol{x}', \end{aligned} \tag{1}$$

where the duplicated copies are marked with underlines. We call $\boldsymbol{x}'$ a *descendant* of $\boldsymbol{x}$, i.e., a string generated from $\boldsymbol{x}$ by a sequence of $\leq$3-TDs. Furthermore, for a string $\boldsymbol{x} \in \Sigma_q^*$, let $D_{\leq k}^*(\boldsymbol{x})$ be the set of all descendants generated from $\boldsymbol{x}$ by an arbitrary number of $\leq k$-TDs.

A *deduplication* of length k replaces a repeat $\boldsymbol{vv}$ by $\boldsymbol{v}$ with $|\boldsymbol{v}| = k$. Then $\leq k$-deduplications are deduplications with length upper bound by k. In this paper, we focus on ≤ 3-deduplications, simply called deduplications in the rest of the paper. For example, the string $\boldsymbol{x}$ in (1) can be recovered from $\boldsymbol{x}'$ by three deduplications.

The set of *$\leq k$-irreducible strings* of length n, denoted $\mathrm{Irr}_{\leq k}(n)$, consists of strings without repeats $\boldsymbol{vv}$, where $|\boldsymbol{v}| \leq k$. Furthermore, $\mathrm{Irr}_{\leq k}(*)$ represents all $\leq k$-irreducible strings of finite length. A *duplication root* of $\boldsymbol{x}'$ is a $\leq k$-irreducible string $\boldsymbol{x}$ such that $\boldsymbol{x}'$ is a descendant of $\boldsymbol{x}$. Equivalently, $\boldsymbol{x}$ can be obtained from $\boldsymbol{x}'$ by performing all possible $\leq k$-deduplications. The set of duplication roots of $\boldsymbol{x}'$ is denoted $R_{\leq k}(\boldsymbol{x}')$, i.e., $R_{\leq k}(\boldsymbol{x}') = \{\boldsymbol{x} \in \mathrm{Irr}(*) \mid \boldsymbol{x}' \in D_{\leq k}^*(\boldsymbol{x})\}$. For $\leq$3-TDs, the work [8] showed that $R_{\leq 3}(\boldsymbol{x}')$ is a singleton, and so we treat it as a string instead of a set. The uniqueness of the root for $k = 3$ implies that if $\boldsymbol{x}''$ is a descendant of $\boldsymbol{x}'$, we have $R_{\leq 3}(\boldsymbol{x}') = R_{\leq 3}(\boldsymbol{x}'')$.

Besides $\leq$ 3-TDs, we consider substitution errors, where each substitution replaces a symbol by another one from the same alphabet. Continuing the example in (1), two substitutions and two duplications applied to $\boldsymbol{x}'$ may produce

$$\begin{aligned} \boldsymbol{x}' &= 213221303012 \to 21321\textcolor{red}{1}303012 \to 2132\textcolor{red}{1}3\underline{21}1303012 \\ &\to 2132\textcolor{red}{1}321\textcolor{red}{1}323012 \to 2132\textcolor{red}{1}321\textcolor{red}{1}32\textcolor{red}{3}3\underline{23}012 = \boldsymbol{x}'', \end{aligned}$$

where the substituted symbols are marked in red. Let $D_{\leq k}^{\alpha,p}(\boldsymbol{x})$ represent the set of strings derived from $\boldsymbol{x}$ by $\alpha \leq k$-TDs and p substitutions. Furthermore, let $D_{\leq k}^{*,\leq p}(\boldsymbol{x})$ represent the set of strings generated by an arbitrary number of $\leq k$-TDs and at most p substitutions. In the example above, we have $\boldsymbol{x}'' \in D_{\leq k}^{*,\leq 2}(\boldsymbol{x})$.

For simplicity, when $k = 3$, we drop the $\leq$3 subscript and write $D^*(\cdot), R(\cdot), \mathrm{Irr}(\cdot), D^{\alpha,p}(\boldsymbol{x})$, and $D^{*,\leq p}(\boldsymbol{x})$. In the rest of the paper, unless otherwise stated, duplications are assumed to be $\leq$3-TDs, and irreducible strings represent $\leq$3-irreducible strings.

We define a *substring edit* in a string $\boldsymbol{x} \in \Sigma_q^*$ as 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. Furthermore, an *L-burst deletion* in $\boldsymbol{x} \in \Sigma_q^*$ is defined as removing a substring $\boldsymbol{v}$ of $\boldsymbol{x}$, where $|\boldsymbol{v}| = L$ is the length of the burst deletion.

Given a sequence $\boldsymbol{x} \in \Sigma_q^n$, we define the binary representation matrix $\mathcal{U}(\boldsymbol{x})$ of $\boldsymbol{x}$ as

$$\begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{\lceil \log q \rceil, 1} & u_{\lceil \log q \rceil, 2} & \cdots & u_{\lceil \log q \rceil, n} \end{bmatrix} \in \{0,1\}^{\lceil \log q \rceil \times n}, \tag{2}$$

where the jth column of $\mathcal{U}(\boldsymbol{x})$ is the binary representation of the jth symbol of $\boldsymbol{x}$ for $j \in [n]$. The ith row of $\mathcal{U}(\boldsymbol{x})$ is denoted as $\mathcal{U}_i(\boldsymbol{x})$ for $i \in \lceil \log q \rceil$.

In order to construct error-correcting codes by applying the syndrome compression technique [32], we first introduce some auxiliary definitions and a theorem.

Suppose $q \geq 3$ is a constant. We start with the definition of confusable sets for a given channel and a given set of strings $S \subseteq \Sigma_q^n$. In our application, S is the set of irreducible strings, upon which the proposed codes will be constructed.

Definition 1. *A* confusable set $B(\boldsymbol{x}) \subseteq S$ *of* $\boldsymbol{x} \in S$ *consists of all* $\boldsymbol{y} \in S$*, excluding* $\boldsymbol{x}$*, such that* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *can produce the same output when passed through the channel.*

Definition 2. *Let* $\mathcal{R}(n)$ *be an integer function of* n*. A* labeling function for the confusable sets $B(\boldsymbol{x}), \boldsymbol{x} \in S$*, is a function*

$$f : \Sigma_q^n \to \Sigma_{2^{\mathcal{R}(n)}}$$

such that, for any $\boldsymbol{x} \in S$ *and* $\boldsymbol{y} \in B(\boldsymbol{x})$*,* $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$*.*

Theorem 3. *(c.f. [32, Theorem 5]) Let $f : \Sigma_q^n \to \Sigma_{2^{\mathcal{R}(n)}}$, where $\mathcal{R}(n) = o(\log\log n \cdot \log n)$, be a labeling function for the confusable sets $B(\boldsymbol{x}), \boldsymbol{x} \in S$. Then there exists an integer $a \leq 2^{\log \|B(\boldsymbol{x})\| + o(\log n)}$ such that for all $\boldsymbol{y} \in B(\boldsymbol{x})$, we have $f(\boldsymbol{x}) \not\equiv f(\boldsymbol{y}) \bmod a$.*

The above definitions and theorem are used in our code construction based on syndrome compression, presented in Section IV. The construction and analysis rely on the confusable sets for the channel, discussed in the next section.

III. Confusable sets for channels with short duplication and substitution errors

In this section, we study the confusable sets of input strings passing through channels with an arbitrary number of $\leq$3-TDs and at most p substitutions.

Since $\leq$3-TDs and deduplications do not alter the duplication root of the input and because the duplication root is unique, $\mathrm{Irr}(n)$ is a code capable of correcting $\leq$3-TDs. The decoding process simply removes all tandem repeats. In other words, if we append a deduplication block, which removes all repeats, to the channel with duplication errors, any irreducible sequence passes through this concatenated channel with no errors. This approach produces codes with the same asymptotic rate as that of [8], achieving the highest known asymptotic rate.

Similar to [31], we extend this strategy to correct duplication and substitution errors. First, we take the code to be a subset of irreducible strings. Second, we find the code for a new channel obtained by concatenating a deduplication block to the channel with duplication and substitution errors (recall that duplications and substitution errors can occur in any order). Denote the channel that introduces any number of duplications and $\leq p$ substitutions, followed by a deduplication block that removes all repeats, as the DSD(p) channel. It is clear that an error-correcting code for the DSD(p) channel is also an error-correcting code for the channel with duplications and $\leq p$ substitutions. We define the confusable sets over $\mathrm{Irr}(n)$ for the DSD(p) channel.

Definition 4. *Suppose $\boldsymbol{x} \in \mathrm{Irr}(n)$ is an irreducible string of length n. Let*

$$B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathrm{Irr}(n) : \boldsymbol{y} \neq \boldsymbol{x}, \\ R(D^{*,\leq p}(\boldsymbol{x})) \cap R(D^{*,\leq p}(\boldsymbol{y})) \neq \varnothing\} \tag{3}$$

denote the irreducible-confusable set *of $\boldsymbol{x}$.*

We now find a bound on $\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\|$, which is needed to construct the code and determine its rate. Since deduplications can be undone by duplications, instead of the DSD(p) channel, we can consider a concatenation of p DSD(1) channels, without reducing the size of the confusable set. The input of each DSD(1) channel suffers a number of duplications, at most one substitution, and then all possible deduplications. Fig. 1 shows a confusable string $\boldsymbol{z}$, obtainable from both $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{Irr}(n)$, after passing through p DSD(1) channels, each represented by a solid arrow. More precisely, $\boldsymbol{x}_i \in R(D^{*,\leq 1}(\boldsymbol{x}_{i-1}))$ and $\boldsymbol{y}_i \in R(D^{*,\leq 1}(\boldsymbol{y}_{i-1}))$, where $\boldsymbol{x} = \boldsymbol{x}_0, \boldsymbol{y} = \boldsymbol{y}_0, \boldsymbol{z} = \boldsymbol{x}_p = \boldsymbol{y}_p$.

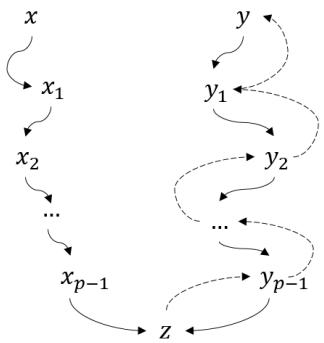

Figure 1. Confusable strings for a concatenation of channels with deduplications, at most 1 substitution, and all possible deduplications.

Lemma 5. *Given an irreducible string $\boldsymbol{x} \in \mathrm{Irr}(n)$, the irreducible-confusable set of $\boldsymbol{x}$ satisfies*

$$\left\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\right\| \leq \max_{\substack{\{\boldsymbol{x}_i\},\\ \{\boldsymbol{y}_j\}}} \prod_{i=0}^{p-1} \left\|R(D^{*,\leq 1}(\boldsymbol{x}_i))\right\| \prod_{j=1}^{p} \left\|R(D^{*,\leq 1}(\boldsymbol{y}_i))\right\|$$

where the maximums are over $\boldsymbol{x}_i \in R(D^{*,\leq i}(\boldsymbol{x}))$, $\boldsymbol{y}_j \in R(D^{*,\leq j}(\boldsymbol{y}))$ with $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$, and $\boldsymbol{x}_p = \boldsymbol{y}_p$.

Proof sketch: As discussed above, if $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$, then there exists $\boldsymbol{z}$ that can be obtained from both $\boldsymbol{x}$ and $\boldsymbol{y}$ via p concatenated DSD(1) channels, as shown in Fig. 1. Critically, the DSD(1) channel is reversible. That is, if $\boldsymbol{u}$ and $\boldsymbol{v}$ are irreducible sequences such that $\boldsymbol{u} \in R(D^{*,\leq 1}(\boldsymbol{v}))$, we also have $\boldsymbol{v} \in R(D^{*,\leq 1}(\boldsymbol{u}))$. The dashed lines in the figure represent the reverse channels. So the total number of possibilities for $\boldsymbol{y}$ can be bounded by considering the number of possibilities in each step from $\boldsymbol{x}$ to $\boldsymbol{y}$. The lemma then follows by induction on the number of possibilities for each string in the sequence $\boldsymbol{x} \to \boldsymbol{x}_1 \to \cdots \to \boldsymbol{z} \to \boldsymbol{y}_{p-1} \to \cdots \to \boldsymbol{y}$. ∎

Our next step in bounding $\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\|$ is to bound $\|R(D^{*,\leq 1}(\boldsymbol{u}))\|$ for an irreducible string $\boldsymbol{u}$. We need only consider $\|R(D^{*,1}(\boldsymbol{u}))\|$ as $\|R(D^{*,\leq 1}(\boldsymbol{u}))\| \leq \|R(D^{*,1}(\boldsymbol{u}))\| + 1$. The next lemma provides a bound on $\|R(D^{*,1}(\boldsymbol{u}))\|$, which depends on the length of $\boldsymbol{u}$. The proof relies on first reducing the problem to a special case in which $|\boldsymbol{u}| = 5$ and then leveraging the regular language representing $D^*(\boldsymbol{u})$.

Lemma 6. *Let $\boldsymbol{u} \in \mathrm{Irr}(n) \subseteq \Sigma_q^n$ be an irreducible string of length n with $q \geq 3$. Then*

$$\|R(D^{*,1}(\boldsymbol{u}))\| \leq 968nq.$$

Proof sketch: We first show that the effect of the substitution can be isolated to a substring of length 5, i.e.,

$$\|R(D^{*,1}(\boldsymbol{u}))\| \leq n \max_{abcde \in \Sigma_q^5} \|R(D^{*,1}(abcde))\|.$$

It can then be shown that the maximum on the right side can be obtained by assuming all symbols are distinct, i.e., replacing $abcde$ by 01234. We then use a finite automaton to represent the set $D^*(01234)$, where paths correspond to descendants of 01234. A substitution can be represented by changing the label of an edge. With certain modifications, the automaton can also

be used to represent $R(D^{*,1}(01234))$ and to bound its size, which is determined to be $968q$. ∎

According to Lemma 6, the upper bound of $\|R(D^{*,1}(\boldsymbol{u}))\|$ depends on the length n of $\boldsymbol{u}$. Therefore, we can obtain upper bounds on $\|R(D^{*,\leq 1}(\boldsymbol{x}_i))\|$ and $\|R(D^{*,\leq 1}(\boldsymbol{y}_j))\|$ in Lemma 5 if upper bounds on $|\boldsymbol{x}_i|$ and $|\boldsymbol{y}_j|$ are available. To obtain these bounds, we recall a theorem from [36].

Theorem 7. *[36, Theorem 5] Given strings $\boldsymbol{x} \in \Sigma_q^n$ and $\boldsymbol{v} \in D^{*,1}(\boldsymbol{x})$, $R(\boldsymbol{v})$ can be obtained from $R(\boldsymbol{x})$ by deleting a substring of length at most $\mathcal{L} = 17$ and inserting a substring of length at most $\mathcal{L}$ in the same position.*

In Lemma 5, we have $\boldsymbol{x}_i \in R(D^{*,\leq i}(\boldsymbol{x}))$ and $\boldsymbol{y}_j \in R(D^{*,\leq j}(\boldsymbol{y}))$ with $i, j \leq p$, implying that

$$|\boldsymbol{x}_i| \leq n + p\mathcal{L}, \quad |\boldsymbol{y}_j| \leq n + p\mathcal{L}. \tag{4}$$

The next theorem follows from Lemmas 5 and 6.

Theorem 8. *Let $\boldsymbol{x} \in \mathrm{Irr}(n) \subseteq \Sigma_q^n$ be an irreducible string of length n, with $q \geq 3$. Then*

$$\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\| \leq (968q(n + p\mathcal{L}) + 1)^{2p}.$$

IV. Error-correcting codes

As stated in Section III, our error-correcting code for correcting duplications and substitutions is a subset of irreducible strings of a given length. In this section, we construct this subset by applying the syndrome compression technique [32], where we will make use of the size of the irreducible-confusable set $\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\|$ derived in Section III. In this section, unless otherwise stated, we assume $q \geq 4$.

A. Code construction

We will start from a preliminary code, given in (5), and address its shortcomings, building up to the final code given in Construction 12.

Suppose p is constant with respect to n. Furthermore, suppose there exists a labeling function f and, for each $\boldsymbol{x} \in \mathrm{Irr}(n)$, an integer a such that for any $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$, $f(\boldsymbol{x}) \not\equiv f(\boldsymbol{y}) \bmod a$. Let $\boldsymbol{r}$ be a vector which encodes the information $(a, f(\boldsymbol{x}) \bmod a)$, to be precisely determined later. The set

$$\{\boldsymbol{x}\boldsymbol{r} : \boldsymbol{x} \in \mathrm{Irr}(n)\} \tag{5}$$

is a code capable of correcting duplications and at most p substitutions provided that, given the output $\boldsymbol{w} \in R(D^{*,\leq p}(\boldsymbol{x}\boldsymbol{r}))$, the following conditions are satisfied: 1) we can recover $(a, f(\boldsymbol{x}) \bmod a)$ and 2) we can find some $\boldsymbol{v} \in R(D^{*,\leq p}(\boldsymbol{x}))$. To see this, observe that if $\boldsymbol{y} \neq \boldsymbol{x}$ can also produce $\boldsymbol{v}$, then $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$, and hence it can be eliminated as an input candidate since $f(\boldsymbol{y}) \not\equiv f(\boldsymbol{x}) \bmod a$.

The first condition can be addressed by adapting the code given in [31, Construction 10], which has asymptotic rate $\geq \log(q-2)$. More precisely, a straightforward extension of [31, Theorem 11] leads to the following lemma.

Lemma 9. *Let $\boldsymbol{\sigma} = 01020$. There exists an encoder $\mathcal{E}_1 : \Sigma_2^m \to \mathrm{Irr}(m')$ such that i) $\boldsymbol{\sigma}\mathcal{E}_1(\boldsymbol{u}) \in \mathrm{Irr}(*)$ and ii) for any string $\boldsymbol{x} \in \mathrm{Irr}(*)$ with $\boldsymbol{x}\boldsymbol{\sigma}\mathcal{E}_1(\boldsymbol{u}) \in \mathrm{Irr}(*)$, we can recover $\boldsymbol{u}$ from any $\boldsymbol{w} \in R(D^{*,\leq p}(\boldsymbol{x}\boldsymbol{\sigma}\mathcal{E}_1(\boldsymbol{u})))$. Asymptotically, $m' \leq m/\log(q-2)(1 + o(1))$.*

We use $\mathcal{E}_1(a, f(\boldsymbol{x}) \bmod a)$ to denote $\mathcal{E}_1(\boldsymbol{u})$, where $\boldsymbol{u}$ is a binary sequence representing the pair $(a, f(\boldsymbol{x}) \bmod a)$. From the lemma, letting $\boldsymbol{r} = \boldsymbol{\sigma}\mathcal{E}_1(a, f(\boldsymbol{x}) \bmod a)$ will enable us to recover $(a, f(\boldsymbol{x}) \bmod a)$ from any $\boldsymbol{w} \in R(D^{*,\leq p}(\boldsymbol{x}\boldsymbol{r}))$ for any $\boldsymbol{x} \in \mathrm{Irr}(*)$ provided that $\boldsymbol{x}\boldsymbol{r} \in \mathrm{Irr}(*)$. We will discuss ensuring $\boldsymbol{x}\boldsymbol{r}$ is irreducible later.

The second condition requires us to find some $\boldsymbol{v}$ that is only a function of $\boldsymbol{x}$ rather than $\boldsymbol{x}\boldsymbol{r}$. This is more challenging as the boundary between $\boldsymbol{x}$ and $\boldsymbol{r}$ becomes unclear or may not even exist after duplication and substitution errors, making it difficult to find $\boldsymbol{v} \in R(D^{*,\leq p}(\boldsymbol{x}))$ from $\boldsymbol{w} \in R(D^{*,\leq p}(\boldsymbol{x}\boldsymbol{r}))$. To address this, instead of finding $\boldsymbol{v}$, we find some string $\boldsymbol{s}$ that is only a function of $\boldsymbol{x}$ as follows. Denote by $D^{*,\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$ the set of strings that can be obtained by deleting a suffix of length at most $2p\mathcal{L}$ from some $\boldsymbol{v} \in R(D^{*,\leq p}(\boldsymbol{x}))$.

Lemma 10. *Let $\boldsymbol{x}$ be an irreducible string of length n and $\boldsymbol{r}$ be any string such that $\boldsymbol{x}\boldsymbol{r}$ is irreducible. Let $\boldsymbol{w} \in R(D^{*,\leq p}(\boldsymbol{x}\boldsymbol{r}))$ and $\boldsymbol{s}$ be the prefix of $\boldsymbol{w}$ of length $n - p\mathcal{L}$. Then $\boldsymbol{s}$ can be obtained from some $\boldsymbol{v} \in R(D^{*,\leq p}(\boldsymbol{x}))$ by deleting a suffix of length at most $2p\mathcal{L}$. That is, $\boldsymbol{s} \in D^{*,\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$.*

Hence, by choosing the first $n - p\mathcal{L}$ elements of $\boldsymbol{w} \in R(D^{*,\leq p}(\boldsymbol{x}\boldsymbol{r}))$, we find $\boldsymbol{s} \in D^{*,\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$, which is a function of only $\boldsymbol{x}$ rather than $\boldsymbol{x}\boldsymbol{r}$. But in doing so, we have introduced an additional error, namely deleting a suffix of length at most $2p\mathcal{L}$. As a result, we need to replace the labeling function f with a stronger labeling function g_q that, in addition to handling both substitutions and duplications, can handle deleting a suffix of $\boldsymbol{x}$. More precisely, g_q is a labeling function for the confusable set

$$B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathrm{Irr}(n) : \boldsymbol{y} \neq \boldsymbol{x}, \\ D^{*,\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x}) \cap D^{*,\leq p,\leq 2p\mathcal{L}}(\boldsymbol{y}) \neq \varnothing\}. \tag{6}$$

The details of determining g_q will be discussed in Section IV-B.

The final piece of the construction is ensuring that the concatenation of $\boldsymbol{x}$ and $\boldsymbol{r}$ is irreducible. This can be done by adding a buffer $\boldsymbol{b}_{\boldsymbol{x}}$ between them.

Lemma 11. *For $q \geq 3$ and any irreducible string $\boldsymbol{x}$ over Σ_q, there is a string $\boldsymbol{b}_{\boldsymbol{x}}$ of length c_q such that $\boldsymbol{x}\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{\sigma}$ is irreducible. Furthermore, it suffices to choose $c_3 = 13$, $c_4 = 7$, and $c_q = 6$ for $q \geq 5$.*

The lemma implies $\boldsymbol{x}\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{r}$ is irreducible, because $\boldsymbol{r}$ starts with $\boldsymbol{\sigma}$, which has length 5, and because short repeats have length at most 6. So any repeat must be contained in $\boldsymbol{x}\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{\sigma}$ or in $\boldsymbol{r}$, which is not possible as they are both irreducible.

We are now ready to present the code construction and then a theorem that establishes its error-correcting capability. The proof of the theorem summarizes our preceding discussion.

Construction 12. *Let g_q be a labeling function for the confusable sets $B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x}), \boldsymbol{x} \in \mathrm{Irr}(n)$. Furthermore, for each*

$\boldsymbol{x}$, *let* a_1 *be an integer such that* $g_q(\boldsymbol{x}) \not\equiv g_q(\boldsymbol{y}) \bmod a_1$ *for* $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$. *Let*

$$\mathcal{C}_n = \{\boldsymbol{x}\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{r} : \boldsymbol{x} \in \mathrm{Irr}(n), \boldsymbol{r} = \boldsymbol{\sigma}\mathcal{E}_1(a_1, g_q(\boldsymbol{x}) \bmod a_1)\}.$$

Note that for simplicity, we index the code by the length of $\boldsymbol{x}$ rather than the length of the codewords $\boldsymbol{x}\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{r}$, i.e., n in $\mathcal{C}_n$ refers to the length of $\boldsymbol{x}$. The length of $\boldsymbol{r}$ is discussed in Subsection IV-C below.

Theorem 13. *The code in Construction 12 can correct any number of short duplications and at most* p *substitutions.*

Proof: Let the retrieved word be $\boldsymbol{w} \in R(D^{*,\leq p}(\boldsymbol{x}\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{r}))$. From Lemma 9, given $\boldsymbol{w}$, we can find a_1 and $g_q(\boldsymbol{x}) \bmod a_1$. Let $\boldsymbol{s}$ be the $(n - p\mathcal{L})$-prefix of $\boldsymbol{w}$. By Lemma 11, $\boldsymbol{x}\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{r}$ is irreducible. Then, by Lemma 10, the $(n - p\mathcal{L})$-prefix of $\boldsymbol{w}$, denoted $\boldsymbol{s}$, satisfies $\boldsymbol{s} \in D^{*,\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$. By definition, for all $\boldsymbol{y} \neq \boldsymbol{x}$ that could produce the same $\boldsymbol{s}$, we have $\boldsymbol{y} \in B^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$. But then, $g_q(\boldsymbol{y}) \not\equiv g_q(\boldsymbol{x}) \bmod a_1$, and so we can determine $\boldsymbol{x}$ by exhaustive search. ∎

B. The labeling function

In this subsection, we present the labeling function g_q such that $g_q(\boldsymbol{x}) \neq g_q(\boldsymbol{y})$ for $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$. As shown in Theorem 7, each substitution is reflected in the root of the sequence as a substring edit of length at most $\mathcal{L}$. Considering also the suffix deletion of length at most $2p\mathcal{L}$, it follows that $\boldsymbol{s} \in B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$ can be obtained from both $\boldsymbol{x}$ and $\boldsymbol{y}$ by at most $3p\mathcal{L}$ deletions and at most $p\mathcal{L}$ insertions. Hence, it suffices to find g_q such that $g_q(\boldsymbol{x}) \neq g_q(\boldsymbol{y})$ if there is a string $\boldsymbol{s}$ that can be obtained from both $\boldsymbol{x}$ and $\boldsymbol{y}$ through $4p\mathcal{L}$ indels. Note that since we are utilizing syndrome compression, choosing a more "powerful" labeling function does not increase the redundancy, which is still primarily controlled by $\max_{\boldsymbol{x}\in\mathrm{Irr}(n)} \|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\|$. To find g_q, we use the labeling function for binary sequences given in [35], whose properties are discussed in the following theorem.

Theorem 14. *[35] There exists a labeling function* $g : \{0,1\}^n \to \Sigma_{2^{\mathcal{R}(t,n)}}$ *such that for any two distinct strings* $\boldsymbol{c}_1$ *and* $\boldsymbol{c}_2$ *confusable under* t *insertions, deletions, and substitutions, we have* $g(\boldsymbol{c}_1) \neq g(\boldsymbol{c}_2)$, *where* $\mathcal{R}(t,n) = [(t^2+1)(2t^2+1) + 2t^2(t-1)]\log n + o(\log n)$.

Since $\boldsymbol{s} \in D^{*,\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$ can be obtained from $\boldsymbol{x}$ via $4p\mathcal{L}$ indels, $\mathcal{U}_i(\boldsymbol{s})$ can be derived from $\mathcal{U}_i(\boldsymbol{x})$ by at most $4p\mathcal{L}$ indels for $i \in [\lceil \log q \rceil]$. Based on Theorem 14, by letting $t = 4p\mathcal{L}$, we can obtain a labeling function g for recovering $\mathcal{U}_i(\boldsymbol{x})$ from $\mathcal{U}_i(\boldsymbol{s})$ under at most $4p\mathcal{L}$ indels. Therefore, $g_q : \Sigma_q^n \to \Sigma_{2^{\lceil \log q \rceil \mathcal{R}(t,n)}}$,

$$g_q(\boldsymbol{x}) = \sum_{i=1}^{\lceil \log q \rceil} 2^{\mathcal{R}(t,n)(i-1)} g(\mathcal{U}_i(\boldsymbol{x})), \tag{7}$$

where $t = 4p\mathcal{L}$, is a labeling function for the confusable sets $B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$, $\boldsymbol{x} \in \mathrm{Irr}(n)$. For each $\boldsymbol{x}$, a value a_1 needs to be also determined such that $g_q(\boldsymbol{x}) \not\equiv g_q(\boldsymbol{y}) \bmod a_1$ for $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$. The existence of such a_1, satisfying $\log a_1 \leq \log \|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| + o(\log n)$, is guaranteed by Theorem 3 provided that p is a constant (ensuring that $p^4 = o(\log\log n)$).

C. The redundancy of the error-correcting codes

In this section, we study the rate and the redundancy of the code proposed in Construction 12 and compare these to those of the short-duplication-correcting code given in [8], which has the highest known asymptotic rate. A simplified version of the construction of [8] is given below.

Construction 15. *(c.f. [8]) For a positive integer* n, *let*

$$\mathcal{C}_n^d = \{\boldsymbol{x} \in \Sigma_q^n : \boldsymbol{x} \in \mathrm{Irr}(n)\}.$$

Given $q \geq 4$, this code has the same asymptotic rate as the original in [8] and its size differs by only a constant factor. We thus compare the proposed code with $\mathcal{C}_n^d$.

The codes $\mathcal{C}_n$ and $\mathcal{C}_n^d$ have the same size but the length of $\mathcal{C}_n$ is larger by $|\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{r}|$ symbols. We have seen in Lemma 11 that $|\boldsymbol{b}_{\boldsymbol{x}}| \leq 13$. The extra redundancy is then $|\boldsymbol{r}| + O(1)$, which depends on $\|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\|$, investigated in the next lemma.

Lemma 16. *For* $\boldsymbol{x} \in \mathrm{Irr}(n)$ *over the alphabet* Σ_q *with* $q \geq 3$,

$$\|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| \leq q^{4p\mathcal{L}}(n + p\mathcal{L})^{2p}.$$

The proof relies on Theorem 8 and takes into account the effect of the suffix deletion.

Lemma 17. *Suppose both* p *and* $q \geq 4$ *are constant with respect to* n. *Given* $\boldsymbol{x} \in \mathrm{Irr}(n) \subseteq \Sigma_q^n$, *the length* L *of* $\boldsymbol{r} = \boldsymbol{\sigma}\mathcal{E}_1(a_1, g_q(\boldsymbol{x}) \bmod a_1)$ *satisfies* $L \leq 8p(\log_q n)(1 + o(1))$.

Proof: From the previous subsection, assuming p is a constant, we have that $\log a_1 \leq \log \|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| + o(\log n)$. Since $(g_q(\boldsymbol{x}) \bmod a_1) \leq a_1$, we need $L_r = 2\log \|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| + o(\log n)$ bits to represent the pair $(a_1, g_q(\boldsymbol{x}) \bmod a_1)$. By Lemma 16, assuming q is a constant, we have $L_r \leq 4p\log n + o(\log n) = 4p\log n(1 + o(1))$. Then, by Lemma 9, $|\mathcal{E}_1((a_1, g_q(\boldsymbol{x}) \bmod a_1))| \leq 4p\log n(1 + o(1))/\log(q-2)$. The lemma follows from the facts that $\frac{\log q}{\log(q-2)} \leq 2$ for $q \geq 4$ and that $|\boldsymbol{\sigma}| = 5$. ∎

Using Lemma 17, the next theorem gives the extra redundancy of correcting p additional substitutions compared to [8]. It also shows that there is no asymptotic rate penalty, in contrast to prior work [31], which also corrects duplications and p substitutions.

Theorem 18. *Assuming* p *and* q *are constants, compared to* $\mathcal{C}_n^d$, *the proposed code* $\mathcal{C}_n$ *has the same size but is longer by* $8p\log_q n(1 + o(1))$ *symbols. The codes have the same asymptotic rate, which equals* $\log 2.6590$ *for* $q = 4$.

D. The time complexities of encoding and decoding

The encoding process relies on determining a value for a_1 satisfying the condition discussed in Subsection IV-B among at most $2^{\log \|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| + o(\log n)}$ possibilities. This step has complexity $O(n^{4p+1})$ in n, making the total complexity of encoding polynomial in n. Decoding requires deduplication, which is linear in the length of the retrieved sequence, and a brute-force search among all inputs that can lead to the same output $(n - p\mathcal{L})$-prefix of the root of the retrieved sequence, which is polynomial in n. Hence, decoding is polynomial in the length of the retrieved sequence.

REFERENCES

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

[2] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Noise and uncertainty in string-duplication systems," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 3120–3124.

[3] S. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Scientific reports*, vol. 5, no. 1, pp. 1–10, 2015.

[4] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, 2017.

[5] M. Blawat, K. Gaedke, I. Huetter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," *Procedia Computer Science*, vol. 80, pp. 1011–1022, 2016.

[6] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen *et al.*, "Random access in large-scale DNA data storage," *Nature biotechnology*, vol. 36, no. 3, pp. 242–248, 2018.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[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, "Correcting deletion errors in DNA data storage with enzymatic synthesis," in *2021 IEEE Information Theory Workshop (ITW)*, 2021, pp. 1–6.

[25] ——, "Error-correcting codes for noisy duplication channels," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3452–3463, 2021.

[26] 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.

[27] M. Kovačević, "On the maximum number of non-confusable strings evolving under short tandem duplications," *Problems of Information Transmission*, vol. 58, no. 2, pp. 111–121, 2022.

[28] 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.

[29] 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.

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

[31] Y. Tang, H. Lou, and F. Farnoud, "Error-correcting codes for short tandem duplications and at most p substitutions," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 1835–1840.

[32] J. Sima, R. Gabrys, and J. Bruck, "Syndrome compression for optimal redundancy codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 751–756.

[33] J. Sima and J. Bruck, "On optimal k-deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3360–3375, 2020.

[34] J. Sima, R. Gabrys, and J. Bruck, "Optimal codes for the q-ary deletion channel," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 740–745.

[35] ——, "Optimal systematic t-deletion correcting codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 769–774.

[36] Y. Tang and F. Farnoud, "Error-correcting codes for short tandem duplication and edit errors," *IEEE Transactions on Information Theory*, vol. 68, no. 2, pp. 871–880, 2022.