Correcting deletion errors in DNA data storage with enzymatic synthesis

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

***Abstract*—DNA is considered a promising alternative to traditional storage media because of advantages such as high data density, longevity, and ease of generating copies. One of the major drawbacks of DNA data storage however is that DNA synthesis is costly and resource intensive. A newly proposed enzymatic method has the potential to decrease the cost of synthesis but has the disadvantage that the number of times a base is repeated cannot be precisely controlled. The method is also prone to deletion of runs. Existing encoding approaches for this synthesis method either have a low rate, specifically, $\leq \log_2 3$ per run, or cannot protect against deletion errors. The current paper proposes a new error-correcting code and a synchronization algorithm that can combat deletions and achieve a code rate higher than $\log_2 3$ bits per unit time.**

I. Introduction

Due to its high density, longevity, energy-efficiency, and ease of generating copies [1], [2], DNA is considered a promising candidate for data storage. For example, the amount of DNA in a single human cell can store as much as 6.4 Gbs of information, and the DNA of species extinct for thousands of years can still be successfully decoded [3]. Recent works [3]–[8] have demonstrated the feasibility of DNA data storage.

However, due to high cost [4], [9] and chemistry limitations [10], [11], DNA sequences produced by common synthesis methods have limited quantity and quality [1]. Recently, the authors in [1] proposed a new inexpensive enzymatic method to synthesize DNA sequences. Compared to conventional methods with single-base accuracy, in each synthesis round of the enzymatic method, a number of nucleotides of the same type are appended to the sequence. The number of nucleotides added is a random number with a distribution that is affected by multiple factors, including previous bases and the *synthesis duration* of the current round [1]. After n synthesis rounds, this process produces N sequences, each with at most n runs, where the length of each run is noisy. While the synthesized DNA sequences by the enzymatic method also suffer deletions, insertions, and substitutions [1], the focus of this paper is on combating errors arising from the noisiness of the run lengths. To combat those errors, the authors in [1] encoded information in *transitions* between adjacent non-identical bases. Since the run lengths are not considered, the rate of the code is upper bounded by $\log_2 3$ bits per run [1] [1], [12].

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

[1]We define the unit of time as the amount of time that it takes to produce a run in this method, so the rate is equivalent to $\log_2 3$ bits per unit time.

To make full use of run length distributions of nucleotides, the authors in [12] constructed codes using the *precision-resolution* (PR) framework [13]. In this method, a limited set of synthesis times are allowed and based on the observed run lengths, it is determined which synthesis time was used for each run. This approach can increase information density per unit time compared to the method in [1], in which all runs are synthesized with the same synthesis time. This work also takes advantage of the fact that enzymatic synthesis can produce N sequences in parallel (the run lengths may be different among these sequences). It is shown that with this method rates larger than $\log_2 3$ bits per unit time can be achieved. The main limitation of this work is that it assumes runs cannot be deleted, which contradicts the fact that deletions are the dominant type of error in enzymatic synthesis.

The channel resulting from enzymatic synthesis with noisy run lengths can also be viewed as an insertion-deletion channel with multiple traces. Such a view would benefit from the extensive literature on the topic, including [14]–[19]. However, this approach would not be able to take advantage of the specific structure of the deletions and insertions. Namely, each deletion or insertion event modifies a single run and follows specific distributions given the synthesis time for the corresponding round. As each run may be altered by inserting or deleting a large number of symbols, the resulting insertion-deletion probability would be very large. Burst insertion-deletion channels would be more appropriate, but these also ignore the fact that each event is limited to a single run.

To *correct errors arising from noisy run lengths*, similar to [12], the current paper also uses the PR framework but also proposes error-correcting codes to fight against deletions in the DNA sequences as well as achieve a code rate higher than $\log_2 3$ bits per unit time. The code construction contains two codes. Based on [20], we introduce the *block-based Tenegolts q-ary* (BTq) code by concatenating Tenegolts q-ary codes. This code is used to protect against deletions of runs. Then, after synchronizing the runs in sequences synthesized in parallel, the second *substitution-correcting* (SC) code decodes the information stored in the run lengths [12]. Based on numerical results, in some settings, the constructed code can decode codewords with small error probability while achieving a code rate higher than $\log_2 3$ bits per unit time.

The rest of this paper is organized as follows. Section II provides the notation and preliminaries. Section III introduces the channel model. The code construction as well as its rate are presented in Section IV, followed by decoding procedures in Section V. Finally, Section VI presents the numerical results.

II. Notation and Preliminaries

Let Σ_q denote the alphabet $\{0, 1, \ldots, q-1\}$. In particular, let $\Sigma_4 = \{0, 1, 2, 3\}$ represent the alphabet corresponding to $\{A, C, G, T\}$ in DNA sequences. Let Σ_q^n denote the set of all strings of length n over Σ_q, and $\Sigma_q^* = \bigcup_{n=0}^{+\infty} \Sigma_q^n$ represent the set of all strings of finite length, including the empty string Λ. Furthermore, $\Sigma_q^+ = \Sigma_q^* \backslash \{\Lambda\}$. Let $[n]$ denote the set $\{1, \ldots, n\}$ and $\mathbb{N}$ and $\mathbb{N}_+$ denote the sets of nonnegative and positive integers, respectively.

We use bold symbols to denote strings over Σ_q, such as $\boldsymbol{x}$ and $\boldsymbol{y}_j$. The elements of strings are shown with plain typeface, e.g., $\boldsymbol{x} = x_1 x_2 \cdots x_n$, where $x_i \in \Sigma_q$. Given two strings $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, $\boldsymbol{xy}$ denotes their concatenation. For a string $\boldsymbol{x} \in \Sigma_q^*$, let $|\boldsymbol{x}|$ denote its length. Given 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 $\boldsymbol{x} = \boldsymbol{uvw}$. Let $\boldsymbol{x} = x_1 x_2 \cdots x_n \in \Sigma_q^n$ be an *alternating string* of length n over Σ_q if $x_i \neq x_{i+1}$ for $i \in [n-1]$. Let S_q^n be the set of alternating strings of length n. Therefore, $S_q^n \subseteq \Sigma_q^n$.

A *run* is a maximal substring consisting of a single symbol. For $a \in \Sigma_q, r \in \mathbb{N}$, let $a^r = a \cdots a$ represent a repeated r times. For $\boldsymbol{a} \in \Sigma^n$ and $\boldsymbol{r} \in \mathbb{N}^n$, let $\boldsymbol{x} = \boldsymbol{a}^{\boldsymbol{r}} = a_1^{r_1} a_2^{r_2} \cdots a_n^{r_n}$. If $r_i > 0$ and a_i is different from a_{i-1} and a_{i+1}, then ${a_i}^{r_i}$ is a run in $\boldsymbol{x}$ and r_i is the *run length* of a_i. Note that if $\boldsymbol{a} \in S_q^n$ is alternating and $\boldsymbol{r} \in \mathbb{N}_+^n$ is positive, then $\boldsymbol{x} = \boldsymbol{a}^{\boldsymbol{r}}$ has n runs. We define Ω_q^n as the set consisting of all strings with n runs.

Conversely, given a string $\boldsymbol{x} \in \Omega_q^n$, we can decompose it into its alternating string $\boldsymbol{a} \in S_q^n$ and run length sequence $\boldsymbol{r} \in \mathbb{N}_+^n$ such that $\boldsymbol{a}^{\boldsymbol{r}} = \boldsymbol{x}$. Formally, we define a mapping $\boldsymbol{\psi} : \Omega_q^n \to S_q^n \times \mathbb{N}_+^n$ as

$$\boldsymbol{\psi}(\boldsymbol{x}) = (\boldsymbol{a}, \boldsymbol{r}) = (\boldsymbol{\psi}_1(\boldsymbol{x}), \boldsymbol{\psi}_2(\boldsymbol{x})). \tag{1}$$

For example, given a string $\boldsymbol{x} = 11144331 = 1^3 4^2 3^2 1$, we have $\boldsymbol{a} = \boldsymbol{\psi}_1(\boldsymbol{x}) = 1431$ and $\boldsymbol{r} = \boldsymbol{\psi}_2(\boldsymbol{x}) = 3221$.

Based on [12], given a finite directed and labeled graph $\mathcal{G} = (V, E, \mathcal{L})$ with the set of vertices V, a finite multiset of edges (allowing parallel edges) $E \subseteq V \times V$, and the labels on the edges $\mathcal{L} : E \to \Sigma^+$, the length of an edge $e \in E$ is $l(e) = |\mathcal{L}(e)|$. If all edges have length 1, the graph $\mathcal{G}$ is *ordinary*. Furthermore, a single path $\boldsymbol{s} = e_1 e_2 \cdots e_m$ in $\mathcal{G}$ generates the word $\mathcal{L}(\boldsymbol{s}) = \mathcal{L}(e_1)\mathcal{L}(e_2)\cdots\mathcal{L}(e_m)$, where $e_i \in E$. Given two vertices $v_1, v_2 \in V$ and a string $\boldsymbol{x} \in \Sigma^*$, the graph $\mathcal{G}$ is called *lossless* if we can find at most one path $\boldsymbol{s}$ such that $\mathcal{L}(\boldsymbol{s}) = \boldsymbol{x}$. Based on the graph $\mathcal{G}$, we define a constrained system consisting of all words generated by finite paths in $\mathcal{G}$, $S(\mathcal{G}) = \{\mathcal{L}(\boldsymbol{r}) | \boldsymbol{r}$ is a finite path in $\mathcal{G}\}$. If $\mathcal{G}$ is not ordinary, an equivalent ordinary graph $\mathcal{G}'$ can be constructed by converting each edge $v \xrightarrow{w_1 \ldots w_l} u$ into a path $v \xrightarrow{w_1} v_1 \xrightarrow{w_2} \cdots \xrightarrow{w_{l-1}} v_{l-1} \xrightarrow{w_l} u$ by inserting auxiliary vertices $v_1, \ldots, v_{l-1}$, where $w_i \in \Sigma$.

We next recall the construction of Tenegolts q-ary codes [21] that are capable of correcting a single insertion or deletion (indel) over Σ_q.

Construction 1. Based on Tenegolts q-ary (Tq) code [21], given integers $m \geq 1$, $0 \leq \alpha \leq (q-1)$ and $0 \leq \beta \leq (m-1)$, we construct the code $C_{Tq}(\alpha, \beta, m)$ over Σ_q as

$$C_{Tq}(\alpha,\beta,m) = \Bigg\{ \boldsymbol{z} \in \Sigma_q^m \Bigg| \sum_{j=1}^{m} z_j = \alpha \bmod q, \quad \sum_{i=1}^{m} (i-1)\zeta_i(\boldsymbol{z}) = \beta \bmod m \Bigg\}, \tag{2}$$

where the i-th symbol of $\boldsymbol{\zeta} : \Sigma_q^m \to \Sigma_2^m$ is

$$\zeta_i(\boldsymbol{z}) = \begin{cases} 1, & \text{if } z_i \geq z_{i-1}, \\ 0, & \text{if } z_i < z_{i-1}, \end{cases} \qquad i = 2, 3, \ldots, m,$$

with $\zeta_1(\boldsymbol{z}) = 1$.

III. Channel model for enzymatic DNA synthesis

In this section, we describe the channel model for the enzymatic DNA synthesis in the precision-resolution (PR) framework with deletions of runs.

In enzymatic synthesis N DNA sequences are synthesized in parallel. The synthesis process consists of n rounds, and in each round, a number of nucleotides of the same type, e.g., A, are added to the sequences. Nucleotides added in rounds i and $i+1$ are of different types, so each round adds a run to each sequence. The lengths of the runs are controlled by the duration of the round. We assume that the number of bases synthesized is a random variable r with distribution[2] $D(t)$, independent of all other events, where t is the associated time spent on the synthesis of the run. Given the noisiness of the run lengths, following the precision-resolution framework of [12], we choose t from a set of discrete times $T_{syn} = \{t_0, t_1, \ldots, t_{L-1}\}$. In this method, N sequences are synthesized simultaneously, all of whom have the same alternating sequence (unless a run is deleted due to no bases being added to the sequence) but the run lengths may differ due to synthesis noise.

We thus model the enzymatic channel in the PR framework as follows: Given two sequences $\boldsymbol{a} \in S_q^n$, $\boldsymbol{b} \in \Sigma_L^n$, and the set T_{syn}, enzymatic DNA synthesis creates N traces $\boldsymbol{x}_j = \boldsymbol{a}^{\boldsymbol{r}_j}$, where the run length r_{ji} for the ith run in the jth trace is determined as $r_{ji} \sim D(t_{b_i})$ with $t_{b_i} \in T_{syn}$.

At the decoder, we determine $\boldsymbol{a}$ and $\boldsymbol{b}$ from $(\boldsymbol{x}_j)_{j=1}^N$, by first finding $\bar{\boldsymbol{a}}_j = \boldsymbol{\psi}_1(\boldsymbol{x}_j)$ and $\bar{\boldsymbol{r}}_j = \boldsymbol{\psi}_2(\boldsymbol{x}_j)$.

There are two sources of error in this channel. For the moment, assume that we are given $(r_{ji})_{i=1}^N$, from which we must find the value of t_{b_i} and thus b_i. Given that r_{ji} are random quantities, we may decode b_i incorrectly, leading to substitution errors in the codewords $\boldsymbol{b}$. This is the first source of error. The larger the gap between t_k and t_{k+1}, the smaller the probability of this type of error. The second source of error arises from the fact that if $r = 0$, the corresponding run disappears from the trace (this occurs if during synthesis, no base is added to the sequence), leading to deletions of runs. This causes synchronization issues, which may prevent us from determining all of the values $(r_{ji})_{j=1}^N$. The following example demonstrates the two possible ways deletion of runs can affect the alternating and run length sequences.

[2]In general this distribution may depend on the previous symbol but for simplicity we assume the same distribution regardless of the previous base.

Example 1. Suppose $N = 2$ and let $\boldsymbol{a} = CGAGT$. Suppose $\boldsymbol{r}_1 = (3, 0, 3, 2, 2)$ and $\boldsymbol{r}_2 = (3, 1, 0, 3, 2)$. We thus have:

1) $\boldsymbol{x}_1 = C^3G^0A^3G^2T^2 = C^3A^3G^2T^2$. Then $\bar{\boldsymbol{a}}_1 = CAGT$ and $\bar{\boldsymbol{r}}_1 = (3, 3, 2, 2)$. Therefore, $r_{1,2} = 0$ leads to a deletion in $\bar{\boldsymbol{a}}_1$ compared to $\boldsymbol{a}$ and a deletion in the run lengths $\bar{\boldsymbol{r}}_1$ compared to $\boldsymbol{r}_1$.
2) $\boldsymbol{x}_2 = C^3G^1A^0G^3T^2 = C^3G^4T^2$. Then $\boldsymbol{a}_2 = CGT$ and $\bar{\boldsymbol{r}}_2 = (3, 4, 2)$. Therefore, $r_{2,3} = 0$ leads to two deletions in $\boldsymbol{a}_2$ compared to $\boldsymbol{a}$ and two deletions and a substitution in $\bar{\boldsymbol{r}}_2$ compared to $\boldsymbol{r}_2$. This occurs because the two runs of G are *merged*. △

IV. Encoding and Code Rate

Our overall error-correction strategy is to protect $\boldsymbol{a}$ against deletions through coded trace reconstruction. After decoding $\boldsymbol{a}$ from the traces $(\bar{\boldsymbol{a}}_j)_{j=1}^N$, we align the traces with $\boldsymbol{a}$ with the goal to identify $(r_{ji})_{j=1}^N$ for each i. From $(r_{ji})_{j=1}^N$, we estimate b_i. We use a substitution-correcting code over $\boldsymbol{b}$ to protect against errors that may arise in this phase.

A. Code for correcting deletions in the alternating sequence

To correct deletions in the alternating sequence $\boldsymbol{a}$, we use a trace reconstruction method based on the binary construction proposed in [20]. In this construction, each codeword is divided into blocks, each of which can correct a single deletion via the VT code. The construction can be easily extended to the q-ary alphabet by using Tenegolts q-ary single-deletion-correcting code, described in Construction 1. An additional constraint to ensure the codewords are alternating sequences is also needed.

Construction 2. Given the alphabet Σ_q, code length n, and block length m that divides n, the Block-based Tenegolts q-ary code (BTq code) is

$$C_{BTq}(\boldsymbol{\alpha}, \boldsymbol{\beta}, n) = \Big\{ \boldsymbol{a} \in S_q^n \Big| \boldsymbol{a} = \boldsymbol{z}_1\boldsymbol{z}_2\cdots\boldsymbol{z}_b, \boldsymbol{z}_i \in C_{Tq}(\alpha_i, \beta_i, m) \cap S_q^m, i \in [b] \Big\}, \tag{3}$$

where $b = n/m$, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_b) \in \Sigma_q^b$, and $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_b) \in \Sigma_m^b$.

Lemma 1. *There exist choices for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ such that*

$$|\mathcal{C}_{BTq}(\boldsymbol{\alpha}, \boldsymbol{\beta}, n)| \geq (q-1)^n/(q^b m^b). \tag{4}$$

Given n, m, to protect an alternating sequence with length k over Σ_q, there is a BTq code with an encoder $\mathcal{E}_{BTq} : S_q^k \to S_q^n$ such that $n = k/C_m$ asymptotically, where

$$C_m = 1 - \frac{1}{m}\log_{q-1}(qm). \tag{5}$$

B. Code for correcting substitutions

As discussed earlier, the length of each run is controlled by the duration of each synthesis round. Following the precision-resolution framework [12], the synthesis time in each round is an element of the set $T_{syn} = \{t_0, t_1, \ldots, t_{L-1}\}$, with $t_b \in \mathbb{N}_+$ and $b \in \Sigma_L$, where $1 \leq t_0 < t_1 < \ldots < t_{L-1} \leq M$. Note that a larger L can increase the information per run, but also increases the probability that the synthesis time is not decoded correctly. Given the set of run lengths associated with synthesis round i, a quantizer function, discussed later, will be used to determine the index b_i of the synthesis time t_{b_i} in round i in the set $T_{syn} = \{t_0, \ldots, t_{L-1}\}$. Given the noisinsess of the run lengths, substitution errors are possible in $\boldsymbol{b}$. We assume the substitution probability is bounded by $\delta/2$. Given δ, a systematic substitution-correcting code (SC code) over Σ_L exists for large n with asymptotic redundancy $n(1 - C_{\delta,L})$, where

$$C_{\delta,L} = 1 + \delta\log_L\frac{\delta}{L-1} + (1-\delta)\log_L(1-\delta), \tag{6}$$

that can correct $n\delta/2$ substitution errors [22]. We refer to this code as $\mathcal{C}_{sc}$ and to its systematic encoder as $\mathcal{E}_{sc} : \Sigma_L^s \to \Sigma_L^n$, where $n = s/C_{\delta,L}$ asymptotically.

C. Combined codes

Given the codes discussed in the previous two subsections, we introduce the overall code for correcting both deletions and errors arising in the PR framework.

Construction 3. The BTq-SC code is defined as

$$\mathcal{C} = \{(\boldsymbol{a}, \boldsymbol{b}) : \boldsymbol{a} \in \mathcal{C}_{BTq}, \boldsymbol{b} \in \mathcal{C}_{sc}\}. \tag{7}$$

Observe that $\boldsymbol{a} \in S_q^n$ and $\boldsymbol{b} \in \Sigma_L^n$.

Let $\mathcal{G}_q = (V, E, \mathcal{L})$ be the graph with $V = \Sigma_q$. For every two distinct vertices $u, v \in V$, L directed parallel edges with labels $v^{t_b}, b \in \Sigma_L$ are added from u to v, where $t_b \in T_{syn}$. Based on [12, Lemma 1], the graph $\mathcal{G}_q$ is lossless and the capacity of $S(\mathcal{G}_q)$ is

$$\mathrm{Cap}(S(\mathcal{G}_q)) = \log_2\lambda(A_{\mathcal{G}_q'}), \tag{8}$$

where $\mathcal{G}_q'$ is the equivalent ordinary graph of $\mathcal{G}_q$, $A_{\mathcal{G}_q'}$ is the adjacency matrix of $\mathcal{G}_q'$, and $\lambda(A_{\mathcal{G}_q'})$ is the largest eigenvalue of $A_{\mathcal{G}_q'}$. The code rate is given in the next theorem, which in contrast to [12], also takes into account the deletion-correcting code.

Theorem 1. *Given $0 < \delta \leq 1 - 1/L$, T_{syn}, m, and M, the asymptotic rate of the code $\mathcal{C}$ of Construction 3* per unit time *satisfies*

$$R(\mathcal{C}) \geq \mathrm{Cap}(S(\mathcal{G}_q))/(1 + M\alpha(1/C_{lb} - 1)), \tag{9}$$

where α is the sum of the probabilities of non-auxiliary vertices in the stationary distribution of the max-entropic Markov chain (MEMC) over $\mathcal{G}_q'$ [23] and $C_{lb} = \min(C_{\delta,L}, C_m)$.

Proof: Based on Construction 3, given the alternating sequence $\boldsymbol{a} \in S_q^n$ and the synthesis times in n rounds $(t_{b_1}, \cdots, t_{b_n}) \in T_{syn}^n$, every string $a_1^{t_{b_1}}\cdots a_n^{t_{b_n}}$ generated from a codeword $(\boldsymbol{a}, \boldsymbol{b})$ can be represented by a path in $\mathcal{G}_q$, belonging to the constrained system $S(\mathcal{G}_q)$. Furthermore, the capacity of the constrained system $S(\mathcal{G}_q)$ is $\mathrm{Cap}(S(\mathcal{G}_q))$.

By the state-splitting algorithm, we can build an encoder $\mathcal{E}$ for $S(\mathcal{G}_q)$ with the code rate approaching $\mathrm{Cap}(S(\mathcal{G}_q))$ [12]. Given k message bits, the encoder $\mathcal{E}$ generates a sequence in

$S(\mathcal{G}_q)$ with $\frac{k}{\mathrm{Cap}(S(\mathcal{G}_q))}$ synthesis time units and $\frac{k}{\mathrm{Cap}(S(\mathcal{G}_q))}\alpha$ runs asymptotically [12, Theorem 3]. Then the encoded information contains an alternating sequence and a synthesis time sequence both with length $\frac{k}{\mathrm{Cap}(S(\mathcal{G}_q))}\alpha$. By using the encoders $\mathcal{E}_{BTq}$ and $\mathcal{E}_{sc}$, the sequence with $\frac{k}{\mathrm{Cap}(S(\mathcal{G}_q))}\alpha$ runs, generated by the encoder $\mathcal{E}$, can be encoded as a codeword with at most $\frac{k}{\mathrm{Cap}(S(\mathcal{G}_q))}\alpha\frac{1}{C_{lb}}$ runs. Then the error-correcting code $\mathcal{C}$ introduces at most $\frac{k}{\mathrm{Cap}(S(\mathcal{G}_q))}\alpha(\frac{1}{C_{lb}}-1)M$ time units for parity symbols. Since $\mathcal{C}_{sc}$ is a systematic code, the total synthesis time is at most $\frac{k}{\mathrm{Cap}(S(\mathcal{G}_q))}(1+M\alpha(\frac{1}{C_{lb}}-1))$ time units. Therefore, the code rate satisfies $R(\mathcal{C}) \geq \mathrm{Cap}(S(\mathcal{G}_q))/(1+M\alpha(\frac{1}{C_{lb}}-1))$ bits per unit time. ∎

Figure 1 shows the lower bound of $R(\mathcal{C})$ given in Theorem 1. The parameters are discussed in detail in Section VI-A. It can be observed that rates larger than $\log_2(3)$ bits per unit time are achievable.

V. Decoding

In this section, we describe the decoding procedure. Our goal is to decode $(\boldsymbol{a}, \boldsymbol{b})$ from $\{\bar{\boldsymbol{a}}_j\}_1^N$ and $\{\bar{\boldsymbol{r}}_j\}_1^N$. First, $\boldsymbol{a}$ will be decoded and used to identify $(r_{ji})_{j=1}^N$ for each i through alignment. As we will see, there is no guarantee that $(r_{ji})_{j=1}^N$ can be fully determined. A quantizer is used to determine b_i based on the available values of $(r_{ji})_{j=1}^N$.

A. Decoding the alternating sequence $\boldsymbol{a}$

We now describe the decoding procedure for the alternating sequence $\boldsymbol{a}$ from $\{\bar{\boldsymbol{a}}_j\}_{j=1}^N$. Given that the code $\mathcal{C}_{BTq}$ is an extension of the binary coded trace reconstruction method from [20], the decoding method is similar, and is described briefly here. Recall that each codeword consists of blocks capable of correcting a single deletion. In the first phase, the aim is to identify blocks that are deletion-free in at least one trace and to correct these blocks in all traces using their error-free copies. Assuming the first phase has succeeded, the remaining blocks have errors in all their copies. In the second phase, a decoder for Tenegolts code is used to correct deletion errors in these blocks to the extent possible.

B. Decoding the run length sequence $\boldsymbol{b}$

In this subsection, we assume that $\boldsymbol{a}$ has been decoded correctly. If this is not the case, a decoding failure (possibly undetected) has occurred.

For each synthesis round i, if $(r_{1i}, \ldots, r_{Ni})$ were known, we could decode b_i using the fact that $r_{ji} \sim D(t_{b_i})$. However, our decoding is based on $\bar{\boldsymbol{a}}_j$ and $\bar{\boldsymbol{r}}_j$ and as Example 1 shows due to merging some values in $\boldsymbol{r}_j$ are summed and appear as a single value in $\bar{\boldsymbol{r}}_j$. So to decode b_i, we first find a subset R_i of the samples $\{r_{1i}, \ldots, r_{Ni}\}$ that are known to be generated in the ith round (are not the sum of two or more elements of $\boldsymbol{r}_j$). Given this set, then a maximum-likelihood (ML) or Maximum a posteriori (MAP) quantizer can be used to decode b_i. In the following, we first show how to find R_i, then discuss the quantizer in detail.

The elements of the set R_i are determined by aligning $\bar{\boldsymbol{a}}_j$ with $\boldsymbol{a}$. We consider two specific alignments, the *forward alignment* and the *backward alignment*. In the forward alignment, starting from the left, each element α of $\bar{\boldsymbol{a}}_j$ is aligned with the leftmost non-aligned instance of α in $\boldsymbol{a}$. The backward alignment is similar but it starts from the right side and each symbol α in $\bar{\boldsymbol{a}}_j$ is mapped with the rightmost non-aligned instance of α in $\boldsymbol{a}$. Let f_{jk} be the position in $\boldsymbol{a}$ to which the kth element $\bar{r}_{jk}$ of $\bar{\boldsymbol{r}}_j$ is aligned to in the forward alignment, and let g_{jk} represent the corresponding position in the backward alignment.

Example 2. Continuing Example 1, the forward and backward alignments of $\bar{\boldsymbol{a}}_2 = CGT$ with $\boldsymbol{a} = CGAGT$ are given as

$$\begin{array}{cccc} \boldsymbol{a} & = & CGAGT \\ \bar{\boldsymbol{a}}_2 & = & CG\quad T \end{array}, \qquad \begin{array}{cccc} \boldsymbol{a} & = & CGAGT \\ \bar{\boldsymbol{a}}_2 & = & C\quad GT \end{array}$$

We have $f_{21} = g_{21} = 1$, $f_{22} = 2$ but $g_{22} = 4$, and $f_{23} = g_{23} = 5$. For $\bar{\boldsymbol{a}}_1 = CAGT$, both alignment methods lead to the same alignment,

$$\begin{array}{cccc} \boldsymbol{a} & = & CGAGT \\ \bar{\boldsymbol{a}}_1 & = & C\ \ AGT \end{array}.$$

In this case, we have $f_{11} = g_{11} = 1$, $f_{12} = g_{12} = 3$, $f_{13} = g_{13} = 4$, and $f_{14} = g_{14} = 5$. △

The following lemma, whose proof is omitted due to space limitation, describes the relationship between $\boldsymbol{r}$ and $\bar{\boldsymbol{r}}_j$.

Lemma 2. *Suppose the kth element of $\bar{\boldsymbol{a}}_j$ is α. Then $\bar{r}_{jk}$ is the sum of the elements of the set $\{r_{ji} : f_{jk} \leq i \leq g_{jk}, a_i = \alpha\}$.*

When considering $\boldsymbol{x}_j$, it follows from the lemma that if the $f_{jk} = g_{jk}$, then $\bar{r}_{jk} = r_{jf_{jk}}$. For the sake of simplicity, we only use these values for the purpose of decoding. Specifically, let R_i be the multiset of values $\bar{r}_{jk}$ for which $f_{jk} = g_{jk} = i$.

Example 3. Continuing Example 1, we have $R_1 = \{3, 3\},\ R_2 = \{\},\ R_3 = \{3\},\ R_4 = \{2\},\ R_5 = \{2, 2\}$. △

Quantizer function $\mathcal{Q}(R_i)$: As stated above, each b_i is decoded based on R_i, which consists of run lengths $r \sim D(t_{b_i})$. Let $\hat{N}_i$ denote the number of elements in R_i. For simplicity of notation and without loss of generality, we assume $R_i = \{r_{1,i}, r_{2,i}, \ldots, r_{\hat{N}_i,i}\}$.

We assume that D is a *Poisson distribution* with parameter λt, where λ determines the expected number of nucleotides added per unit time and t is the duration of the synthesis round. Hence, for all i, j, we have $r_{ji} \sim \mathrm{Poi}(\lambda t_{b_i})$. By rescaling the unit of time and λ, we assume t_1, the smallest synthesis time, is equal to 1. This allows us to represent the rate in a simple way, and compare with the conventional system that uses only a single synthesis time, whose rate is upper bounded by $\log_2 3$ bits. We note however that the approach is also applicable to other distributions.

The quantizer $\mathcal{Q}$ produces an estimate $\hat{b}_i$ of b_i as follows. Let $\bar{R}_i = \sum_{r \in R_i} r$ denote the sum of the elements of R_i. We define $\hat{b}_i$ as[3]

$$\hat{b}_i = \arg\max_{b \in \Sigma_L} \mathrm{Poi}(\bar{R}_i | \lambda \hat{N}_i t_b)\pi(t_b),$$

[3]We note that this estimator is not exactly the MAP estimator since the precise joint distribution $P(\boldsymbol{b}_i, R_i)$ is more complex due to the fact that the size $\hat{N}_i$ of R_i is also random and not independent of its elements.

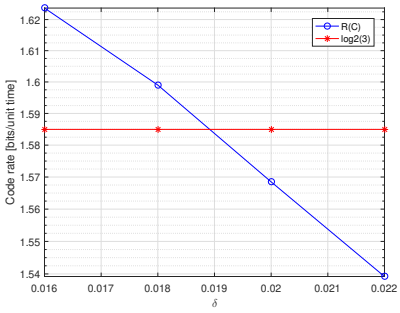

Figure 1. The lower bound of the code rate $R(\mathcal{C})$ of BTq-SC codes with respect to $\delta \in \{0.016, 0.018, 0.020, 0.022\}$.

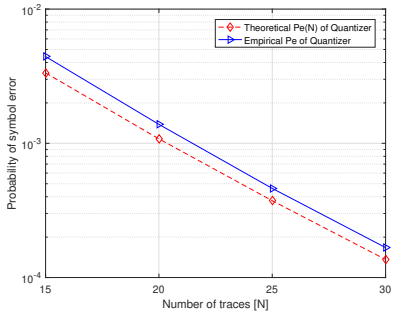

Figure 2. The comparison of empirical and theoretical error probabilities of estimating a synthesis time t_i. Here $\delta = 0.018$.

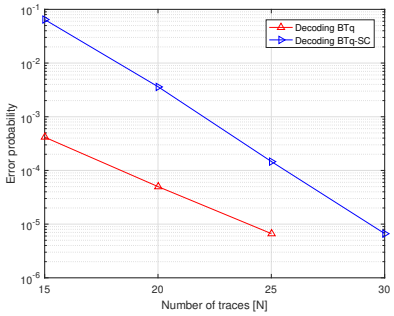

Figure 3. The error probability of decoding BTq and BTq-SC codewords. Here, $\delta = 0.018$.

where $\text{Poi}(k|\lambda)$ is the value of the Poisson pmf with parameter λ at k and $\pi(t_b)$ is the prior probability of $t_b \in T_{syn}$. It then follows that we can find $\hat{b}_i$ by comparing it with thresholds

$$\rho_l = \frac{\hat{N}_i \lambda (t_l - t_{l-1}) + \ln(\pi(t_{l-1})/\pi(t_l))}{\ln(t_l/t_{l-1})}, \quad l \in [L-1],$$

$\rho_0 = 0, \rho_L = \infty$. Specifically, if $\rho_l \le \bar{R}_i < \rho_{l+1}$, then $\hat{b}_i = l$. If $d_H(\hat{\boldsymbol{b}}, \boldsymbol{b}) \le \delta n/2$, then $\boldsymbol{b}$ can be identified correctly.

While it is difficult to determine the probability of incorrectly decoding b_i, an approximate lower bound may be obtained by considering the probability of incorrect decoding based on $(r_{1i}, \ldots, r_{Ni})$. Assuming a prior probability vector $\boldsymbol{\pi} = (\pi(t_0), \ldots, \pi(t_{L-1}))$, this probability is given as

$$P_e(N, \boldsymbol{\pi}) = \sum_{l=0}^{L-1} \pi(t_l) \, \text{Poi}(\bar{R}_i < \rho_l \cup \bar{R}_i \ge \rho_{l+1} | N\lambda t_l),$$

where $\bar{R}_i = \sum_{j=1}^{N} r_{j,i}$.

For information symbols of the codewords $\boldsymbol{b} \in \mathcal{C}_{sc}$, the prior distributions $\boldsymbol{\pi}$ can be obtained from the MEMC over $\mathcal{G}'_q$, which we denote by $\boldsymbol{\pi}_M$. To approximate the average probability of error for all symbols of the codeword, we assume that the parity symbols have a uniform distribution $\boldsymbol{\pi}_U$. Then the lower bound on the probability of symbol error can be approximated as

$$P_e(N) \simeq C_{\delta,L} P_e(N, \boldsymbol{\pi}_M) + (1 - C_{\delta,L}) P_e(N, \boldsymbol{\pi}_U). \quad (10)$$

VI. Code Parameters and Simulation Results

In this section, we will first briefly discuss some of the code parameters and then present the numerical results.

A. Parameters

If the number of deletions in each block of length m is too large, the error-correcting capabilities of the BTq code will become ineffective. Based on Poisson distribution, the deletion probability of a given run is at most $e^{-\lambda}$ when $t_1 = 1$. Hence the expected number of deleted runs is at least $\xi = me^{-\lambda}$ (note that more runs can be deleted due to merging). Choosing an appropriately small target value for ξ, we can set $m = \xi e^{\lambda}$. In the results below, we use $\xi = 1.7$.

In the simulation, we let $L = 3$, $\boldsymbol{a} \in S_q^n$, $\boldsymbol{b} \in \Sigma_L^n$, $T_{syn} = \{t_0, t_1, t_2\} = \{1, 2, 3\}$, $M = t_2 = 3$, $\lambda = 3.5$. Then the lengths of each block in the BTq code is $m = 57$, and we set the codeword length at $n = 4m = 228$. Furthermore, the expected number of deletions is about $4\xi = 6.8$ for each trace. Based on the MEMC over $\mathcal{G}'_q$, the prior probabilities for elements in $T_{syn} = \{1, 2, 3\}$ are $\boldsymbol{\pi}_M = (\pi(t_0), \pi(t_1), \pi(t_2)) = (0.759, 0.192, 0.049)$. The number of traces N is in the range $\{15, 20, 25, 30\}$. Furthermore, we set δ in the range $\{0.016, 0.018, 0.020, 0.022\}$.

B. Simulation Results

In this subsection, we present the simulation results. We note that given some aspects of our code construction are not explicit, they are not fully implemented in the simulation but rather the results are obtained based on their properties. In particular, the code $\mathcal{C}_{sc}$ is not explicit. We generate a set of strings $\boldsymbol{b}$ to simulate codewords from $\mathcal{C}_{sc}$ with the code rate $C_{\delta,L}$. For each string $\boldsymbol{b} \in \Sigma_L^n$, the first $\lfloor nC_{\delta,L} \rfloor$ message symbols satisfy the distribution $\boldsymbol{\pi}_M$ and the other $\lceil n(1 - C_{\delta,L}) \rceil$ parity symbols satisfy the uniform distribution $\boldsymbol{\pi}_U$. For $\boldsymbol{a}$, a random codeword from the code $\mathcal{C}_{BTq}$ with $\boldsymbol{\alpha} = (0, 0, 0, 0), \boldsymbol{\beta} = (1, 2, 3, 4)$ is chosen. Traces are generated based on codewords $(\boldsymbol{a}, \boldsymbol{b})$ and then decoded to produce $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}})$. If $\boldsymbol{a} = \hat{\boldsymbol{a}}$ and $d_H(\boldsymbol{b}, \hat{\boldsymbol{b}}) \le \lfloor n\delta/2 \rfloor$, decoding has succeeded.

Figure 2 compares the analytical and empirical probabilities of symbol error. The analytical error is based on $P_e(N)$ given in (10), which assumes all run lengths are available. It can be seen that the empirical error rate is close to this, implying that the synchronization method for identifying the run lengths (Subsection V-B) is effective. Furthermore, we can observe that the error rate falls nearly exponentially as N increases.

Figure 3 shows the error probability of decoding BTq codewords and BTq-SC codewords. Based on the simulation results, with N increasing, the error rate for BTq-SC gets close to that of BTq. Furthermore, as N increases, the error probability of decoding BTq-SC codewords quickly diminishes.

Based on the numerical results, given $\delta = 0.018$ and $N = 30$, the error-correcting code in Construction 3 can achieve a code rate higher than $\log_2 3$ bits per unit time and a decoding error probability lower than 10^{-5} simultaneously.

Acknowledgement: The authors thank Dr. Mete Civelek for helpful discussions and insightful comments.

REFERENCES

[1] 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.

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

[3] 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.

[4] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013.

[5] 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, p. 14138, 2015.

[6] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition*, vol. 54, no. 8, pp. 2552–2555, 2015.

[7] 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, p. 242, 2018.

[8] L. C. Meiser, P. L. Antkowiak, J. Koch, W. D. Chen, A. X. Kohll, W. J. Stark, R. Heckel, and R. N. Grass, "Reading and writing digital data in DNA," *Nature Protocols*, vol. 15, no. 1, pp. 86–101, 2020.

[9] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, pp. 1628–1628, 2012.

[10] P. Gaytán, "Chemical synthesis of oligonucleotides using acetone as a washing solvent," *Biotechniques*, vol. 47, no. 2, pp. 701–702, 2009.

[11] E. M. LeProust, B. J. Peck, K. Spirin, H. B. McCuen, B. Moore, E. Namsaraev, and M. H. Caruthers, "Synthesis of high-quality libraries of long (150mer) oligonucleotides by a novel depurination controlled process," *Nucleic acids research*, vol. 38, no. 8, pp. 2522–2540, 2010.

[12] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Coding for optimized writing rate in DNA storage," *ISIT2020*, vol. 1, pp. 1–6, 2020.

[13] M. Schwartz and J. Bruck, "On the capacity of the precision-resolution system," *IEEE transactions on information theory*, vol. 56, no. 3, pp. 1028–1037, 2010.

[14] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder, "Trace reconstruction with constant deletion probability and related results." in *SODA*, vol. 8, 2008, pp. 389–398.

[15] F. Nazarov and Y. Peres, "Trace reconstruction with $\exp(o(n^{1/3}))$ samples," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017, pp. 1042–1046.

[16] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6084–6103, 2020.

[17] J. Brakensiek, R. Li, and B. Spang, "Coded trace reconstruction in a constant number of traces," in *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2020, pp. 482–493.

[18] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, "Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 2453–2458.

[19] A. Lenz, I. Maarouf, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. G. i Amat, "Concatenated codes for recovery from multiple reads of DNA sequences," in *2020 IEEE Information Theory Workshop (ITW)*. IEEE, 2021, pp. 1–5.

[20] M. Abroshan, R. Venkataramanan, L. Dolecek, and A. G. i Fabregas, "Coding for deletion channels with multiple traces," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 1372–1376.

[21] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion (corresp.)," *IEEE Transactions on Information Theory*, vol. 30, no. 5, pp. 766–769, 1984.

[22] R. M. Roth, "Introduction to coding theory," *IET Communications*, vol. 47, 2006.

[23] M. George, S. Jafarpour, and F. Bullo, "Markov chains with maximum entropy for robotic surveillance," *IEEE Transactions on Automatic Control*, vol. 64, no. 4, pp. 1566–1580, 2018.