Efficient Encoding/Decoding of GC-Balanced Codes Correcting Tandem Duplications

Yeow Meng Chee, *Senior Member, IEEE*, Johan Chrisnata, Han Mao Kiah, *Member, IEEE*, and Tuan Thanh Nguyen, *Member, IEEE*

***Abstract*— Tandem duplication is the process of inserting a copy of a segment of DNA adjacent to the original position. Motivated by applications that store data in living organisms, Jain *et al.* (2017) proposed the study of codes that correct tandem duplications. All code constructions are based on *irreducible words*. Such code constructions are *almost optimal* to combat tandem duplications of length at most k where $k \leq 3$. However, the problem of designing efficient encoder/decoder for such codes has not been investigated. In addition, the method cannot be extended to deal with the case of arbitrary k, where $k \geq 4$. In this work, we study efficient encoding/decoding methods for irreducible words over general q-ary alphabet. Our methods provide the first known efficient encoder/decoder for q-ary codes correcting tandem duplications of length at most k, where $k \leq 3$. In particular, we describe an (ℓ, m)-finite state encoder and show that when $m = \Theta(1/\epsilon)$ and $\ell = \Theta(1/\epsilon)$, the encoder achieves rate that is ϵ away from the optimal rate. We also provide ranking/unranking algorithms for irreducible words and modify the algorithms to reduce the space requirements for the finite state encoder. Over the DNA alphabet (or quaternary alphabet), we also impose weight constraint on the codewords. In particular, a quaternary word is GC-balanced if exactly half of the symbols of are either C or G. Via a modification of Knuth's balancing technique, we provide an efficient method that translates quaternary messages into GC-balanced codewords and the resulting codebook is able to correct tandem duplications of length at most k, where $k \leq 3$. In addition, we provide the first known construction of codes to combat tandem duplications of length at most k, where $k \geq 4$. Such codes can correct duplication errors in linear-time and they are *almost optimal* in terms of rate.**

***Index Terms*— Error-correction codes, DNA storage, tandem duplication, GC-balanced codes.**

I. Introduction

ADVANCES in synthesis and sequencing technologies have made DNA macromolecules an attractive medium for digital information storage. Besides being biochemically robust, DNA strands offer ultrahigh storage densities of 10^{15}-10^{20} bytes per gram of DNA, as demonstrated in recent experiments (see [21, Table 1]).

These synthetic DNA strands may be stored *ex vivo* or *in vivo*. When the DNA strands are stored *ex vivo* or in a non-biological environment, code design takes into account the synthesizing and sequencing platforms being used (see [22] for a survey of the various coding problems). In contrast, when the DNA strands are stored *in vivo* or recombined with the DNA of a living organism, we design codes to correct errors due to the biological mutations.

This work looks at the latter case, and specifically, examines codes that correct errors due to *tandem duplications*. Tandem duplications or repeats is one of the two common repeats found in the human genome [11] and they are caused by slipped-strand mispairings [13]. They occur in DNA when a pattern of one or more nucleotides is repeated and the repetitions are directly adjacent to each other. For example, consider the string or word AGTAGTCTGC. The substring AGTAGT is a tandem repeat, and we say that AGTAGTCTGC is generated from AGTCTGC by a *tandem duplication* of length three.

Jain *et al.* [4] first proposed the study of codes that correct errors due to tandem duplications. In the same paper, Jain *et al.* used *irreducible* words (see Section I-A for definition) to construct a family of codes that correct tandem duplications of length at most k, where $k \in \{2, 3\}$. While these codes are optimal in size for the case $k = 2$, these codes are not optimal for $k = 3$, and in fact, Chee *et al.* [1] constructed a family of codes with strictly larger size. Recently, Jain *et al.* [5] looked at other error mechanisms, and studied the capacity of these tandem-duplication systems in the presence of point-mutation noise (substitution errors). In this paper, we first look at encoding/decoding methods for irreducible words. In particular, we provide linear-time algorithm that encodes into irreducible words with exact rate or close to the asymptotic rate. While the technique is standard in constrained coding [17] and combinatorics literature [14], our contribution is a detailed analysis of the space and time complexities of the respective algorithm.

To further reduce errors, we also impose certain weight constraints on the individual codewords. Specifically, the GC-content of a DNA string refers to the number of nucleotides that corresponds to G or C, and DNA strings with GC-content that are too high or too low are more prone to both synthesis and sequencing errors (see for example, [16], [20]). Therefore, most of the earlier work use DNA strings whose

Manuscript received March 27, 2019; revised February 29, 2020; accepted March 4, 2020. Date of publication March 16, 2020; date of current version July 14, 2020. The work of Yeow Meng Chee, Han Mao Kiah, and Tuan Thanh Nguyen was supported in part by the Ministry of Education, Singapore, under Grant MOE2015-T2-2-086. This article was presented in part at the 2018 IEEE International Symposium on Information Theory. *(Corresponding author: Tuan Thanh Nguyen.)*

Yeow Meng Chee is with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 119077 (e-mail: pvocym@nus.edu.sg).

Johan Chrisnata and Han Mao Kiah are with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 639798 (e-mail: johanchr001@ntu.edu.sg; hmkiah@ntu.edu.sg).

Tuan Thanh Nguyen is with the Singapore University of Technology and Design, Singapore 487372 (e-mail: tuanthanh_nguyen@sutd.edu.sg).

Communicated by P. Sadeghi, Associate Editor for Coding Techniques.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2020.2981069

GC-content is close to 50% as codewords and use randomizing techniques to encode binary message into the latter [15]. Recently, in addition to the GC-content constraint, Immink and Cai also studied the homopolymer runlength constraint for DNA codewords [6]. In this work, via a modification of Knuth's balancing technique, we provide a linear-time map that translates quaternary alphabet messages into GC-balanced codewords and the resulting codebook is able to correct tandem duplications of length at most three.

Unfortunately, the set of all irreducible words cannot be extended to combat tandem duplications of length at most k, when $k \geq 4$. Nevertheless, in Section VI, we provide two efficient constructions of q-ary codes capable of correcting tandem duplications of length at most k with $k \geqslant 4$. However, we require $q \geq k+1$. These code families are the first known families of $\leqslant k$-TD codes for $k \geqslant 4$.

In this work, we study codes that correct an *unbounded* number of tandem duplications. For works that deal with a *bounded* number of tandem duplications, we refer the interested reader to [4], [8], [12]. Recently, Kovačević [9] studies a model that bounds the number of tandem duplications for each segment, instead of bounding the total number of tandem duplications.

Before we state the main results of the paper, we go through certain notation.

A. Notation and Terminology

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Let $\Sigma_q = \{0, 1, \cdots q-1\}$ be an alphabet of $q \geqslant 2$ symbols. For a positive integer n, let Σ_q^n denote the set of all words of length n over Σ_q, and let Σ_q^* denote the set of all words over Σ_q with finite length. Given two words $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, we denote their concatenation by $\boldsymbol{xy}$.

We state the *tandem duplication* rules. For integers $k \leqslant n$ and $i \leqslant n-k$, we define $T_{i,k} : \Sigma_q^n \to \Sigma_q^{n+k}$ such that $T_{i,k}(\boldsymbol{x}) = \boldsymbol{uvvw}$, where $\boldsymbol{x} = \boldsymbol{uvw}$, $|\boldsymbol{u}| = i$, $|\boldsymbol{v}| = k$.

If a finite sequence of tandem duplications, where each duplication is of length at most k, is performed to obtain $\boldsymbol{y}$ from $\boldsymbol{x}$, then we say that $\boldsymbol{y}$ is a $\leqslant$*k-descendant* of $\boldsymbol{x}$, or $\boldsymbol{x}$ is a $\leqslant$*k-ancestor* of $\boldsymbol{y}$. Given a word $\boldsymbol{x}$, we define the $\leqslant$*k-descendant cone* of $\boldsymbol{x}$ as the set of all $\leqslant k$-descendants of $\boldsymbol{x}$ and denote this cone by $D^*_{\leqslant k}(\boldsymbol{x})$.

Example 1: Consider $\boldsymbol{x} = 01210$ over Σ_3. We have $T_{1,3}(\boldsymbol{x}) = 01211210$ and $T_{0,2}(01211210) = 0101211210$. So, $0101211210 \in D^*_{\leqslant 3}(\boldsymbol{x})$.

Definition 1 ($\leqslant$k-Tandem-Duplication Codes): A subset $\mathcal{C} \subseteq \Sigma_q^n$ is a $\leqslant$*k-tandem-duplication code* if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ and $\boldsymbol{x} \neq \boldsymbol{y}$, we have that $D^*_{\leqslant k}(\boldsymbol{x}) \cap D^*_{\leqslant k}(\boldsymbol{y}) = \varnothing$. We say that $\mathcal{C}$ is an $(n, \leqslant k; q)$-TD code.

The *size* of $\mathcal{C}$ is denoted by $|\mathcal{C}|$, while the *rate* of $\mathcal{C}$ is given by $(1/n) \log_q |\mathcal{C}|$. Given an infinite family $\{\mathcal{C}_n : \mathcal{C}_n \text{ is of length } n\}_{n=1}^{\infty}$, its *asymptotic rate* is given by $\operatorname{limsup}_{n\to\infty} (1/n) \log_q |\mathcal{C}_n|$.

When $q = 4$, consider the alphabet $\mathcal{D} = \{\mathtt{A}, \mathtt{T}, \mathtt{C}, \mathtt{G}\}$ and the following one-to-one correspondence between $\mathcal{D}$ and $\Sigma_4 = \{0, 1, 2, 3\}$: $\mathtt{A} \leftrightarrow 0$, $\mathtt{T} \leftrightarrow 1$, $\mathtt{C} \leftrightarrow 2$, $\mathtt{G} \leftrightarrow 3$.

Let n be even. We say that $\sigma \in \mathcal{D}^n$ is GC-*balanced* if the number of symbols in σ that correspond to C and G is exactly $n/2$. For DNA-based storage, we are interested in codewords that are GC-balanced.

B. Irreducible Words

Of interest is a family of tandem-duplication codes constructed by Jain *et al.* [4]. Crucial to the code construction is the concept of irreducible words and roots.

Definition 2: A word is $\leqslant$*k-irreducible* if it cannot be deduplicated into shorter words with deduplications of length at most k. We use $\mathrm{Irr}_{\leqslant k}(n, q)$ to denote the set of all $\leqslant k$-irreducible words of length n over Σ_q. The $\leqslant k$-ancestors of $\boldsymbol{x} \in \Sigma_q^*$ that are $\leqslant k$-irreducible words are called the $\leqslant$*k-roots* of $\boldsymbol{x}$.

Construction 1 (Jain et al. *[4]):* For $k \in \{2, 3\}$ and $n \geqslant k$. An $(n, \leqslant k; q)$-TD-code $\mathcal{C}(n, \leqslant k; q)$ is given by

$$\mathcal{C}(n, \leqslant k; q) \triangleq \bigcup_{i=1}^{n} \{\xi_{n-i}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathrm{Irr}_{\leqslant k}(i, q)\}.$$

Here, $\xi_i(\boldsymbol{x}) = \boldsymbol{x} z^i$, where z is the last symbol of $\boldsymbol{x}$.

We point out certain advantages of Construction 1.

(a) *Asymptotic optimal rates*. Jain *et al.* demonstrated that Construction 1 is optimal for $k = 2$. However, when $k = 3$, Chee *et al.* [1] provided constructions that achieve almost twice the size in Construction 1 (see [1, Table I]). Nevertheless, Kovačević recently demonstrated that the rates of the codes given by Construction 1 are asymptotically optimal [10].

Linear-time

(b) *decoding*. Consider $\boldsymbol{x} \in \mathcal{C}(n, \leqslant k; q)$ and we read $\boldsymbol{y} \in D^*_{\leqslant k}(\boldsymbol{x})$. To retrieve the codeword $\boldsymbol{x}$, we simply compute the $\leqslant k$-root of $\boldsymbol{y}$ and extend the root if the length of the root is shorter than n. Jain *et al.* showed that there is at most one root when $k \in \{2, 3\}$, while Chee *et al.* provided algorithms to compute the root in linear time [1].

In view of these points, we study other practical aspects of Construction 1. Specifically, we look at *efficient encoding* of messages in Σ_q^{ℓ} to codewords in $\boldsymbol{x} \in \mathcal{C}(n, \leqslant k; q)$ for some $\ell < n$.

To this end, we look at the rate of $\mathcal{C}(n, \leqslant k; q)$. Let $I_{\leqslant k}(n, q) \triangleq |\mathrm{Irr}_{\leqslant k}(n, q)|$. Then the size of $\mathcal{C}(n, \leqslant k; q)$ is given by $\sum_{i=1}^{n} I_{\leqslant k}(i, q)$. Let $\mathrm{rate}_{\leqslant k}(n, q)$ and $\mathrm{rate}_{\leqslant k}(q)$ denote the rate and asymptotic rate of $\mathcal{C}(n, \leqslant k; q)$, respectively. In other words, $\mathrm{rate}_{\leqslant k}(n, q) \triangleq (1/n) \log_q |\mathcal{C}(n, \leqslant k; q)|$ and $\mathrm{rate}_{\leqslant k}(q) \triangleq \lim_{n\to\infty} \mathrm{rate}_{\leqslant k}(n, q)$. Jain *et al.* observed that $\bigcup_{n=1}^{\infty} \mathrm{Irr}_{\leqslant k}(n, q)$ is a *regular language* (refer to [18] as a reference for regular language) and hence,

$$\mathrm{rate}_{\leqslant k}(q) = \lim_{n\to\infty} \frac{\log_q I_{\leqslant k}(n, q)}{n}. \tag{1}$$

Furthermore, using Perron-Frobenius theory (see [17]), Jain *et al.* computed $\mathrm{rate}_{\leqslant 3}(3)$ to be approximately 0.347934. In view of (1), we look at encoding of the words to $\mathrm{Irr}_{\leqslant k}(n, q)$ instead and the extension of our encoding methods to $\mathcal{C}(n, \leqslant k; q)$ is straightforward.

In the special case of $q = 4$, we are interested in $\leqslant k$-irreducible words that are GC-balanced. Specifically, we set

$$\begin{aligned}\mathrm{Irr}^{\mathcal{B}}_{\leq k}(n; 4) \;\; &\triangleq \;\; \{\boldsymbol{x} \in \mathcal{D}^n : \boldsymbol{x} \in \mathrm{Irr}_{\leq k}(n; 4) \\ &\qquad \text{and } \boldsymbol{x} \text{ is GC-balanced}\},\end{aligned}$$

and study efficient mapping of messages in Σ_4^ℓ into $\mathrm{Irr}_{\leq k}^{\mathcal{B}}(n;4)$. Since the latter is a subcode of $\mathcal{C}(n,\leqslant k;4)$, it is able to correct tandem duplications of length at most k, for $k=2,3$.

C. Our Contributions

(A) In Section II, we first develop a recursive formula for $I_{\leqslant k}(n,q)$ and hence, provide a *closed* formula for the asymptotic rate for $\mathcal{C}(n,\leqslant k;q)$. Then we compute $\mathrm{rate}_{\leqslant k}(q)$ for all q and $k\in\{2,3\}$. Using combinatorial insights provided by the recursive formula, we then provide efficient encoding methods and analyse their corresponding space and time complexities.

(B) In Section III, we propose an (ℓ,m)-finite state encoder that maps messages into words in $\mathrm{Irr}_{\leqslant k}(n,q)$ with rate ℓ/m for $k\in\{2,3\}$. Furthermore, we show that we can choose ℓ and m to be small and yet come close to the asymptotic rate. In particular, if we choose $m=\Theta(1/\epsilon)$ and $\ell=\Theta(1/\epsilon)$, we show that the rate is at least $\mathrm{rate}_{\leqslant k}(q)-\epsilon$. Here, the running time for the encoder is linear in codeword length n for constant ϵ.

(C) Using bijections developed in Section II, we provide a ranking/unranking encoder with rate equal to $(1/n)\log_q(I_{\leqslant k}(n,q))$ in Section IV. This algorithm runs in $O(n^2)$ time using $O(n^2)$ space. Furthermore, this ranking/unranking technique can be modified to reduce the space requirement from $O(q^m)$ to $O(m^2)$ for the (ℓ,m)-finite state encoder (Section III).

(D) Section V focusses on the case $q=4$ and provides a linear-time encoder that maps messages in Σ_4^ℓ into $\mathrm{Irr}_{\leq k}^{\mathcal{B}}(n;4)$. We achieve this by combining the finite state encoder of Section III and a novel modification of Knuth's balancing technique [7]. Our encoder introduces $2\log_2\ n+O(1)$ additional redundant bases, where n is the word length.

(E) In Section VI, we provide two efficient constructions of q-ary codes capable of correcting tandem duplications of length at most k, where $k\geq 4$ and $q\geqslant k+1$. These two families of codes are the first known families of $\leqslant k$-TD codes for $k\geqslant 4$. Linear-time error decoders are also provided in this section.

II. Enumerating Irreducible Words

In this section, we compute $\mathrm{rate}_{\leqslant k}(q)$ for all q and $k\in\{2,3\}$ by obtaining a recursive formula for $I_{\leqslant k}(n,q)$. While the Perron-Frobenius theory (see [17]) is sufficient to determine the asymptotic rates, the recursive formula is useful in the analysis of the finite state encoder in Section III and the development of the ranking/unranking methods in Section IV.

To this end, we partition the set of irreducible words into two classes and provide bijections from irreducible words of shorter lengths into them. Specifically, notice that the suffix of an irreducible word is of the form either aba or abc, where a, b, c are distinct symbols. Hence, we let $\mathrm{Irr}_{\leqslant k}^{(s)}(2,n,q)$ and $\mathrm{Irr}_{\leqslant k}^{(s)}(3,n,q)$ to denote the set of irreducible words, whose length-three suffixes have two and three distinct symbols, respectively.

In the case $k=2$, we consider the following maps for $n\geqslant 4$,

$$\phi:\mathrm{Irr}_{\leqslant 2}(n-1,q)\times[q-2]\to\mathrm{Irr}_{\leqslant 2}^{(s)}(3,n,q),$$
$$\psi:\mathrm{Irr}_{\leqslant 2}(n-2,q)\times[q-2]\to\mathrm{Irr}_{\leqslant 2}^{(s)}(2,n,q).$$

We first define ϕ. If $\boldsymbol{x}=x_1x_2\dots x_{n-1}\in\mathrm{Irr}_{\leqslant 2}(n-1,q)$ and $i\in[q-2]$, set σ to be the ith element in $\Sigma_q\setminus\{x_{n-2},x_{n-1}\}$. Then set $\phi(\boldsymbol{x},i)=x_1x_2\dots x_{n-1}\sigma$.

For ψ, let $\boldsymbol{x}=x_1x_2\dots x_{n-2}\in\mathrm{Irr}_{\leqslant 2}(n-2,q)$ and $i\in[q-2]$ and set σ to be the ith element in $\Sigma_q\setminus\{x_{n-3},x_{n-2}\}$. Then set $\psi(\boldsymbol{x},i)=x_1x_2\dots x_{n-2}\sigma x_{n-2}$.

Proposition 1: The maps ϕ and ψ are bijections.

Proof: We construct the inverse map for ϕ. Specifically, we set $\phi^{-1}:\mathrm{Irr}_{\leqslant 2}^{(s)}(3,n,q)\to\mathrm{Irr}_{\leqslant 2}(n-1,q)\times[q-2]$ such that $\phi^{-1}(\boldsymbol{x})=(x_1\dots x_{n-1},i)$, where i is the index of x_n in $\Sigma_q\setminus\{x_{n-2},x_{n-1}\}$. It can be verified that $\phi\circ\phi^{-1}$ and $\phi^{-1}\circ\phi$ are identity maps on their respective domains. Similarly, the inverse map for ψ is given by $\psi^{-1}:\mathrm{Irr}_{\leqslant 2}^{(s)}(2,n,q)\to\mathrm{Irr}_{\leqslant 2}(n-2,q)\times[q-2]$ such that $\psi^{-1}(\boldsymbol{x})=(x_1\dots x_{n-2},i)$, where i is the index of x_{n-1} in $\Sigma_q\setminus\{x_{n-3},x_{n-2}\}$. ■

The following corollary is then immediate.

Corollary 1: We have that $I_{\leqslant 2}(2,q)=q(q-1)$, $I_{\leqslant 2}(3,q)=q(q-1)^2$, and

$$I_{\leqslant 2}(n,q)=(q-2)I_{\leqslant 2}(n-1,q)+(q-2)I_{\leqslant 2}(n-2,q)\quad(2)$$

for $n\geqslant 4$. Therefore, the asymptotic rate is $\mathrm{rate}_{\leqslant 2}(q)=\log_q\lambda_2$, where $\lambda_2=(q-2+\sqrt{q^2-4})/2$.

The recurrence relation given by (2) defines a special instance of *Lucas sequences* with integer coefficients $q-2$ (see for example, [19]). When $q=3$, we recover the well-known Fibonacci recurrence.

In the next section, we are interested in irreducible words with certain prefixes or suffixes. Specifically, let $\boldsymbol{p}$ be a word of length $\ell<n$. Then we denote the set of irreducible words of length n with prefix $\boldsymbol{p}$ by $\mathrm{Irr}_{\leqslant k}^{(p)}(\boldsymbol{p},n,q)$. The set of irreducible words of length n with suffix $\boldsymbol{p}$ is denoted by $\mathrm{Irr}_{\leqslant k}^{(s)}(\boldsymbol{p},n,q)$.

Fix $\boldsymbol{p}$. Notice that the maps ϕ and ψ simply append one and two symbols to words in their domains. Hence, if we apply the maps to a word with prefix $\boldsymbol{p}$, the image also has the same prefix $\boldsymbol{p}$. Therefore, both ϕ and ψ remain as bijections when we restrict the domains and codomains to the irreducible words with prefix $\boldsymbol{p}$. In other words, we obtain a similar recursion for $\mathrm{Irr}_{\leqslant 2}^{(p)}(\boldsymbol{p},n,q)$.

Corollary 2: Let $\boldsymbol{p}\in\Sigma_q^\ell$ For $n\geqslant\ell+2$,

$$\begin{aligned}\left|\mathrm{Irr}_{\leqslant 2}^{(p)}(\boldsymbol{p},n,q)\right|=&(q-2)\left|\mathrm{Irr}_{\leqslant 2}^{(p)}(\boldsymbol{p},n-1,q)\right|\\&+(q-2)\left|\mathrm{Irr}_{\leqslant 2}^{(p)}(\boldsymbol{p},n-2,q)\right|.\end{aligned}\quad(3)$$

We conclude this section with the recursion for $\mathrm{Irr}_{\leqslant 3}(n,q)$.

Proposition 2: We have that $I_{\leqslant 3}(3,q)=q(q-1)^2$, $I_{\leqslant 3}(4,q)=q^2(q-1)(q-2)$, $I_{\leqslant 3}(5,q)=q(q-1)(q-2)(q^2-q-1)$ and for $n\geqslant 6$,

$$\begin{aligned}I_{\leqslant 3}(n,q)=&(q-2)I_{\leqslant 3}(n-1,q)+(q-3)I_{\leqslant 3}(n-2,q)\\&+(q-2)I_{\leqslant 3}(n-3,q).\end{aligned}\quad(4)$$

Therefore, $\mathrm{rate}_{\leqslant 3}(q) = \log_q \lambda_3$, where λ_3 is the largest real root of equation $x^3-(q-2)x^2-(q-3)x-(q-2)=0$.

Proof: Recall that for a word $\boldsymbol{p}$ of length $\ell < n$, $\mathrm{Irr}^{(s)}_{\leqslant k}(\boldsymbol{p}, n, q)$ is the set of irreducible words of length n with suffix $\boldsymbol{p}$. For any arbitrary set of words $\boldsymbol{L}$, we let $\mathrm{Irr}^{(s)}_{\leqslant k}(\boldsymbol{L}, n, q)$ to denote the set of irreducible words of length n with suffix in $\boldsymbol{L}$. To prove this proposition, we partition the set of irreducible words $\mathrm{Irr}_{\leqslant 3}(n, q)$ into three classes and provide bijections from irreducible words of shorter length into them. Specifically, we consider all possible suffixes of length six of an irreducible word. For a word $\boldsymbol{x} = x_1 \dots x_{n-5}x_{n-4}x_{n-3}x_{n-2}x_{n-1}x_n \in \mathrm{Irr}_{\leqslant 3}(n, q)$, if $x_n = x_{n-3} = a$, every suffix of length six must be of the form $\{bcabda, bcacba, bcacda, bcadba, bcadca, bcadea, abacba, abacda\}$, where a, b, c, d, e are different elements in Σ_q. On the other hand, if $x_n \neq x_{n-3}$, every suffix of length four must be of the form $\{abcd, abcb, abac\}$. Let $\boldsymbol{L}_1 = \{abcd, abcb, abac \mid a, b, c, d \in \Sigma_q\}$, $\boldsymbol{L}_2 = \{bcabda, bcacba, bcacda, bcadea, bcadca \mid a, b, c, d, e \in \Sigma_q\}$, $\boldsymbol{L}_3 = \{bcadba, abacba, abacda \mid a, b, c, d \in \Sigma_q\}$. We consider the following maps for $n \geqslant 6$.

$$\varphi_1 : \mathrm{Irr}_{\leqslant 3}(n-1, q) \times [q-2] \to \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_1, n, q),$$

$$\varphi_2 : \mathrm{Irr}_{\leqslant 3}(n-2, q) \times [q-2] \to \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_2, n, q),$$

$$\varphi_3 : \mathrm{Irr}_{\leqslant 3}(n-3, q) \times [q-3] \to \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_3, n, q).$$

Recall that $\mathrm{Irr}_{\leqslant k}(n, q) = \mathrm{Irr}^{(s)}_{\leqslant k}(3, n, q) \cup \mathrm{Irr}^{(s)}_{\leqslant k}(2, n, q)$. We first define φ_1. If $\boldsymbol{x} = x_1 \dots x_{n-3}x_{n-2}x_{n-1} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(3, n-1, q)$ and $i \in [q-2]$, then set σ to be the ith element in $\Sigma_q \setminus \{x_{n-3}, x_{n-1}\}$, and set

$$\varphi_1(\boldsymbol{x}, i) = \boldsymbol{x}\sigma = x_1 \dots x_{n-3}x_{n-2}x_{n-1}\sigma.$$

If $\boldsymbol{x} = x_1 \dots x_{n-3}x_{n-2}x_{n-1} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(2, n-1, q)$ where $x_{n-3} = x_{n-1}$ and $i \in [q-2]$, then set σ to be the ith element in $\Sigma_q \setminus \{x_{n-2}, x_{n-1}\}$, and set

$$\varphi_1(\boldsymbol{x}, i) = \boldsymbol{x}\sigma = x_1 \dots x_{n-1}x_{n-2}x_{n-1}\sigma.$$

Similarly, we now define φ_2, φ_3 as follows.
If $\boldsymbol{x} = x_1 \dots x_{n-5}x_{n-4}x_{n-3}x_{n-2} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(3, n-2, q)$ and $i \in [q-3]$, then if $x_{n-5} \notin \{x_{n-2}, x_{n-3}\}$, set σ to be the ith element in $\Sigma_q \setminus \{x_{n-5}, x_{n-3}, x_{n-2}\}$, otherwise if $x_{n-5} \in \{x_{n-2}, x_{n-3}\}$, set σ to be the ith element in $\Sigma_q \setminus \{x_{n-4}, x_{n-3}, x_{n-2}\}$, and set

$$\varphi_2(\boldsymbol{x}, i) = \boldsymbol{x}\sigma x_{n-3} = x_1 \dots x_{n-5}x_{n-4}x_{n-3}x_{n-2}\sigma x_{n-3}.$$

If $\boldsymbol{x} = x_1 \dots x_{n-5}x_{n-4}x_{n-3}x_{n-2} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(2, n-2, q)$ where $x_{n-4} = x_{n-2}$ and $i \in [q-3]$, then set σ to be the ith element in $\Sigma_q \setminus \{x_{n-5}, x_{n-3}, x_{n-2}\}$ and we set

$$\varphi_2(\boldsymbol{x}, i) = \boldsymbol{x}\sigma x_{n-3} = x_1 \dots x_{n-5}x_{n-4}x_{n-3}x_{n-2}\sigma x_{n-3}.$$

If $\boldsymbol{x} = x_1 \dots x_{n-5}x_{n-4}x_{n-3} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(3, n-3, q)$ and $i \in [q-2]$, then set σ to be the ith element in $\Sigma_q \setminus \{x_{n-5}, x_{n-3}\}$, and set

$$\varphi_3(\boldsymbol{x}, i) = \boldsymbol{x}\sigma x_{n-5}x_{n-3} = x_1 \dots x_{n-5}x_{n-4}x_{n-3}\sigma x_{n-5}x_{n-3}.$$

TABLE I

THE ASYMPTOTIC INFORMATION RATES FOR k-IRREDUCIBLE WORDS FOR $k \in \{2, 3\}$

q	3	4	5	6	7	8
$\mathrm{rate}_{\leqslant 2}(q)$	0.4380	0.7249	0.8280	0.8788	0.9081	0.9269
$\mathrm{rate}_{\leqslant 3}(q)$	0.3479	0.7054	0.8208	0.8753	0.9062	0.9258

If $\boldsymbol{x} = x_1 \dots x_{n-5}x_{n-4}x_{n-3} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(2, n-3, q)$ where $x_{n-5} = x_{n-3}$ and $i \in [q-2]$, then set σ to be the ith element in $\Sigma_q \setminus \{x_{n-4}, x_{n-3}\}$, and set

$$\varphi_3(\boldsymbol{x}, i) = \boldsymbol{x}\sigma x_{n-4}x_{n-3} = x_1 \dots x_{n-5}x_{n-4}x_{n-3}\sigma x_{n-4}x_{n-3}.$$

We can prove that φ_i is bijection for $i = 1, 2, 3$ by constructing the inverse map for each φ_i. We first prove φ_1 is bijection. Specifically, we set $\varphi_1^{-1} : \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_1, n, q) \to \mathrm{Irr}_{\leqslant 3}(n-1, q) \times [q-2]$ such that $\varphi_1^{-1}(\boldsymbol{x}) = (x_1 \dots x_{n-3}x_{n-2}x_{n-1}, i)$ where i is the index of x_n in $\Sigma_q \setminus \{x_{n-3}, x_{n-1}\}$ if $x_{n-1} \neq x_{n-3}$ or i is the index of x_n in $\Sigma_q \setminus \{x_{n-2}, x_{n-1}\}$ otherwise. It can be verified that $\varphi_1 \circ \varphi_1^{-1}$ and $\varphi_1^{-1} \circ \varphi_1$ are identity maps on their respective domains. Similarly, the inverse maps for φ_2, φ_3 are given by

$$\varphi_2^{-1} : \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_2, n, q) \to \mathrm{Irr}_{\leqslant 3}(n-2, q) \times [q-2],$$

$$\varphi_3^{-1} : \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_3, n, q) \to \mathrm{Irr}_{\leqslant 3}(n-3, q) \times [q-3],$$

such that $\varphi_2^{-1}(\boldsymbol{x}) = (x_1 \dots x_{n-3}x_{n-2}, i)$ where i is the index of x_{n-1} in

- $\Sigma_q \setminus \{x_{n-5}, x_{n-3}, x_{n-2}\}$ if $x_{n-5} \notin \{x_{n-3}, x_{n-2}\}$ or $x_{n-4} = x_{n-2}$,
- $\Sigma_q \setminus \{x_{n-4}, x_{n-3}, x_{n-2}\}$ if $x_{n-5} \in \{x_{n-3}, x_{n-2}\}$ and $x_{n-4} \neq x_{n-2}$.

and $\varphi_3^{-1}(\boldsymbol{x}) = (x_1 \dots x_{n-3}, i)$ where i is the index of x_{n-2} in

- $\Sigma_q \setminus \{x_{n-5}, x_{n-3}\}$ if $x_{n-5} \neq x_{n-3}$,
- $\Sigma_q \setminus \{x_{n-4}, x_{n-3}\}$ if $x_{n-5} = x_{n-3}$.

We can prove that φ_2, φ_3 are bijections as $\varphi_2 \circ \varphi_2^{-1}$, $\varphi_2^{-1} \circ \varphi_2$, $\varphi_3 \circ \varphi_3^{-1}$, and $\varphi_3^{-1} \circ \varphi_3$ are identity maps on their respective domains. Since $\mathrm{Irr}_{\leqslant 3}(n, q) = \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_1, n, q) \cup \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_2, n, q) \cup \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_3, n, q)$, we have

$$\begin{aligned} I_{\leqslant 3}(n, q) = (q-2)I_{\leqslant 3}(n-1, q) + (q-3)I_{\leqslant 3}(n-2, q) \\ + (q-2)I_{\leqslant 3}(n-3, q). \end{aligned}$$

■

The following corollary is analogous to Corollary 2.

Corollary 3: For any $\boldsymbol{p} \in \Sigma_q^{\ell}$ and $n \geqslant \ell + 3$, we have

$$\begin{aligned} \left|\mathrm{Irr}^{(p)}_{\leqslant 3}(\boldsymbol{p}, n, q)\right| =& (q-2)\left|\mathrm{Irr}^{(p)}_{\leqslant 3}(\boldsymbol{p}, n-1, q)\right| \\ &+ (q-3)\left|\mathrm{Irr}^{(p)}_{\leqslant 3}(\boldsymbol{p}, n-2, q)\right| \\ &+ (q-2)\left|\mathrm{Irr}^{(p)}_{\leqslant 3}(\boldsymbol{p}, n-3, q)\right|. \end{aligned} \quad (5)$$

Remark 1: We compute the values of $\mathrm{rate}_{\leqslant k}(q)$ for $k \in \{2, 3\}$ in Table I. Let $T(n, q)$ be the largest size of an $(n, \leqslant 3; q)$-TD code and define $\tau(q) \triangleq \limsup_{n\to\infty}(1/n)\log_q T(n, q)$. From [1], [4], we have that that $\mathrm{rate}_{\leqslant 3}(q) \leqslant \tau(q) \leqslant \mathrm{rate}_{\leqslant 2}(q)$. Therefore, Table I demonstrates that $\mathcal{C}(n, \leqslant 3; q)$ is *almost* optimal for $q \geqslant 5$.

When $q = 4$, we are interested in the asymptotic information rate for GC-balanced $\leqslant k$-irreducible words. Even though we do not provide an explicit recursion formula for the size of $\mathrm{Irr}^{\mathcal{B}}_{\leq k}(n; 4)$, the following asymptotic information rate is implied by the encoder described in Section V. We defer the proof of Corollary 4 to Section V.

Corollary 4: For $k \in \{2, 3\}$, we have that

$$\lim_{n \to \infty} \frac{\log_q |\mathrm{Irr}^{\mathcal{B}}_{\leq k}(n; 4)|}{n} = \mathrm{rate}_{\leqslant k}(q). \tag{6}$$

III. Finite State Encoder

In this section, we propose an (ℓ, m)-finite state encoder that maps messages into words in $\mathrm{Irr}_{\leqslant k}(n, q)$ with rate ℓ/m for $k \in \{2, 3\}$ (refer to [17] as a reference for finite state encoders). In particular, for arbitrary $\epsilon > 0$, if we choose $m = \Theta(1/\epsilon)$ and $\ell = \Theta(1/\epsilon)$, we show that the rate of our encoder is at least $\mathrm{rate}_{\leqslant k}(q) - \epsilon$.

For integers $\ell < m$, an (ℓ, m)-*finite state encoder* is a triplet $(\mathcal{S}, \mathcal{E}, \mathcal{L})$, where $\mathcal{S}$ is a set of *states*, $\mathcal{E} \subset \mathcal{S} \times \mathcal{S}$ is a set of *directed edges*, and $\mathcal{L} : \mathcal{E} \to \Sigma_q^\ell \times \Sigma_q^m$ is an *edge labeling*.

To encode into irreducible words, we choose $m \geqslant 2k - 1$, and set

$$\mathcal{S} \triangleq \mathrm{Irr}_{\leqslant k}(m, q) \text{ and } \mathcal{E} \triangleq \{(\boldsymbol{x}, \boldsymbol{x}') : \boldsymbol{x}\boldsymbol{x}' \in \mathrm{Irr}_{\leqslant k}(2m, q)\}.$$

For $\boldsymbol{x} \in \mathcal{S}$, we define the *neighbors* of $\boldsymbol{x}$ to be $N(\boldsymbol{x}) \triangleq \{\boldsymbol{x}' : (\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{E}\}$. We also consider the quantity $\Delta_{\leqslant k}(m, q) \triangleq \min\{|N(\boldsymbol{x})| : \boldsymbol{x} \in \mathcal{S}\}$ and choose ℓ such that

$$\Delta_{\leqslant k}(m, q) \geqslant q^\ell. \tag{7}$$

We now define the edge labeling $\mathcal{L}$ using this choice of ℓ. For $\boldsymbol{x} \in \mathcal{S}$, since $|N(\boldsymbol{x})| \geqslant q^\ell$, we may use the set Σ^ℓ to index the first q^ℓ words in $N(\boldsymbol{x})$. Hence, for $\boldsymbol{x}' \in \mathcal{S}$, if $\boldsymbol{x}'$ is one of the first q^ℓ words, we let $\boldsymbol{y}_{\boldsymbol{x}'} \in \Sigma^\ell$ to denote the index. Otherwise, we simply set $\boldsymbol{y}_{\boldsymbol{x}'} = -$. Therefore, for $(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{E}$, we set $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{y}_{\boldsymbol{x}'}, \boldsymbol{x}')$. Finally, we call this triplet $(\mathcal{S}, \mathcal{E}, \mathcal{L})$ an (ℓ, m)- *finite state encoder for irreducible words*.

Example 2: Let $k = 2$, $q = 3$, $m = 3$. Then $\mathcal{S} = \{010$, 012, 020, 021, 101, 102, 120, 121, 201, 202, 210, $212\}$, and

$$N(010) = \{201, 210, 212\},$$
$$N(012) = \{010, 012, 021, 101, 102\}.$$

We verify that $\Delta_{\leqslant 2}(3, 3) = 3$ and so, we choose $\ell = 1$. So, we can set $\mathcal{L}$ to map the edges exiting the state 010 as follow: $(010, 201) \mapsto (0, 201), (010, 210) \mapsto (1, 210)$, and $(010, 212) \mapsto (2, 212)$. We represent the mapping $\mathcal{L}$ using the following lookup table.

$\boldsymbol{x}$	$N(\boldsymbol{x})$				
	0	1	2	–	–
010	201	210	212	–	–
012	010	012	021	101	102
020	102	120	121	–	–
021	012	020	021	201	202
101	201	202	210	–	–
102	010	012	101	102	120
120	102	120	121	210	212
121	012	020	021	–	–
201	020	021	201	202	210
202	101	102	120	–	–
210	120	121	201	210	212
212	010	012	021	–	–

Here, to determine $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}')$, we look at the row corresponding to $\boldsymbol{x}$ and look at the column corresponding to $\boldsymbol{x}'$. If the column is $\boldsymbol{y}_{\boldsymbol{x}'}$, then $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{y}_{\boldsymbol{x}'}, \boldsymbol{x}')$. So, $\mathcal{L}(012, 010) = (0, 010)$.

A. Encoding

(ℓ, m)**-Finite State Encoder**. Let s be a positive integer and set $n = s\ell$.

Input: message $\boldsymbol{y} = \boldsymbol{y}_1\boldsymbol{y}_2 \ldots \boldsymbol{y}_s \in \Sigma^{s\ell}$, where $\boldsymbol{y}_i \in \Sigma^\ell$, for $1 \leq i \leq s$

Output: $\boldsymbol{x} \in \mathrm{Irr}_{\leqslant k}(sm; q)$

(I) Set $\boldsymbol{x}_0$ to be the first word lexicographically in $\mathcal{S} = \mathrm{Irr}_{\leqslant k}(m, q)$.

(II) For $i \in [s]$, set $\boldsymbol{x}_i$ to be the unique word such that $\mathcal{L}(\boldsymbol{x}_{i-1}, \boldsymbol{x}_i) = (\boldsymbol{y}_i, \boldsymbol{x}_i)$.

(III) The encoded word is $\boldsymbol{x} = \boldsymbol{x}_1\boldsymbol{x}_2 \ldots \boldsymbol{x}_s$.

Example 3 (Example 2 continued): Let $s = 3$ and consider the message $\boldsymbol{y} = 012$. First, we set $\boldsymbol{x}_0 = 010$. Then $\boldsymbol{x}_1 = 201$ since $\mathcal{L}(010, 201) = (0, 201)$. Similarly, $\boldsymbol{x}_2 = 021$ and $\boldsymbol{x}_3 = 021$. Therefore, the encoded word $\boldsymbol{x}$ is 201021021.

Since the encoded word has length sm, the (ℓ, m)-finite state encoder for irreducible words has rate ℓ/m. In the next subsection, we see that ℓ and m can be chosen in such a way that the rate ℓ/m approaches $\mathrm{rate}_{\leqslant k}(q)$ quickly.

B. Approaching the Asymptotic Information Rate

Pick $\epsilon > 0$. We find suitable values for ℓ and m so that the encoding rate satisfies

$$\ell/m \geqslant \mathrm{rate}_{\leqslant k}(q) - \epsilon. \tag{8}$$

In particular, we show that $\ell = \Theta(1/\epsilon)$ and $m = \Theta(1/\epsilon)$ suffice to guarantee (8).

Recall that ℓ and m are required to satisfy (7). Hence, we determine $\Delta_{\leqslant k}(m, q)$. These values have the same recursive structure as $I_{\leqslant k}(m, q)$ and therefore have the same growth rate.

Proposition 3: We have that $\Delta_{\leqslant 2}(3, q) = q(q-2)^2$, $\Delta_{\leqslant 2}(4, q) = (q-2)^2(q^2 - q - 1)$, and for $m \geqslant 5$,

$$\Delta_{\leqslant 2}(m, q) = (q-2)\Delta_{\leqslant 2}(m-1, q) + (q-2)\Delta_{\leqslant 2}(m-2, q). \tag{9}$$

Proof: Observe that by symmetry, $|N(\boldsymbol{x})| = |N(\boldsymbol{x}')|$ for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathrm{Irr}^{(s)}_{\leqslant 2}(2, m, q)$. Similarly, $|N(\boldsymbol{y})| = |N(\boldsymbol{y}')|$ for any $\boldsymbol{y}, \boldsymbol{y}' \in \mathrm{Irr}^{(s)}_{\leqslant 2}(3, m, q)$. Therefore, to show that $|N(\boldsymbol{x})| \leqslant |N(\boldsymbol{y})|$ for $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(2, m, q)$ and $\boldsymbol{y} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(3, m, q)$, it suffices to pick any $\boldsymbol{x}_0 \in \mathrm{Irr}^{(s)}_{\leqslant 2}(2, m, q)$ and any $\boldsymbol{y}_0 \in \mathrm{Irr}^{(s)}_{\leqslant 2}(3, m, q)$ and show that $|N(\boldsymbol{x}_0)| \leqslant |N(\boldsymbol{y}_0)|$.

Hence, we assume that $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(010, m, q)$ and $\boldsymbol{y} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(210, m, q)$. Then the neighbors of $\boldsymbol{x}$ and $\boldsymbol{y}$ are given by

$$N(\boldsymbol{x}) = \left\{ \boldsymbol{x}' : 10\boldsymbol{x}' \in \bigcup_{\sigma \notin \{0,1\}} \mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m+2, q) \right\}, \quad (10)$$

$$N(\boldsymbol{y}) = \left\{ \boldsymbol{y}' : 10\boldsymbol{y}' \in \bigcup_{\sigma \neq 0} \mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m+2, q) \right\}. \quad (11)$$

Since $N(\boldsymbol{x}) \subseteq N(\boldsymbol{y})$, the inequality $|N(\boldsymbol{x})| \leqslant |N(\boldsymbol{y})|$ follows. Hence, $\Delta_{\leqslant 2}(m, q) = |N(\boldsymbol{x})|$ where $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(010, m, q)$.

Since $\Delta_{\leqslant 2}(m, q) = \sum_{\sigma \notin \{0,1\}} \left|\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m+2, q)\right|$, the recursive equation (9) follows from Corollary 2. ■

Proposition 4: We have that

$$\Delta_{\leqslant 3}(5, q) = (q-2)(q^2-2q-1)^2,$$
$$\Delta_{\leqslant 3}(6, q) = (q-1)(q^5-6q^4+9q^3+4q^2-8q-9),$$
$$\Delta_{\leqslant 3}(7, q) = (q-2)(q^6-6q^4+9q^3+4q^2-8q-10q+3),$$

and for $m \geqslant 8$,

$$\Delta_{\leqslant 3}(m, q) = (q-2)\Delta_{\leqslant 3}(m-1, q) + (q-3)\Delta_{\leqslant 3}(m-2, q) + (q-2)\Delta_{\leqslant 3}(m-3, q). \quad (12)$$

Proof: Let $\boldsymbol{L} = \{abcab,\ abcac,\ abcad,\ abcba,\ abcbd,\ abaca,\ abacb,\ abacd,\ abcde,\ abcdb,\ abcdc,\ abcda$: where a, b, c, d, e are distinct symbols in $\Sigma_q\}$ be the set of all possible suffixes of length five of an irreducible word. We first show that $\Delta_{\leqslant 3}(m, q) = |N(\boldsymbol{x})|$ where $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(abcab, m, q)$, where a, b, c are distinct symbols in Σ_q. In other words, we need to show that for any $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(abcab, m, q)$ and for any $\boldsymbol{y} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{p}, m, q)$, where $\boldsymbol{p} \in \boldsymbol{L}$, $|N(\boldsymbol{x})| \leqslant |N(\boldsymbol{y})|$. We illustrate our idea of comparison by one particular case which is when $p = abcad$, while the remaining cases can be done similarly. Without loss of generality, we assume that $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(01201, m, q)$ and $\boldsymbol{y} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(03201, m, q)$. Then the neighbors of $\boldsymbol{x}$ and $\boldsymbol{y}$ are given by

$$N(\boldsymbol{x}) = \left\{ \boldsymbol{x}' : 201\boldsymbol{x}' \in \bigcup_{\sigma \notin \{1,2\}} \mathrm{Irr}^{(p)}_{\leqslant 3}(201\sigma, m+3, q) \right\}, \quad (13)$$

$$N(\boldsymbol{y}) = \left\{ \boldsymbol{y}' : 201\boldsymbol{y}' \in \bigcup_{\sigma \neq 1} \mathrm{Irr}^{(p)}_{\leqslant 3}(201\sigma, m+3, q) \right\}. \quad (14)$$

Since $N(\boldsymbol{x}) \subseteq N(\boldsymbol{y})$, the inequality $|N(\boldsymbol{x})| \leqslant |N(\boldsymbol{y})|$ follows. Hence, $\Delta_{\leqslant 3}(m, q) = |N(\boldsymbol{x})|$ where $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(01201, m, q)$.

Since $\Delta_{\leqslant 3}(m, q) = \sum_{\sigma \notin \{1,2\}} \left|\mathrm{Irr}^{(p)}_{\leqslant 3}(201\sigma, m+3, q)\right|$, the recursive equation (12) follows from Corollary 3. ■

Recall that λ_2 and λ_3 are roots of the equations $x^2-(q-2)x-(q-2) = 0$ and $x^3-(q-2)x^2-(q-3)x-(q-2) = 0$, respectively. Set κ_2 such that $\Delta_{\leqslant 2}(m, q) \geqslant \kappa_2\lambda_2^m$ for $m \in \{3, 4\}$. Similarly, set κ_3 so that $\Delta_{\leqslant 3}(m, q) \geqslant \kappa_3\lambda_3^m$ for $m \in \{5, 6, 7\}$. By inductive argument, it follows from (9) and (12) that for $k \in \{2, 3\}$,

$$\Delta_{\leqslant k}(m, q) \geqslant \kappa_k\lambda_k^m \text{ for all } m. \quad (15)$$

We are now ready to present the main theorem of this section.

Theorem 1: Let $k \in \{2, 3\}$. Set $c_k = \mathrm{rate}_{\leqslant k}(q) = \log_q \lambda_k$. For $\epsilon > 0$, if we choose m and ℓ such that

$$\ell = \left\lceil \frac{(c_k-\epsilon)(c_k-\log_q \kappa_k)}{\epsilon} \right\rceil, \quad (16)$$

$$m = \left\lceil \frac{\ell-\log_q \kappa_k}{c_k} \right\rceil, \quad (17)$$

then the (ℓ, m)-finite state encoder has rate at least $\mathrm{rate}_{\leqslant k}(q)-\epsilon$.

Proof: We have to verify that (7) and (8) hold for the choice of ℓ and m. Now, (16) implies that $\epsilon\ell \geqslant (c_k-\epsilon)(c_k-\log_q \kappa_k)$, and equivalently, $c_k\ell/(\ell-\log_q \kappa_k+c_k) \geqslant c_k-\epsilon$. Therefore,

$$\frac{\ell}{m} \geqslant \frac{\ell}{1+(\ell-\log_q \kappa_k)/c_k} = \frac{c_k\ell}{\ell-\log_q \kappa_k+c_k} \geqslant c_k-\epsilon.$$

Thus, we verify (8).

Next, from (15) and (17), we have that

$$\Delta_{\leqslant k}(m, q) \geqslant \kappa_k\lambda_k^{(\ell-\log_q \kappa_k)/\log_q \lambda_k} = q^\ell.$$

Hence, we verify (7) and complete the proof. ■

Therefore, to achieve encoding rate at least $\mathrm{rate}_{\leqslant k}(q)-\epsilon$, we only require $\ell = \Theta(1/\epsilon)$ and $m = \Theta(1/\epsilon)$. If we naively use a lookup table to represent $(\mathcal{S}, \mathcal{E}, \mathcal{L})$, we require $q^{\Theta(1/\epsilon)}$ space. Furthermore, using binary search, the (ℓ, m)-finite state encoder for irreducible words encodes in $O(n/\epsilon)$ time. In the next section, we use combinatorial insights from (2) and (4) to reduce the space requirement to $O(1/\epsilon^2)$.

IV. Ranking/Unranking Algorithm

A *ranking function* for a finite set S of cardinality N is a bijection $\mathrm{rank} : S \to [N]$. Associated with the function rank is a unique *unranking function* $\mathrm{unrank} : [N] \to S$, such that $\mathrm{rank}(s) = j$ if and only if $\mathrm{unrank}(j) = s$ for all $s \in S$ and $j \in [N]$. In this section, we present an algorithm for ranking and unranking $\mathrm{Irr}_{\leqslant k}(n, q)$. This method is also known as *enumerative coding* [3]. For ease of exposition, we focus on the case where $k = 2$ and present the ranking/unranking algorithm for $k = 3$ at the end of the section. The basis of our ranking and unranking algorithms is the bijections defined in Section II. As implied by the codomains of ϕ and ψ, for $n \geqslant 4$, we order the words in $\mathrm{Irr}_{\leqslant 2}(n, q)$ such that words in $\mathrm{Irr}^{(s)}_{\leqslant 2}(3, n, q)$ are ordered before words in $\mathrm{Irr}^{(s)}_{\leqslant 2}(2, n, q)$. For words in $\mathrm{Irr}_{\leqslant 2}(2, q)$ and $\mathrm{Irr}_{\leqslant 2}(3, q)$, we simply order them lexicographically. We illustrate the idea behind the unranking algorithm through an example.

Example 4: Let $n = 6$ and $q = 3$. Then the values of $I_{\leqslant 2}(m, q)$ are as follow.

m	2	3	4	5	6
$I_{\leqslant 2}(m,q)$	6	12	18	30	48

Suppose we want to compute $\mathrm{unrank}(40)$. Proposition 1 gives

$$\mathrm{Irr}_{\leqslant 2}(6,3) = \phi(\mathrm{Irr}_{\leqslant 2}(5,3)\times[1]) \cup \psi(\mathrm{Irr}_{\leqslant 2}(4,3)\times[1]).$$

Now, we are interested in the 40th word of $\mathrm{Irr}_{\leqslant 2}(6,3)$. Since $40 > I_{\leqslant 2}(5,3) = 30$, the 40th word of $\mathrm{Irr}_{\leqslant 2}(6,3)$ is the image of the $40-30 = 10$th word in $\mathrm{Irr}_{\leqslant 2}(4,3)$ under ψ. Recursing tells us that the 10th word in $\mathrm{Irr}_{\leqslant 2}(4,3)$ is the image of the 10th word under $\phi(\mathrm{Irr}_{\leqslant 2}(3,3)\times[1])$. The 10th word of $\mathrm{Irr}_{\leqslant 2}(3,3)$ is 202. This gives

$$\begin{aligned}\mathrm{unrank}(40) &= \psi(\phi(202,1),1)\\ &= \psi(2021,1) = 202101.\end{aligned}$$

The formal unranking algorithm is described in Algorithm 1.

Algorithm 1 $\texttt{unrank}(n,q,j)$

```
Input: Integers $n \geq 2$, $q \geqslant 3$, $1 \leq j \leq I_{\leqslant 2}(n,q)$
Output: $\boldsymbol{x}$, where $\boldsymbol{x}$ is the codeword of rank $j$ in $\mathrm{Irr}_{\leqslant 2}(n,q)$
  if $n \leq 3$ then
    return $j$-th codeword in $\mathrm{Irr}_{\leqslant 2}(n,q)$
  if $j \leqslant (q-2)I_{\leqslant 2}(n-1,q)$ then
    $j' \leftarrow 1 + \lfloor (j-1)/(q-2) \rfloor$
    $i \leftarrow (j-1) \pmod{q-2} + 1$
    return $\phi(\texttt{unrank}(n-1,q,j'),i)$
  else
    $j' \leftarrow 1 + \lfloor (j-(q-2)I_{\leqslant 2}(n-1,q)-1)/(q-2) \rfloor$
    $i \leftarrow (j-(q-2)I_{\leqslant 2}(n-1,q)-1) \pmod{q-2} + 1$
    return $\psi(\texttt{unrank}(n-2,q,j'),i)$
```

The corresponding ranking algorithm for $\mathrm{Irr}_{\leqslant 2}(n,q)$ has a similar recursive structure and is described in Algorithm 2.

Algorithm 2 $\texttt{rank}(n,q,\boldsymbol{x})$

```
Input: $n \geq 2$, $q \geqslant 3$ and irreducible word $\boldsymbol{x}$ of length $n$
Output: $j$, where $1 \leq j \leq I_{\leqslant 2}(n,q)$, the rank of $\boldsymbol{x}$ in $\mathrm{Irr}_{\leqslant 2}(n,q)$
  if $n \leq 3$ then
    return $\texttt{rank}(\boldsymbol{x})$ in $\mathrm{Irr}_{\leqslant 2}(n,q)$
  if $x_n \neq x_{n-2}$ then
    $\boldsymbol{x}' \leftarrow x_1 x_2 \ldots x_{n-1}$
    $i \leftarrow$ the index of $x_n$ in $\Sigma_q \setminus \{x_{n-2}, x_{n-1}\}$
    return $(\texttt{rank}(n-1,q,\boldsymbol{x}')-1)(q-2)+i$
  else
    $\boldsymbol{x}' \leftarrow x_1 x_2 \ldots x_{n-2}$
    $i \leftarrow$ the index of $x_{n-1}$ in $\Sigma_q \setminus \{x_{n-3}, x_{n-2}\}$
    return $(\texttt{rank}(n-2,q,\boldsymbol{x}')-1)(q-2)+i+(q-2)I_{\leqslant 2}(n-1,q)$
```

Example 5: Let $n = 6$ and $q = 3$ as before. Suppose we want to compute $\texttt{rank}(202101)$. Since $202101 \in \mathrm{Irr}^{(s)}_{\leqslant 2}(2,6,3)$, we have that 202101 is obtained from applying ψ to $2021 \in \mathrm{Irr}_{\leqslant 2}(4,3)$. Again, since $2021 \in \mathrm{Irr}^{(s)}_{\leqslant 2}(3,6,3)$, we have that 2021 is obtained from applying ϕ to $202 \in \mathrm{Irr}_{\leqslant 2}(3,3)$. Therefore,

$$\begin{aligned}\mathrm{rank}(202101) &= \mathrm{rank}(2021) + I_{\leqslant 2}(5,3)\\ &= \mathrm{rank}(202) + I_{\leqslant 2}(5,3)\\ &= 10 + 30 = 40\end{aligned}$$

The set of values of $\{I_{\leqslant 2}(m,q) : m \leqslant n\}$ required in Algorithms 1 and 2 can be precomputed based on the recurrence (2). Since the numbers $I_{\leqslant 2}(m,q)$ grow exponentially, these n stored values require $O(n^2)$ space.

Next, Algorithms 1 and 2 involve $O(n)$ iterations and each iteration involves a constant number of arithmetic operations. Therefore, Algorithms 1 and 2 involve $O(n)$ arithmetics operations and have time complexity $O(n^2)$. Similarly, the corresponding ranking/unranking algorithm for $\mathrm{Irr}_{\leqslant 3}(n,q)$ have similar recursive structures and are described in Algorithm 3 and 4.

Algorithm 3 $\texttt{unrank}(n,q,j)$

```
Input: Integers $n \geq 3$, $q \geqslant 3$, $1 \leq j \leq I_{\leqslant 3}(n,q)$
Output: $\boldsymbol{x}$, where $\boldsymbol{x}$ is the codeword of rank $j$ in $\mathrm{Irr}_{\leqslant 3}(n,q)$
  if $n \leq 5$ then
    return $j$-th codeword in $\mathrm{Irr}_{\leqslant 3}(n,q)$
  if $j \leqslant (q-2)I_{\leqslant 3}(n-1,q)$ then
    $j' \leftarrow 1 + \lfloor (j-1)/(q-2) \rfloor$
    $i \leftarrow (j-1) \pmod{q-2} + 1$
    return $\varphi_1(\texttt{unrank}(n-1,q,j'),i)$
  else
    $j' \leftarrow j-(q-2)I_{\leqslant 3}(n-1,q)$
  if $j' \leqslant (q-3)I_{\leqslant 3}(n-2,q)$ then
    $j' \leftarrow 1 + \lfloor (j'-1)/(q-3) \rfloor$
    $i \leftarrow (j'-1) \pmod{q-3} + 1$
    return $\varphi_2(\texttt{unrank}(n-2,q,j'),i)$
  else
    $j' \leftarrow j'-(q-3)I_{\leqslant 3}(n-2,q)$
    $i \leftarrow (j'-1) \pmod{q-2} + 1$
    return $\varphi_3(\texttt{unrank}(n-3,q,j'),i)$
```

Algorithm 4 $\texttt{rank}(n,q,\boldsymbol{x})$

```
Input: $n \geq 3$, $q \geqslant 3$ and irreducible word $\boldsymbol{x}$ of length $n$
Output: $j$, where $1 \leq j \leq I_{\leqslant 3}(n,q)$, the rank of $\boldsymbol{x}$ in $\mathrm{Irr}_{\leqslant 3}(n,q)$
  if $n \leq 5$ then
    return $\texttt{rank}(\boldsymbol{x})$ in $\mathrm{Irr}_{\leqslant 3}(n,q)$
  if $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_1,n,q)$ then
    $(\boldsymbol{x}',i) \leftarrow \varphi_1^{-1}(\boldsymbol{x})$
    return $(\texttt{rank}(n-1,q,\boldsymbol{x}')-1)(q-2)+i$
  if $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_2,n,q)$ then
    $(\boldsymbol{x}',i) \leftarrow \varphi_2^{-1}(\boldsymbol{x})$
    return $(\texttt{rank}(n-2,q,\boldsymbol{x}')-1)(q-3)+i+(q-2)I_{\leqslant 3}(n-1,q)$
  if $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 3}(\boldsymbol{L}_3,n,q)$ then
    $(\boldsymbol{x}',i) \leftarrow \varphi_3^{-1}(\boldsymbol{x})$
    return $(\texttt{rank}(n-3,q,\boldsymbol{x}')-1)(q-2)+i+(q-2)I_{\leqslant 3}(n-1,q)+(q-3)I_{\leqslant 3}(n-2,q)$
```

A. Reducing the Space Requirement for the Finite State Encoder

As discussed earlier, a naive implementation of the (ℓ,m)-finite state encoder in Section III requires $q^{\Theta(m)}$ space (assuming $\ell = \Theta(m)$). Here, we modify our unranking algorithm to reduce the space requirement to $O(m)$ integers or $O(m^2)$ bits.

Recall the notation in Section III. In particular, let $\boldsymbol{x}_{i-1} \in \mathrm{Irr}_{\leqslant 2}(m,q)$ and $\boldsymbol{y}_i \in \Sigma_q^{\ell}$. Our encoding task is to determine the irreducible word $\boldsymbol{x}_i$ in $N(\boldsymbol{x}_{i-1})$ whose index corresponds to $\boldsymbol{y}_i$. Equivalently, if j is the rank of $\boldsymbol{y}_i \in \Sigma_q^{\ell}$, then our task is to find $\boldsymbol{x}_i$ such that its rank in $N(\boldsymbol{x}_{i-1})$ is j. Since $\boldsymbol{x}_{i-1}$ is irreducible and using symmetry, we assume that $\boldsymbol{x}_{i-1} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(010, m, q)$ or $\boldsymbol{x}_{i-1} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(210, m, q)$. Furthermore, (10) and (11) imply that $N(\boldsymbol{x}_{i-1})$ corresponds to a union of $\leqslant$2-irreducible words with prefixes of the form 10σ. Therefore, it suffices to provide ranking/unranking algorithms for $\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m, q)$.

Since (3) implies that $\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m, q)$ has the same recursive structure as $\mathrm{Irr}_{\leqslant 2}(m, q)$, we can modify Algorithms 1 and 2 to unrank and rank $\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m, q)$.

Now, to rank/unrank $\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m, q)$, we require $O(m)$ precomputed integers. Assuming q is constant, we require only $O(m)$ integers or $O(m^2)$ bits. However, the running time is increased to $O(m^2)$.

V. Encoder for DNA GC-Balanced Codes

Set $q = 4$ and $k \in \{2,3\}$. Recall that $\mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(n;4)$ is the set of all GC-balanced $\leqslant k$-irreducible words of length n. In this section, we look at efficient ways of encoding quaternary messages in $\Sigma_4^{\ell} = \{0,1,2,3\}^{\ell}$ into codewords in $\mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(N;4)$.

To do so, we first map a message $\boldsymbol{x} \in \Sigma_4^{\ell}$ to an irreducible word $\boldsymbol{y} \in \mathrm{Irr}_{\le k}(n;4)$ using the finite state encoder in Section III. We then modify *Knuth's balancing technique* to give an linear-time algorithm that encodes $\boldsymbol{y}$ to $\boldsymbol{z} \in \mathrm{Irr}^{\mathcal{B}}_{\le k}(n+r;4)$ by introducing additional r redundant bases. We refer to the latter algorithm as the GC-*balanced encoder*. In Section V-A, we describe the algorithm in detail and show that r is at most $2\lceil \log_2 n \rceil + 8$. Consequently, if the rate of the initial finite state encoder is $\mathrm{rate}_{\leqslant k}(4) - \epsilon$, the rate of the GC-balanced encoder tends to $\mathrm{rate}_{\leqslant k}(4) - \epsilon$ as word length increases.

Alternatively, we can modify the finite state encoder $(\mathcal{S}_{\mathcal{B}}, \mathcal{E}_{\mathcal{B}}, \mathcal{L}_{\mathcal{B}})$ in Section III so that the resulting codewords belong to $\mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(N;4)$. In Section V-B, we describe this alternative finite state encoder in detail and compare it with our GC-balanced encoder.

A. GC-*Balanced Encoder*

We first modify Knuth's balancing technique to map a quaternary $\leqslant k$-irreducible word to a GC-balanced $\leqslant k$-irreducible word. Specifically, Knuth's balancing technique is a linear-time algorithm that maps a binary message $\boldsymbol{x}$ to a balanced word $\boldsymbol{z}$ of the same length by flipping the first t bits of $\boldsymbol{x}$ [7]. The crucial observation demonstrated by Knuth is that such an index t always exists and t is commonly referred to as the *balancing index*. Formally, we have the following theorem.

Theorem 2 (Knuth [7]): There exists a pair of linear-time algorithms $\textsc{Enc} : \{0,1\}^n \to \{0,1\}^n \times [n]$ and $\textsc{Dec} : \{0,1\}^n \times [n] \to \{0,1\}^n$ such that the following holds. If $\textsc{Enc}(\boldsymbol{x}) = (\boldsymbol{z}, t)$, then $\boldsymbol{z}$ is balanced and $\textsc{Dec}(\boldsymbol{z}, t) = \boldsymbol{x}$.

To represent the balancing index, Knuth appends $\boldsymbol{z}$ with a short balanced suffix of length $\lceil \log n \rceil$ and so, a lookup table of size n is required. However, for the encoding of irreducible words, this short balanced suffix may introduce unwanted duplications and so, certain modifications are required. Here, we develop a method to represent the balancing index with a balanced suffix of length $2\lceil \log n \rceil$. In contrast with Knuth's technique, the balancing index can be encoded and decoded in linear time *without* the use of a lookup table.

To apply Knuth's method, we define the flipping rule $f : \mathcal{D} \to \mathcal{D}$, where

$$f(\mathtt{A}) = \mathtt{C}, f(\mathtt{C}) = \mathtt{A}, f(\mathtt{G}) = \mathtt{T} \text{ and } f(\mathtt{T}) = \mathtt{G}.$$

For a word $\boldsymbol{x} \in \mathcal{D}^n$ and index i with $0 \le i \le n$, $f_i(\boldsymbol{x})$ denotes the word obtained by flipping the first i symbols of $\boldsymbol{x}$ under the mapping f. Similar to Knuth's balancing method, it is easy to see that there exists an index i so that $f_i(\boldsymbol{x})$ is GC-balanced. Let $\mathrm{Index}(\boldsymbol{x}) \triangleq \min_{0 \le i < n}\{i : f_i(\boldsymbol{x}) \text{ is GC-balanced}\}$.

Example 6: Let $\boldsymbol{x} = \mathtt{TCACGCATCG}$. Observe that $f_5(\boldsymbol{x}) = \mathtt{GACATCATCG}$ is GC-balanced, while $f_i(\boldsymbol{x})$ is not GC-balanced for $0 \leqslant i < 5$. Hence, $\mathrm{Index}(\boldsymbol{x}) = 5$.

Let $\mathrm{Index}(\boldsymbol{x})$ be t. The above example also illustrates that even though $\boldsymbol{x}$ is $\leqslant k$-irreducible, the word $f_t(\boldsymbol{x})$ may not be $\leqslant k$-irreducible. Nevertheless, we observe that certain prefix and suffix of $f_t(\boldsymbol{x})$ remain $\leqslant k$-irreducible. For brevity, given a word $\boldsymbol{x} \in \mathcal{D}^n$, we use $\boldsymbol{P}_i(\boldsymbol{x})$ and $\boldsymbol{S}_i(\boldsymbol{x})$ to denote the prefix and suffix of $\boldsymbol{x}$ of length i, respectively.

Proposition 5: Let $0 \leqslant t \leqslant n$. If a word $\boldsymbol{x}$ is $\leqslant k$-irreducible and $\boldsymbol{y} = f_t(\boldsymbol{x})$, then $\boldsymbol{P}_t(\boldsymbol{y})$ and $\boldsymbol{S}_{n-t}(\boldsymbol{y})$ are both $\leqslant k$-irreducible.

Proof: The complement of $\boldsymbol{P}_t(\boldsymbol{y})$ is $\boldsymbol{P}_t(\boldsymbol{x})$. Since $\boldsymbol{x} \in \mathrm{Irr}_{\le k}(n;4)$, $\boldsymbol{P}_t(\boldsymbol{x}) \in \mathrm{Irr}_{\le k}(t;4)$. Therefore, since f is a bijection of alphabets in $\mathcal{D}$, it implies that $\boldsymbol{P}_t(\boldsymbol{y}) \in \mathrm{Irr}_{\le k}(t;4)$. On the other hand, $\boldsymbol{S}_{n-t}(\boldsymbol{y}) = \boldsymbol{S}_{n-t}(\boldsymbol{x})$, which is the suffix of length $(n-t)$ of $\boldsymbol{x}$. Hence, we also have $\boldsymbol{S}_{n-t}(\boldsymbol{y}) \in \mathrm{Irr}_{\le k}(n-t;4)$. ■

Example 7 (Example 6 continued): Observe that $\boldsymbol{x} = \mathtt{TCACGCATCG}$ is $\leqslant$ 3-irreducible, but $\boldsymbol{y} = f_5(\boldsymbol{x}) = \mathtt{GACATCATCG}$ is not $\leqslant$ 3-irreducible. Nevertheless, we have $\boldsymbol{P}_5(\boldsymbol{y}) = \mathtt{GACAT}$ and $\boldsymbol{S}_5(\boldsymbol{y}) = \mathtt{CATCG}$ are both $\leqslant$ 3-irreducible.

Hence, even though the resulting word $\boldsymbol{y} = f_t(\boldsymbol{x})$ may not be irreducible, Proposition 5 hints at the following solution. Add redundant bases between $\boldsymbol{P}_t(\boldsymbol{y})$ and $\boldsymbol{S}_{n-t}(\boldsymbol{y})$ so that the extended word is both balanced and irreducible.

The high-level idea of our algorithm is as follows. Suppose that $k \in \{2,3\}$ and $\boldsymbol{x} \in \mathrm{Irr}_{\le k}(n;4)$. To encode $\boldsymbol{x}$ into $\mathrm{Irr}^{\mathcal{B}}_{\le k}(n;4)$, we have the following three steps.

(Step 1) Let $t = \mathrm{Index}(\boldsymbol{x})$ and set $\boldsymbol{y} = f_t(\boldsymbol{x})$. So, $\boldsymbol{y} = \boldsymbol{P}_t(\boldsymbol{y})\boldsymbol{S}_{n-t}(\boldsymbol{y})$.

(Step 2) We encode the balancing index t to a word $\boldsymbol{p} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(2\lceil \log_2 n \rceil; 4)$.

(Step 3) Given $\boldsymbol{P}_t(\boldsymbol{y}), \boldsymbol{S}_{n-t}(\boldsymbol{y})$ and $\boldsymbol{p}$, we construct two linking words $\boldsymbol{R}_1, \boldsymbol{R}_2$ of length four such that the resulting word $\boldsymbol{w} = \boldsymbol{P}_t(\boldsymbol{y})\boldsymbol{R}_1\boldsymbol{S}_{n-t}(\boldsymbol{y})\boldsymbol{R}_2\boldsymbol{p}$ is GC-balanced and $\leqslant k$-irreducible. We refer $\boldsymbol{R}_1, \boldsymbol{R}_2$ as *redundant links*. Then the final word $\boldsymbol{w}$ is of length $n + 2\lceil \log_2 n \rceil + 8$.

It remains to present the details of Step 2 and Step 3. In Step 2, we encode an index $0 \leqslant t < n$ to a word in $\mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(2\lceil \log_2\ n \rceil ; 4)$.

Index Encoder. Let $k \in \{2,3\}$ and set $p \triangleq \lceil \log_2\ n \rceil$.

INPUT: t, where $0 \le t \le n-1$

OUTPUT: $\boldsymbol{p} \triangleq \textsc{IndexEnc}_1(t) \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(2p;4)$

(I) Let $\tau_1\tau_2\cdots\tau_p$ be the binary representation of t of length p.

(II) For $i \in [p]$, set $\pi_i = \mathtt{A}$ if $\tau_i = 0$, and set $\pi_i = \mathtt{T}$ if $\tau_i = 1$.

(III) Interleave $\pi_1\pi_2\cdots\pi_p$ with the alternating length-m sequence $\mathtt{CGCG}\cdots\mathtt{CG}$ with to obtain $\boldsymbol{p}$ of length $2m$. In other words, set $\boldsymbol{p} = \pi_1\mathtt{C}\pi_2\mathtt{G}\pi_3\mathtt{C}\pi_4\mathtt{G}\cdots$.

We illustrate the Index Encoder via an example.

Example 8 (Example 6 continued): Let $n = 10$ and so, $p = 4$. Consider the index $t = 5$ and the binary representation of length four of t is 0101. Then $\pi_1\pi_2\pi_3\pi_4 = \mathtt{ATAT}$ and so, $\boldsymbol{p} = \mathtt{ACTGACTG}$. We can verify that $\boldsymbol{p} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(8;4)$ for $k \in \{2,3\}$.

Proposition 6: The Index Encoder is correct. In other words, the word $\boldsymbol{p} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(2\lceil \log_2 n\rceil ; 4)$, where $k \in \{2,3\}$.

Proof: Set $p = \lceil \log_2\ n \rceil$ and let $\boldsymbol{p} = p_1p_2\cdots p_{2p}$. Observe that $p_i \in \{\mathtt{A},\mathtt{T}\}$ if i is odd and $p_i \in \{\mathtt{C},\mathtt{G}\}$ if i is even. Hence, $\boldsymbol{p}$ is GC-balanced. Also, for $h \in \{1,3\}$, it follows that $p_i \neq p_{i+h}$ for $1 \le i \le 2p-h$. Hence, there is no duplication of length h in $\boldsymbol{p}$. Since $p_i \neq p_{i+2}$ for all even i and $1 \leqslant i \leqslant 2p-2$, there is no duplication of length two. Therefore, $\boldsymbol{p} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant 3}(2p;4) \subset \mathrm{Irr}^{\mathcal{B}}_{\leqslant 2}(2p;4)$. ■

In Step 3, we construct the redundant links $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ so that the resultant word is both GC-balanced and $\leqslant k$-irreducible.

Proposition 7: Let $k \in \{2,3\}$. Let $\boldsymbol{X} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(n;4)$ and $\boldsymbol{Y} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(m;4)$. There exists a word $\boldsymbol{Z}$ of length four such that the concatenated word $\boldsymbol{XZY}$ belongs to $\mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(m+n+4;4)$. Furthermore, $\boldsymbol{Z}$ can be constructed in constant time.

Proof: We construct the word $\boldsymbol{Z} = z_1z_2z_3z_4$. Let the suffix $\boldsymbol{S}_3(\boldsymbol{X})$ and the prefix $\boldsymbol{P}_4(\boldsymbol{Y})$ be $x_{n-2}x_{n-1}x_n$ and $y_1y_2y_3y_4$, respectively. To ensure the resulting word is balanced, we choose z_1, z_2, z_3, and z_4 to be the four distinct elements in $\mathcal{D}$. We choose them according to the following cases,

- If $y_1 = y_4$, then
  - set z_1 to be a symbol in $\mathcal{D}\setminus\{x_{n-2}, x_{n-1}, x_n\}$,
  - set z_4 to be a symbol in $\mathcal{D}\setminus\{z_1, y_1, y_3\}$,
  - set z_3 to be a symbol in $\mathcal{D}\setminus\{z_1, z_4, y_1\}$,
  - set z_2 to be a symbol in $\mathcal{D}\setminus\{z_1, z_4, z_3\}$.
- If $y_1 \neq y_4$, then
  - set z_1 to be a symbol in $\mathcal{D}\setminus\{x_{n-2}, x_{n-1}, x_n\}$,
  - set z_4 to be a symbol in $\mathcal{D}\setminus\{z_1, y_1, y_2\}$,
  - set z_2 to be a symbol in $\mathcal{D}\setminus\{z_1, z_4, y_1\}$,
  - set z_3 to be a symbol in $\mathcal{D}\setminus\{z_1, z_4, z_2\}$.

Since $\boldsymbol{X}$ and $\boldsymbol{Y}$ are GC-balanced and z_1, z_2, z_3, z_4 are distinct symbols in $\mathcal{D}$, the concatenation $\boldsymbol{XZY}$ is also GC-balanced.

To demonstrate $\leqslant 3$-irreducibility, it suffices to consider the string $\boldsymbol{S}_5(\boldsymbol{X})\boldsymbol{Z}\boldsymbol{P}_5(\boldsymbol{Y})$ and we check that there is no substring of the form XX, $XYXY$ or $XYZXYZ$, where X, Y, Z are distinct symbols in $\mathcal{D}$. ■

Example 9: Let $\boldsymbol{X} = \mathtt{GACAT} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant 3}(5;4)$ and $\boldsymbol{Y} = \mathtt{CATCG} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant 3}(5;4)$. Notice that $y_1 = y_4 = \mathtt{C}$ and so, we set z_1 to be any symbol in $\mathcal{D}\setminus\{\mathtt{C},\mathtt{A},\mathtt{T}\} = \{\mathtt{G}\}$. So, we set z_1 to be $\mathtt{G}$. Following the rules, the bases z_4, z_3 and z_2 are set to be $\mathtt{A}$, $\mathtt{T}$ and $\mathtt{C}$, respectively. Therefore, we choose $\boldsymbol{Z} = \mathtt{GCTA}$ and verify that $\boldsymbol{XZY} = \mathtt{GACATGCTACATCG} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant 3}(14;4)$.

Remark: Now, given any pair of $\leqslant k$-irreducible words $\boldsymbol{X}$ and $\boldsymbol{Y}$, Proposition 7 states that there exists a redundant link of length four. In some cases, redundant links of length two and zero[1] exists.

However, it turns out a redundant link of length four is *necessary*. Specifically, the following example provides a pair of $\leqslant k$-irreducible words $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that no redundant links of length shorter than four exists.

Example 10: Set $k \in \{2,3\}$. Let $\boldsymbol{X} = \mathtt{CGCATA}, \boldsymbol{Y} = \mathtt{TATCGC} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(6;4)$. Note that $\boldsymbol{XY}$ is not $\leqslant k$-irreducible. Now, all GC-balanced words of length two belong to the set $\mathcal{Z} = \{\mathtt{CA},\mathtt{CT},\mathtt{GA},\mathtt{GT},\mathtt{AC},\mathtt{AG},\mathtt{TC},\mathtt{TG}\}$. It is easy to check that $\boldsymbol{XZY}$ is *not* $\leqslant k$-irreducible for all $\boldsymbol{Z} \in \mathcal{Z}$.

To conclude this section, we formally describe the linear-time encoder that maps an irreducible word to a GC-balanced irreducible word.

GC-Balanced Encoder. Let $k \in \{2,3\}$ and set $p \triangleq \lceil \log_2\ n \rceil$.

INPUT: $\boldsymbol{x} \in \mathrm{Irr}_{\leqslant k}(n;4)$

OUTPUT: $\boldsymbol{w} \triangleq \textsc{Enc}_2(\boldsymbol{x}) \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(n+2p+8;4)$

(I) In linear time, compute $t = \mathrm{Index}(\boldsymbol{x})$. Set $\boldsymbol{y} = f_t(\boldsymbol{x})$.

(II) Use Index Encoder and map the balancing index t to $\boldsymbol{z} = \textsc{IndexEnc}_1(t) \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(2p;4)$.

(III) Using Proposition 7 to construct the links $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ of length four such that

- $\boldsymbol{P}_t(\boldsymbol{y})\boldsymbol{R}_1\boldsymbol{S}_{n-t}(\boldsymbol{y}) \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(n+4;4)$,
- $\boldsymbol{P}_t(\boldsymbol{y})\boldsymbol{R}_1\boldsymbol{S}_{n-t}(\boldsymbol{y})\boldsymbol{R}_2\boldsymbol{p} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(n+2p+8;4)$.

(IV) Return $\boldsymbol{w} = \boldsymbol{P}_t(\boldsymbol{y})\boldsymbol{R}_1\boldsymbol{S}_{n-t}(\boldsymbol{y})\boldsymbol{R}_2\boldsymbol{p}$.

B. Approaching Asymptotic Information Rate

From Table I, we have that $\mathrm{rate}_{\leqslant 2}(4) \approx 0.7249$ and $\mathrm{rate}_{\leqslant 3}(4) \approx 0.7054$. In this subsection, we demonstrate that we can get close to these rates by combining the GC-balanced encoder with an appropriate choice of (m,ℓ)-finite state encoder. Specifically, consider the following encoder.

Encoder A. Let s be a positive integer and set $n = sm + 2\lceil \log_2\ sm \rceil + 8$.

INPUT: message $\boldsymbol{y} = \boldsymbol{y}_1\boldsymbol{y}_2\ldots\boldsymbol{y}_s \in \Sigma_4^{s\ell}$, where $\boldsymbol{y}_i \in \Sigma_4^{\ell}$ for $1 \le i \le s$

OUTPUT: $\boldsymbol{w} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(n;q)$

(I) Using the (ℓ,m)-finite state encoder, map $\boldsymbol{y}$ to $\boldsymbol{x} \in \mathrm{Irr}_{\leqslant k}(sm;q)$.

(II) Using the GC-balanced encoder, map $\boldsymbol{x}$ to $\boldsymbol{w} \in \mathrm{Irr}^{\mathcal{B}}_{\leqslant k}(n;q)$.

We have the following convergence rate of Encoder A.

[1] In other words, $\boldsymbol{XY}$ is also $\leqslant k$-irreducible.

Theorem 3: Let $k \in \{2,3\}$. Set $c_k = \text{rate}_{\leqslant k}(4) = \log_4 \lambda_k$. For $\epsilon > 0$, if we choose m, ℓ and s such that

$$\ell = \left\lceil \frac{(c_k - \epsilon/2)(c_k - \log_4 \kappa_k)}{\epsilon/2} \right\rceil, \tag{18}$$

$$m = \max\left\{ \left\lceil \frac{\ell - \log_4 \kappa_k}{c_k} \right\rceil, \frac{2}{\epsilon} - 1 \right\}, \tag{19}$$

$$s \geqslant 20 + 4\log_2 m, \tag{20}$$

then Encoder A has rate at least $c_k - \epsilon$.

Proof: Given any m, ℓ and s, the rate of Encoder A is $s\ell/n$.

Now, since $s \geqslant 17$, we have that $2^s \geqslant s^4$ or $s/2 \geqslant 2\log_2 s$. Also, since $s \geqslant 20 + 4\log_2 m$, we have $s/2 \geqslant 10 + 2\log_2 m$. Hence,

$$\begin{aligned} 2\lceil \log_2 sm \rceil + 8 &\leqslant 2(\log_2 s + \log_2 m + 1) + 8 \\ &= 2\log_2 s + (2\log_2 m + 10) \\ &\leqslant s. \end{aligned}$$

Also, it follows from the proof of Theorem 1 that $\ell/m \geqslant c_k - \epsilon/2$. Therefore,

$$\begin{aligned} \frac{s\ell}{n} &\geqslant \frac{s\ell}{sm+s} = \frac{\ell}{m+1} = \frac{m}{m+1}\frac{\ell}{m} \\ &\geq \left(1 - \frac{1}{m+1}\right)\left(c_k - \frac{\epsilon}{2}\right) \\ &= \left(c_k - \frac{\epsilon}{2}\right) - \frac{c_k - \epsilon/2}{m+1} \geqslant c_k - \epsilon. \end{aligned}$$

Here, the last inequality follows from the fact that $c_k - \epsilon/2 \leqslant 1$ and $1/(m+1) \leqslant \epsilon/2$. ■

Therefore, the asymptotic rate of GC-balanced irreducible words given in Corollary 4 follows directly from Theorem 3.

To complete our discussion, we define an alternative finite state encoder *Encoder B* that encodes quaternary messages into GC-balanced $\leqslant k$-irreducible words.

Specifically, for $\ell' < m'$, we consider the set of states $\mathcal{S}_{\mathcal{B}} \triangleq \text{Irr}^{\mathcal{B}}_{\leqslant k}(m', 4)$ and the set of directed edges $\mathcal{E}_{\mathcal{B}} \triangleq \{(\boldsymbol{x}, \boldsymbol{x}') : \boldsymbol{x}\boldsymbol{x}' \in \text{Irr}^{\mathcal{B}}_{\leqslant k}(2m', 4)\}$. For $\boldsymbol{x} \in \mathcal{S}_{\mathcal{B}}$, the neighbors of $\boldsymbol{x}$ is defined to be $N_{\mathcal{B}}(\boldsymbol{x}) \triangleq \{\boldsymbol{x}' : (\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{E}_{\mathcal{B}}\}$ and we set $\Delta^{\mathcal{B}}_{\leqslant k}(m', 4) \triangleq \min\{|N_{\mathcal{B}}(\boldsymbol{x})| : \boldsymbol{x} \in \mathcal{S}_{\mathcal{B}}\}$. When $\Delta^{\mathcal{B}}_{\leqslant k}(m') \geqslant 4^{\ell'}$, we can then define the edge labelling $\mathcal{L}_{\mathcal{B}} : \mathcal{E}_{\mathcal{B}} \to \Sigma_4^{\ell'} \times \Sigma_4^{m'}$ so that for all $\boldsymbol{x} \in \mathcal{S}_{\mathcal{B}}$ and $\boldsymbol{y} \in \Sigma_4^{\ell'}$, there exists a unique word $\boldsymbol{x}' \in \mathcal{S}_{\mathcal{B}}$ with $\mathcal{L}_{\mathcal{B}}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{y}, \boldsymbol{x}')$ (see Example 2 for details). We can now formally describe Encoder B.

Encoder B. Let s be a positive integer. Choose m' and ℓ' such that $\Delta^{\mathcal{B}}_{\leqslant k}(m') \geqslant 4^{\ell'}$.

INPUT: message $\boldsymbol{y} = \boldsymbol{y}_1\boldsymbol{y}_2 \ldots \boldsymbol{y}_s \in \Sigma^{s\ell'}$, where $\boldsymbol{y}_i \in \Sigma^{\ell'}$ for $1 \leq i \leq s$
OUTPUT: $\boldsymbol{x} \in \text{Irr}^{\mathcal{B}}_{\leqslant k}(sm'; 4)$

(I) Set $\boldsymbol{x}_0$ to be the first word lexicographically in $\mathcal{S} = \text{Irr}^{\mathcal{B}}_{\leqslant k}(m; 4)$.
(II) For $i \in [s]$, set $\boldsymbol{x}_i$ to be the unique word such that $\mathcal{L}_{\mathcal{B}}(\boldsymbol{x}_{i-1}, \boldsymbol{x}_i) = (\boldsymbol{y}_i, \boldsymbol{x}_i)$.
(III) The encoded GC-balanced irreducible word is $\boldsymbol{x} = \boldsymbol{x}_1\boldsymbol{x}_2 \ldots \boldsymbol{x}_s$.

We now compare the encoding rates of Encoders A and B for various subblock lengths m and m', respectively.

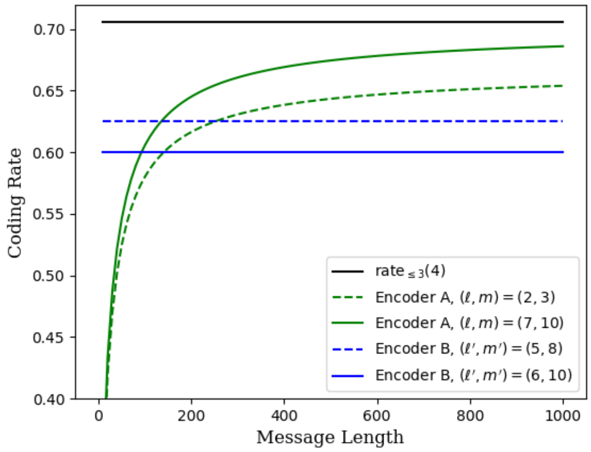

Fig. 1. Coding Rates for Encoders A and B.

Figure 1 plots these coding rates against message lengths and we observe that the coding rates of Encoder A approach the $\text{rate}_{\leqslant 3}(4)$ much faster than Encoder B.

VI. $(n, \leqslant k; q)$-TD CODES, WHERE $k \geq 4$

In this section, we look at correcting tandem duplications of length at most k, where $k \geq 4$. First, we remark that there is no known result on $(n, \leqslant k; q)$-TD codes when $k \geq 4$. Construction 1 cannot be extended for $k \geq 4$ as there are different irreducible words that share a common descendant. For example, when $q = 3, k = 4$, consider the two words $\boldsymbol{x} = 012$ and $\boldsymbol{y} = 0121012$, which are two $\leqslant 4$-irreducible words in $\text{Irr}_{\leqslant 4}(3)$. Since $\boldsymbol{x} = 012 \underset{2}{\Rightarrow} 01212 \underset{4}{\Rightarrow} 012101212$ and $\boldsymbol{y} = 0121012 \underset{2}{\Rightarrow} 012101212$, $\boldsymbol{x}$ and $\boldsymbol{y}$ share a common descendant 012101212. Furthermore, given any two words $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$ with $q \geqslant 3$, we are unaware of any algorithm that is able to decide whether $D^*_{\leqslant k}(\boldsymbol{x}) \cap D^*_{\leqslant k}(\boldsymbol{y}) = \varnothing$ in *finite time* (see [1] for a discussion).

Nevertheless, we construct $(n, \leqslant k; q)$-TD codes given $q \geqslant k+1$. To do so, we consider the following subset of $\leqslant k$-irreducible words.

Definition 3: A q-ary word $\boldsymbol{c}$ is *t-distinct* if every t consecutive positions of $\boldsymbol{c}$, or equivalently, every substring of length t, comprise t distinct symbols.

For example, $\boldsymbol{c} = 012340123$ is 5-distinct, while $\boldsymbol{x} = 0123012340123$ is not 5-distinct. Here, $\boldsymbol{c} \underset{4}{\Rightarrow} \boldsymbol{x}$. Observe that any word $\boldsymbol{c}$ that contains a $\leqslant k$-tandem duplication (i.e. not $\leqslant k$-irreducible) is necessarily not $(k+1)$-distinct. Furthermore, if $\boldsymbol{x}$ is a $\leqslant k$-descendant of $\boldsymbol{c}$, the order of the $k+1$ distinct symbols of a $(k+1)$-substring in $\boldsymbol{c}$ is preserved in $\boldsymbol{x}$.

Therefore, this motivates the following construction.

Theorem 4 (Construction 2): Given n, k and $q \geq k+1$, set $\mathcal{C} = \{\boldsymbol{c} \in \Sigma_q^n : \boldsymbol{c} \text{ is } (k+1)\text{-distinct}\}$. Then $\mathcal{C}$ is a code correcting tandem duplications of length at most k. The size of $\mathcal{C}$ is $|\mathcal{C}| = q(q-1)\cdots(q-k+1)(q-k)^{n-k}$, and hence, the asymptotic rate of the code family is $\log_q(q-k)$.

Proof: We prove the correctness of Construction 2 by providing a decoding algorithm. Suppose that $\boldsymbol{c} = c_1c_2\cdots c_n \in \mathcal{C}$ and $\boldsymbol{x}$ is a $\leqslant k$-descendant of $\boldsymbol{c}$. We now provide an efficient decoding algorithm to recover the original codeword $\boldsymbol{c}$.

Recover the first k **symbols**. Since $\boldsymbol{c}$ is $(k+1)$-distinct, the first k symbols of $\boldsymbol{c}$ are distinct. Hence, we simply search for the first k distinct symbols in $\boldsymbol{x}$. Since the order of these k distinct symbols is preserved, we recover $c_1c_2\ldots c_k$. For $i\in[k]$, let t_i be the first instance in $\boldsymbol{x}$ such that $x_{t_i}=c_i$.

Recover the next symbol. Suppose that we have recovered $c_1c_2\ldots c_i$ with $i\geqslant k$ and we have the indices t_1, t_2, …, t_i. Since $\boldsymbol{c}$ is $(k+1)$-distinct, the next symbol c_{i+1} (if it exists) is necessarily distinct from c_{i-k+1}, c_{i-k+2}, …, c_i. Hence, we set t_{i+1} to be the smallest index larger than t_i such that $x_{t_{i+1}}\notin\{c_{i-k+1},c_{i-k+2},\ldots,c_i\}$ If t_{i+1} exists and since the order of c_{i-k+1}, c_{i-k+2}, …, c_i, c_{i+1} is preserved, c_{i+1} is necessarily $x_{t_{i+1}}$. Otherwise, if such an index does not exist, we terminate the algorithm and $\boldsymbol{c}=c_1c_2\ldots c_i$. ■

Example 11: Consider $q=5$ and $k=4$. Suppose that we receive a vector $\boldsymbol{x}=0101230123412340123$.

First, we recover the first four distinct symbols 0, 1, 2, 3. So, $c_1c_2c_3c_4=0123$ with the indices $t_1=1$, $t_2=2$, $t_3=5$, and $t_4=6$.

In the next step, we search for an index t_5 greater than six such $x_{t_5}\notin\{0,1,2,3\}$. Here, $t_5=11$ and hence, $c_5=4$. Repeating the procedure, we then obtain $\boldsymbol{c}=012340123$ with $t_6=16$, $t_7=17$, $t_8=18$, and $t_9=19$.

In the next construction, we improve the rates of Construction 2. Instead of constraining the entire codeword to be $(k+1)$-distinct, we regard the codeword as a concatenation of s blocks of length k and require that each block is k-distinct. The construction is formally defined below.

Theorem 5 (Construction 3): Given s, k and $q\geq k+1$, set $n=sk$ and $\mathcal{C}'=\{\boldsymbol{c}_1\boldsymbol{c}_2\cdots\boldsymbol{c}_s$, where $\boldsymbol{c}_i$ is a k-distinct word of length k for $i\in[s]$ and the first symbol of $\boldsymbol{c}_{i+1}$ is distinct from the symbols in $\boldsymbol{c}_i$ for $i\in[s-1]\}$. Then $\mathcal{C}'$ is a code correcting tandem duplications of length at most k. The size of $\mathcal{C}'$ is

$$|\mathcal{C}'|=q(q-k)^{s-1}\left((q-1)(q-2)\cdots(q-k+1)\right)^s,$$

and hence, the asymptotic rate of the code family is $\frac{1}{k}\sum_{i=1}^{k}\log_q(q-i)$.

Proof: As before, we prove the correctness of Construction 3 by providing a decoding algorithm. Suppose that $\boldsymbol{c}=\boldsymbol{c}_1\boldsymbol{c}_2\cdots\boldsymbol{c}_s\in\mathcal{C}'$ and $\boldsymbol{x}$ is a $\leqslant k$-descendant of $\boldsymbol{c}$.

We now provide an efficient decoding algorithm to recover the original codeword $\boldsymbol{c}$. In what follows, we recover blocks from left to right. Here, we use c_{i1} to denote the first symbol of block $\boldsymbol{c}_i$.

Recover the first block $\boldsymbol{c}_1$. Similar to Construction 2, since $\boldsymbol{c}_1$ is k-distinct, the first k distinct symbols in $\boldsymbol{x}$ correspond to $\boldsymbol{c}_1$. In addition to the first block, we also look for the $(k+1)$th distinct symbol which corresponds to c_{21} by design. We set t_2 to be the corresponding index.

Recover the i**th block** $\boldsymbol{c}_i$ **for** $2\leqslant i\leqslant s-1$. Suppose that we have recovered $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, …, $\boldsymbol{c}_{i-1}$ and we have the index t_i such that $c_{i1}=x_{t_i}$. We proceed as follows.

(I) We search for the index set T such that (i) $|T|=k+1$; (ii) the indices in T are at least t_i; (iii) the indices in T are the smallest indices of $\boldsymbol{x}$ (after t_i) that correspond to distinct symbols.

(II) Let t_{i+1} be the largest index in T. If there exists index j with $t_i<j\leqslant t_{i+1}$ such that $c_{i1}=x_j$, we choose the smallest such index j and update t_i to j and repeat Step (I) to find another T.

(III) If such an index does not exist, we set $\boldsymbol{c}_i$ to be the subsequence of $\boldsymbol{x}$ restricted on the index set T. Note that t_{i+1} is set to be the largest index in T.

TABLE II

UPPER AND LOWER BOUNDS ON THE RATES OF $(n,\leqslant 4;q)$-TD CODES

q	Upper Bound of Code Rate	Rate of $\mathcal{C}'$	Rate of $\mathcal{C}$
10	0.9542	0.8701	0.7782
15	0.9745	0.9311	0.8855
20	0.9828	0.9547	0.9255
25	0.9873	0.9668	0.9458
30	0.9900	0.9741	0.9579
35	0.9918	0.9789	0.9659

Recover the last block $\boldsymbol{c}_s$. Update t_s to be largest index of $\boldsymbol{x}$ that corresponds to the symbol c_{s1}. Similar to Step (I), we find a set of k smallest indices of $\boldsymbol{x}$ (after t_s) that correspond to distinct symbols. Then set $\boldsymbol{c}_s$ to be the subsequence of $\boldsymbol{x}$ restricted on the index set T. ■

Example 12: Let $s=3$, $q=5$ and $k=4$. Each codeword $\boldsymbol{c}\in\mathcal{C}'$ is the concatenation of three blocks of length four. Suppose that we receive a word $\boldsymbol{x}=0123412343432101212 4124$.

Recovering $\boldsymbol{c}_1$. The first four distinct symbols in $\boldsymbol{x}=012341234343210124$ are highlight in red, and we get $\boldsymbol{c}_1=0123$. In addition, we have $t_2=5$ and $c_{21}=4$.

Recovering $\boldsymbol{c}_2$. Here, we start with $t_2=5$ and $c_{21}=4$

- In Step (I), we have $T=\{5,6,7,8,15\}$ with the corresponding symbols highlighted in red: 012341234343210124. Since $x_9=4$, we are unable to proceed to Step (III). So, we update $t_2=9$ and repeat Step (I).
- Repeating Step (I) with $t_2=9$, we have $T=\{9,10,13,14,15\}$ with the corresponding symbols highlighted in red: 012341234343210124. Again, we are unable to proceed as $x_{11}=4$ and so, we update $t_2=11$.
- Repeating Step (I) with $t_2=11$, we have $T=\{11,12,13,14,15\}$ with the corresponding symbols highlighted in red: 012341234343210124. Now, we proceed to Step (III). We set $\boldsymbol{c}_2=4321$ and set $t_3=15$.

Recovering $\boldsymbol{c}_3$. Since x_{15} is the last occurrence of the symbol 0, we look at the set of indices after 15. Then we have $T=\{15,16,17,18\}$ and $\boldsymbol{c}_3=0124$.

To conclude this section, we compare the rates of $\mathcal{C}$ or $\mathcal{C}'$ for the case $k=4$ and certain values of q. In addition to the rates, we also compute the corresponding upper bound. Observe that an $(n,\leqslant 4;q)$-TD code is also an $(n,\leqslant 2;q)$-TD code while the optimal size of the latter is given by $I_{\leqslant 2}(n,q)$. Hence, using Corollary 1, we compute $\mathrm{rate}_{\leq 2}(q)$ and tabulate the values as an upper bound. Table II demonstrates that $\mathcal{C}'$ is *almost* optimal for large q.

VII. CONCLUSION

For $k\in\{2,3\}$ and all $q\geq 2$, we provided an explicit recursive formula for $\mathrm{Irr}_{\leqslant k}(n,q)$ and hence, derived the expressions for $\mathrm{rate}_{\leqslant k}(q)$. In addition, we designed efficient encoders/decoders for $\mathrm{Irr}_{\leqslant k}(n,q)$.

(i) We provided an (ℓ, m)-finite state encoder and showed that for all $\epsilon > 0$, if we choose $m = \Theta(1/\epsilon)$ and $\ell = \Theta(1/\epsilon)$, the encoder achieves rate that is at least $\mathrm{rate}_{\leqslant k}(q) - \epsilon$. The implementation of the finite state encoder with a lookup table runs in $O(n/\epsilon)$ time and requires $q^{\Theta(1/\epsilon)}$ space. However, if we use the ranking/unranking method in Section IV, the encoder runs in $O(n/\epsilon^2)$ time and requires $O(1/\epsilon)$ space.

(ii) We provided an unranking algorithm for irreducible words whose encoding rate is $(1/n)\log_q(\mathrm{Irr}_{\leqslant k}(n,q)) \geqslant \mathrm{rate}_{\leqslant k}(q)$. The encoder runs in $O(n^2)$ time and requires $O(n^2)$ space.

(iii) Motivated by applications that store data in living organisms, we focused on $q = 4$ and provided a linear-time encoder that translates quaternary alphabet messages into GC-balanced codewords that can correct tandem duplications of length at most k, where $k \in \{2, 3\}$.

For $k \geq 4$, the set $\mathrm{Irr}_{\leqslant k}(n,q)$ does not form an $(n, \leqslant k; q)$-TD code. We provided two methods to construct $(n, \leqslant k; q)$-TD codes with efficient error decoders. Our constructed codes are subsets of $\mathrm{Irr}_{\leqslant k}(n,q)$ and we believe that irreducible words play an important role to design optimal $(n, \leqslant k; q)$-TD codes.

Acknowledgment

The authors would like to thank the editor and the reviewers for their constructive feedback and helpful suggestions.

References

[1] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Deciding the confusability of words under tandem repeats," *ACM Trans. Algorithms*, vol. 15, no. 3, pp. 1–22, Apr. 2018.

[2] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Efficient encoding/decoding of irreducible words for codes correcting tandem duplications," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 2406–2410.

[3] T. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, Jan. 1973.

[4] S. Jain, F. F. Hassanzadeh, M. Schwartz, and J. Bruck, "Duplication-correcting codes for data storage in the DNA of living organisms," *IEEE Trans. Inf. Theory*, vol. 63, no. 8, pp. 4996–5010, Aug. 2017.

[5] S. Jain, F. F. Hassanzadeh, M. Schwartz, and J. Bruck, "Noise and uncertainty in string-duplication systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 3120–3124.

[6] K. S. Immink and K. Cai, "Properties and constructions of constrained codes for DNA-based data storage," 2018, *arXiv:1812.06798*. [Online]. Available: https://arxiv.org/abs/1812.06798

[7] D. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 51–53, Jan. 1986.

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

[9] M. Kovačević, "Zero-error capacity of duplication channels," Feb. 2019, *arXiv:1902.06275*. [Online]. Available: http://arxiv.org/abs/1902.06275

[10] M. Kovačević, "Codes correcting all patterns of tandem-duplication errors of maximum length 3," 2019, *arXiv:1911.06561*. [Online]. Available: http://arxiv.org/abs/1911.06561

[11] E. S. Lander *et al.*, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, no. 6822, pp. 860–921, Feb. 2001.

[12] A. Lenz, N. Jünger, and A. Wachter-Zeh, "Bounds and constructions for multi-symbol duplication error correcting codes," 2018, *arXiv:1807.02874*. [Online]. Available: http://arxiv.org/abs/1807.02874

[13] N. I. Mundy and A. J. Helbig, "Origin and evolution of tandem repeats in the mitochondrial DNA control region of shrikes (Lanius spp.)," *J. Mol. Evol.*, vol. 59, no. 2, pp. 250–257, Aug. 2004.

[14] A. Nijenhuis and H. S. Wilf, *Combinatorial Algorithms: For Computers and Calculators*. Amsterdam, The Netherlands: Elsevier, 2014.

[15] L. Organick *et al.*, "Random access in large-scale DNA data storage," *Nature Biotech.*, vol. 36, no. 3, pp. 242–248, 2018.

[16] M. G. Ross *et al.*, "Characterizing and measuring bias in sequence data," *Genome Biol.*, vol. 14, no. 5, p. R51, 2013.

[17] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An introduction to coding for constrained system," Tech. Rep., Oct. 2001.

[18] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Boston, MA, USA: Cengage, 2013.

[19] E. W. Weisstein. *Lucas Sequence, From MathWorld—A Wolfram Web Resource*. Accessed: Feb. 17, 2020. [Online]. Available: http://mathworld.wolfram.com/LucasSequence.html

[20] P. Yakovchuk, "Base-stacking and base-pairing contributions into thermal stability of the DNA double helix," *Nucleic Acids Res.*, vol. 34, no. 2, pp. 564–574, Jan. 2006.

[21] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 1, Dec. 2017, Art. no. 5011.

[22] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Trans. Mol. Biol. Multi-Scale Commun.*, vol. 1, no. 3, pp. 230–248, Sep. 2015.

Yeow Meng Chee received the B.Math., M.Math., and Ph.D. degrees in computer science from the University of Waterloo, in 1988, 1989, and 1996, respectively.

He was the Head of the Division of Mathematical Sciences from 2008 to 2010, the Chair of the School of Physical and Mathematical Sciences from 2011 to 2017, and the Interim Dean of the College of Science from 2018 to 2019 at the Nanyang Technological University. He is currently a Professor of industrial systems engineering and management, and the Associate Vice President of innovation and enterprise at the National University of Singapore (NUS). He has held senior positions in public service, including the Head of Security (information infrastructure) and the Assistant Director of Internationalization at the National Computer Board, the Deputy Director of Strategic Programs at the Infocomm Development Authority (IDA), and the Program Director of Interactive Digital Media Research and Development in the Media Development Authority. He deployed South East Asia's first certification authority Netrust in 1997, and also founded the Singapore Computer Emergency Response Team (SingCERT). His research interest lies in the interplay between combinatorics and computer science, especially coding theory, external set systems, and their applications.

Dr. Chee is a Fellow and Council Member of the Institute of Combinatorics and Its Applications. He has represented Singapore in various international forums, including a member of the APEC Electronic Commerce Task Force in 1998, the ASEAN Coordinating Committee on Electronic Commerce in 1998, the Working Group on ASEAN Information Infrastructure in 1999, and the APEC Project DARE (Data Analytics Raising Employment) Advisory Board in 2017, the Secretariat of the Canada-Singapore IT Joint Council in 1998, the Co-Chair of the APEC TEL Public Key Interoperability Expert Group in 1999, and the Secretariat of the Australia–Singapore ICT Joint Council in 1999.

Johan Chrisnata received the bachelor's degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2015. He is currently pursuing the joint Ph.D. degrees in mathematics with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, and in computer science with the Department of Computer Science, Technion University, Israel. From August 2015 to August 2018, he was a Research Officer with NTU. His research interests include enumerative combinatorics and coding theory.

Han Mao Kiah received the Ph.D. degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2014. From 2014 to 2015, he was a Post-Doctoral Research Associate with the Coordinated Science Laboratory, University of Illinois at Urbana–Champaign. From 2015 to 2018, he was a Lecturer at the School of Physical and Mathematical Sciences (SPMS), NTU. He is currently an Assistant Professor at SPMS, NTU. His research interests include DNA-based data storage, coding theory, enumerative combinatorics, and combinatorial design theory.

Tuan Thanh Nguyen received the B.Sc. and Ph.D. degrees in mathematics from Nanyang Technological University (NTU), Singapore, in 2014 and 2018, respectively. He was a Research Fellow with the School of Physical and Mathematical Sciences, NTU, from August 2018 to September 2019. He is currently a Research Fellow with the Singapore University of Technology and Design. His research interests include error correction codes and constrained codes for communication systems and data storage systems, especially codes for DNA-based data storage.