Locally-Constrained de Bruijn Codes: Properties, Enumeration, Code Constructions, and Applications

Yeow Meng Chee, *Senior Member, IEEE*, Tuvi Etzion, *Fellow, IEEE*, Han Mao Kiah, *Member, IEEE*, Sagi Marcovich, *Member, IEEE*, Alexander Vardy, *Fellow, IEEE*, Van Khu Vu, *Member, IEEE*, and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*— The *de Bruijn graph*, its sequences, and their various generalizations, have found many applications in information theory, including many new ones in the last decade. In this paper, motivated by a coding problem for emerging memory technologies, a set of sequences which generalize the window property of de Bruijn sequences, on its shorter subsequences, are defined. These sequences can be also defined and viewed as constrained sequences. Hence, they will be called *locally-constrained de Bruijn sequences* and a set of such sequences will be called a *locally-constrained de Bruijn code*. Several properties and alternative definitions for such codes are examined and they are analyzed as generalized sequences in the de Bruijn graph (and its generalization) and as constrained sequences. Various enumeration techniques are used to compute the total number of sequences for any given set of parameters. A construction method of such codes from the theory of shift-register sequences is proposed. Finally, we show how these locally-constrained de Bruijn sequences and codes can be applied in constructions of codes for correcting synchronization errors in the ℓ-symbol read channel and in the racetrack memory channel. For this purpose, these codes are superior in their size to previously known codes.**

***Index Terms*— Constrained codes, de Bruijn sequences, ℓ-symbol read channel, racetrack memories.**

I. Introduction

THE de Bruijn graph of order m, G_m, was introduced in 1946 by de Bruijn [7]. His target in introducing this graph was to find a recursive method to enumerate the number of cyclic binary sequences of length 2^k such that each binary k-tuple appears as a window of length k exactly once in each sequence. It should be mentioned that in parallel also Good [33] defined the same graph and hence it is sometimes called the *de Bruijn-Good graph*. Moreover, it did not take long for de Bruijn himself to find out that his discovery was not novel. In 1894 Flye-Sainte Marie [26] has proved the enumeration result of de Bruijn without the definition of the graph. Nevertheless, the graph continues to carry the name of de Bruijn as well as the related sequences (cycles in the graph) which he enumerated. Later in 1951 van Aardenne-Ehrenfest and de Bruijn [1] generalized the enumeration result for any arbitrary alphabet of finite size σ greater than one, using a generalized graph for an alphabet Σ of size σ. Formally, the de Bruijn graph $G_{\sigma,k}$ has σ^k vertices, each one is represented by a word of length k over an alphabet Σ with σ letters. The in-degree and the out-degree of each vertex in the graph is σ. There is a directed edge from the vertex $(x_0, x_1, \ldots, x_{k-1})$ to the vertex $(y_1, y_2, \ldots, y_k)$, where $x_i, y_j \in \Sigma$, if and only if $y_i = x_i$, $1 \leq i \leq k-1$. This edge is represented by the $(k+1)$-tuple $(x_0, x_1, \ldots, x_{k-1}, y_k)$. The sequences enumerated by de Bruijn are those whose length is σ^k and each k-tuple over Σ appears exactly once in a window of k consecutive (cyclically) symbols in the sequence. Such a sequence enumerated by de Bruijn is represented by an Eulerian cycle in $G_{\sigma,k-1}$, where each k consecutive symbols represent an edge in the graph. This sequence is also a Hamiltonian cycle in $G_{\sigma,k}$, where each k consecutive symbols represent a vertex in the graph. Henceforth, we assume that the cycles are Hamiltonian, i.e., with no repeated vertices.

Throughout the years since de Bruijn introduced his graph and the related sequences, there have been many generalizations for the graph and for the sequences enumerated by de Bruijn. Such generalizations include enumeration of sequences of length ℓ, where $\ell < \sigma^k$, in $G_{\sigma,k}$ [53] or coding for two-dimensional arrays in which all the $n \times m$ sub-arrays appear as windows in exactly one position of the large array [21]. The interest in the de Bruijn graph, its sequences, and their generalizations, is due to their diverse important applications. One of the first applications of this graph was in the introduction of shift-register sequences in general and linear feedback shift registers in particular [30]. These will have an important role also in our research. Throughout the years de Bruijn sequences, the de Bruijn graph, and their generalizations, e.g., for larger dimensions, have

Manuscript received May 5, 2020; revised May 29, 2021; accepted August 20, 2021. Date of publication September 13, 2021; date of current version November 22, 2021. The work of Tuvi Etzion was supported in part by U.S.-Israel Binational Science Foundation (BSF) under Grant 2018218 and in part by Israel Science Foundation (ISF) under Grant 222/19. The work of Han Mao Kiah was supported by Singapore Ministry of Education under Grant MOE2019-T2-2-171. The work of Alexander Vardy was supported by NSF under Grant CCF-1764104. The work of Eitan Yaakobi was supported in part by BSF under Grant 2018048. An earlier version of this paper was presented in part at the 2018 and 2019 IEEE International Symposium on Information Theory. *(Corresponding author: Tuvi Etzion.)*

Yeow Meng Chee and Van Khu Vu are with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 119077 (e-mail: ymchee@nus.edu.sg; isevvk@nus.edu.sg).

Tuvi Etzion, Sagi Marcovich, and Eitan Yaakobi are with the Department of Computer Science, Technion, Haifa 3200003, Israel (e-mail: etzion@cs.technion.ac.il; sagimar@cs.technion.ac.il; yaakobi@cs.technion.ac.il).

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

Alexander Vardy is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: avardy@ucsd.edu).

Communicated by R. Dale Wesel, Associate Editor for Coding Techniques.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2021.3112300.

Digital Object Identifier 10.1109/TIT.2021.3112300

found a variety of applications. These applications include cryptography [27], [44], and in particular linear complexity of sequences, e.g., [10], [22], [24], [29], [41], interconnection networks, e.g., [4], [25], [60], [63], [65], VLSI testing, e.g., [3], [45], two-dimensional generalizations, e.g., [6], [21], [49] with applications to self-locating patterns, range-finding, data scrambling, mask configurations, robust undetectable digital watermarking of two-dimensional test images, and structured light, e.g., [38], [54], [55], [59], [62]. Some interesting modern applications are combined with biology, like the genome assembly as part of DNA sequencing, e.g., [9], [18], [39], [48], [57], [69] and coding for DNA storage, e.g., [11], [28], [42], [58]. This is a small sample of examples for the wide range of applications in which the de Bruijn graph, its sequences, and their generalizations, were used.

The current work is not different. Motivated by applications to certain coding problems for storage, such as the ℓ-symbol read channel and the racetrack memory channel, we introduce a new type of generalization for de Bruijn sequences, the *locally-constrained de Bruijn sequences*. In these sequences, a k-tuple cannot repeat within a segment starting in any of b consecutive positions. This generalization is quite natural and as the name hints, it can be viewed as a type of a constrained sequence. The goal of this paper is to study this type of sequences in all natural directions: enumeration, constructions, and applications.

Our generalization is motivated by the need to combat synchronization errors (which are shift errors known also as deletions and sticky insertions) in certain memories. These types of synchronization errors occur in some new memory technologies, mainly in racetrack memories [13], [15], and in other technologies which can be viewed as an ℓ-symbol read channel [8], [16], [67]. By using locally-constrained de Bruijn sequences to construct such codes we will be able to increase the rate of codes which correct such synchronization errors. But, we believe that locally-constrained de Bruijn sequences and codes (sets of sequences) are of interest in their own right from both practical and theoretical points of view. The newly defined sequences can be viewed as constrained codes and as such they pose some interesting problems. This is the reason that the new sequences and the related codes will be called *locally-constrained de Bruijn sequences* and *locally-constrained de Bruijn codes*, respectively.

The rest of this paper is organized as follows. In Section II we introduce some necessary concepts which are important in our study, such as the length and the period of sequences in general and of de Bruijn sequences in particular. We will also introduce some elementary concepts related to shift-register sequences. In Section III we define the new type of sequences, the locally-constrained de Bruijn sequences and their related codes. There will be a distinction between cyclic and acyclic sequences. We consider the concept of periodicity for acyclic and cyclic sequences and define the concept of forbidden patterns. The main result in this section will be a theorem which reveals that by the given definitions, three types of codes, defined differently, form exactly the same set of sequences for an appropriate set of parameters for each definition. In Section IV the number of locally-constrained de Bruijn sequences with a given set of parameters will be considered. We start with a few very simple enumeration results and continue to show that for some parameters, the rates of the defined codes approach 1, and there are other parameters with one symbol of redundancy. Most of the section will be devoted to a consideration of the codes as constrained codes. Enumeration based on the theory of constrained codes will be applied. These considerations yield also efficient encoding and decoding algorithms for the codes. In Section V, a construction based on shift-register sequences will be given. This construction will be our main construction for codes with large segments in which no repeated k-tuples appear. The next two sections are devoted to applications of locally-constrained de Bruijn sequences in storage memories. In Section VI, the application to the ℓ-symbol read channel is discussed. This application yields another application for another new type of storage channel, the racetrack channel which is considered in Section VII. Finally, we conclude in Section VIII.

II. Preliminaries

In this section we will give some necessary definitions and notations concerning cyclic and acyclic sequences, paths and cycles in the de Bruijn graph, and shift-register sequences, in particular those which are related to primitive polynomials and are known as maximum length shift-register sequences. In Section II-A we discuss the length and period of sequences as well as cyclic and acyclic sequences. Shift-register sequences are discussed in Section II-B.

A. Paths and Cycles in the de Bruijn Graph

A *sequence* $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$ over an alphabet Σ is a sequence of symbols from Σ, i.e., $s_i \in \Sigma$ for $1 \leq i \leq n$. The *length* of such a sequence is the number of its symbols n and the sequence is considered to be acyclic. A *cyclic sequence* $\boldsymbol{s} = [s_1, s_2, \ldots, s_n]$ is a sequence of length n for which the symbols can be read from any position in a sequential order, where the element s_1 follows the element s_n. This means that $\boldsymbol{s}$ can be written also as $[s_i, s_{i+1}, \ldots, s_n, s_1, \ldots, s_{i-1}]$ for each $2 \leq i \leq n$.

Definition 1: A cyclic sequence $\boldsymbol{s} = [s_1, \ldots, s_n]$, where $s_i \in \Sigma$ and $\sigma = |\Sigma|$, is called a *weak de Bruijn sequence (cycle)* of order k, if all the n windows of consecutive k symbols of $\boldsymbol{s}$ are distinct. A cyclic sequence $\boldsymbol{s} = [s_1, \ldots, s_n]$ is called a *de Bruijn sequence* of order k, if $n = \sigma^k$ and $\boldsymbol{s}$ is a weak de Bruijn sequence of order k.

The connection between a weak de Bruijn cycle of length n (as a cycle in the de Bruijn graph) and a weak de Bruijn sequence of length n in the graph is very simple. The sequence is generated from the cycle by considering the first digit of the consecutive vertices in the cycle. The cycle is generated from the sequence by considering the consecutive windows of length k in the sequence. This is the place to define some common concepts and some notation for sequences. An acyclic sequence $\boldsymbol{s} = (s_1, \ldots, s_n)$ has n digits read from the first to the last with no wrap around. If all the windows of length k,

starting at position i, $1 \leq i \leq n-k+1$, are distinct, then the sequence corresponds to a simple path in $G_{\sigma,k}$. The *period* of a cyclic sequence $\boldsymbol{s} = [s_0, s_1, \ldots, s_{n-1}]$ is the least integer p, such that $s_i = s_{i+p}$, where indices are taken modulo n, for each i, $0 \leq i \leq n-1$. It is well known that the period p divides the length of a cyclic sequence n. If $n = rp$ then the periodic sequence $\boldsymbol{s}$ is formed by a concatenation of r copies of the first p entries of $\boldsymbol{s}$. It is quite convenient to have the length n and the period p equal if possible. There is a similar definition for the period of an acyclic sequence with slightly different properties, which is given in Section III. In this paper we will assume (if possible) that for a given cyclic sequence $\boldsymbol{s} = [s_1, \ldots, s_n]$ the period of s is n. Hence, usually we won't distinguish between the length and period for cyclic sequences. We will elaborate more on this point in Section II-B. Finally, the substring (window) $(s_i, s_{i+1}, \ldots, s_j)$ will be denoted by $s[i,j]$ and this substring will be always considered as an acyclic sequence, no matter if $\boldsymbol{s}$ is cyclic or acyclic. In addition, $s[i]$ will be sometimes used instead of s_i and the set $\{1, 2, \ldots, n\}$ is denoted by $[n]$.

B. Feedback Shift Registers and Their Cycle Structure

The theory of the sequences in the de Bruijn graph cannot be separated from the theory of shift-register sequences developed mainly by Golomb [30]. This theory, developed fifty years ago, was very influential in various applications related to digital communication [31], [32]. A short summary on the theory of shift-registers taken from [30] which is related to our work, is given next.

A *characteristic polynomial* (for a linear feedback shift register defined in the sequel) $c(x)$ of degree k over $\mathbb{F}_q$, the finite field with q elements, is a polynomial given by

$$c(x) = 1 - \sum_{i=1}^{k} c_i x^i,$$

where $c_i \in \mathbb{F}_q$. For such a polynomial a function f on k variables from $\mathbb{F}_q$ is defined by

$$f(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{k} c_i x_{k+1-i}.$$

For the *feedback function* $f(x_1, x_2, \ldots, x_k)$ we define a *state diagram* with the set of vertices

$$Q^k \triangleq \{(x_1, x_2, \ldots, x_k) \ : \ x_i \in \mathbb{F}_q\}.$$

If $x_{k+1} = f(x_1, x_2, \ldots, x_k)$ then an edge from $(x_1, x_2, \ldots, x_k)$ to $(x_2, \ldots, x_k, x_{k+1})$ is defined for the related state diagram. This implies that the feedback function defines a mapping from Q^k into Q^k. A *feedback shift register* of length k has q^k states corresponding to the set Q^k of all q^k k-tuples over $\mathbb{F}_q$. The feedback function can be a linear function or a nonlinear function on k variables. The shift register is called *nonsingular* if its state diagram consists of disjoint cycles. Any such state diagram is called a *factor* in $G_{q,k}$, where a factor in a graph is a set of vertex-disjoint cycles which contains all the vertices of the graph. There is a one-to-one correspondence between the set of factors in the de Bruijn graph $G_{q,k}$ and the set of state diagrams for the nonsingular shift registers of order k over $\mathbb{F}_q$. It is well known [30] that a binary feedback shift-register is nonsingular if and only if its feedback function has the form

$$f(x_1, x_2, \ldots, x_k) = x_1 + g(x_2, \ldots, x_k),$$

where $g(x_2, \ldots, x_k)$ is any binary function on $k-1$ variables. A similar representation also exists for nonsingular feedback shift-registers over $\mathbb{F}_q$.

A Hamiltonian cycle in $G_{q,k}$ is a de Bruijn cycle which forms a de Bruijn sequence. There are $(q!)^{q^{k-1}}/q^k$ distinct such cycles in $G_{q,k}$ and there are many methods to generate such cycles [23], [27]. One important class of sequences in the graph, related to de Bruijn sequences, are the so called m-sequences, or *maximal length linear shift-register sequences*. A shift-register is called *linear* if its feedback function $f(x_1, x_2, \ldots, x_k)$ is linear. An *m-sequence* is a sequence of length $q^k - 1$ generated by a linear shift-register associated with a primitive polynomial of degree k over $\mathbb{F}_q$. Each primitive polynomial is associated with such a sequence. In such a sequence all windows of length k are distinct and the only one which does not appear is the all-zero window. The following theorem is well known [30].

Theorem 1: The number of distinct m-sequences of order k over $\mathbb{F}_q$ (the same as the number of primitive polynomials of order k over $\mathbb{F}_q$) is

$$\frac{\phi(q^k - 1)}{k},$$

where ϕ is the Euler function.

The *exponent* of a polynomial $f(x)$ is the smallest integer e such that $f(x)$ divides $x^e - 1$. The length of the longest cycle of the state diagram formed by the shift-register associated with the characteristic polynomial $f(x)$ is the exponent of $f(x)$. The length of the cycles associated with a multiplication of several irreducible polynomials can be derived using the exponents of these polynomials and some related algebraic and combinatorial methods. This theory well-documented in [30] leads immediately to the following important result.

Theorem 2: The state diagram associated with a multiplication of two distinct primitive characteristic polynomials $f(x)$ and $g(x)$ of order k over $\mathbb{F}_q$ contains $q^k + 1$ cycles of length $q^k - 1$ and the all-zero cycle. Each possible $(2k)$-tuple over $\mathbb{F}_q$ appears exactly once as a window in one of these cycles. The m-sequences related to $f(x)$ and $g(x)$ are two of these cycles.

III. Locally-Constrained de Bruijn Codes, Periods, and Forbidden Patterns

In this section we give the formal definitions for locally-constrained de Bruijn sequences and codes. We present a definition for a family of sequences with a certain period and a definition for a family of sequences avoiding certain substrings. Three different definitions will be given and it will be proved that with the appropriate parameters the related three families of sequences contain the same sequences. Each definition will have later some role in the enumeration of

locally-constrained de Bruijn sequences which will be given in Section IV.

A. Locally-Constrained de Bruijn Codes

Definition 2:

- A sequence $\boldsymbol{s} = (s_1, \ldots, s_n)$, over some alphabet Σ, is called a (b,k)-***locally-constrained de Bruijn sequence*** if $\boldsymbol{s}[i, i+k-1] \neq \boldsymbol{s}[j, j+k-1]$ for all $i, j \in [n-k+1]$ such that $0 < |i-j| \leq b-1$. In other words, in each substring of $\boldsymbol{s}$ whose length is $b+k-1$ there is no repeated k-tuple, i.e., each subset of b consecutive windows of length k in $\boldsymbol{s}$ contains b distinct k-tuples.
- A set of distinct (b,k)-locally-constrained de Bruijn sequences of length n is called a (b,k)-***locally-constrained de Bruijn code***. The set of all (b,k)-locally-constrained de Bruijn sequences of length n will be denoted by $\mathbb{C}_{DB}(n,b,k)$. The alphabet Σ and its size σ should be understood from the context throught our discussion.
- A cyclic sequence $\boldsymbol{s} = [s_1, \ldots, s_n]$, over Σ, is called a ***cyclic*** (b,k)-***locally-constrained de Bruijn sequence*** if $(s_1, \ldots, s_n)$ is a (b,k)-locally-constrained de Bruijn sequence and for each cyclic shift of $\boldsymbol{s}$, $[s'_1, \ldots, s'_n]$, $(s'_1, \ldots, s'_n)$ is a (b,k)-locally-constrained de Bruijn sequence. In other words, in each substring of $\boldsymbol{s}$ whose length is $b+k-1$ there is no repeated k-tuple, where a substring which starts in one of the last $b+k-2$ bits of $\boldsymbol{s}$ continues with bits from the beginning of $\boldsymbol{s}$. Note, that two sequences which differ only in a cyclic shift are considered to be the same sequence.
- A set of distinct cyclic (b,k)-locally-constrained de Bruijn sequences of length n is called a ***cyclic*** (b,k)-***locally-constrained de Bruijn code***. The set of all cyclic (b,k)-locally-constrained de Bruijn sequences of length n will be denoted by $\mathbb{C}^*_{DB}(n,b,k)$. We note that by the definition of a cyclic locally-constrained de Bruijn sequence two codewords in a cyclic (b,k)-locally-constrained de Bruijn code cannot differ only in a cyclic shift, but in some applications one might consider these cyclic shifts of codewords as distinct codewords.

Example 1:

- The sequence $\boldsymbol{s}_1 = (0,0,1,1,0,1,0) \in \{0,1\}^7$ is a $(3,3)$-locally-constrained de Bruijn sequence since in each substring of $\boldsymbol{s}_1$ of length 5, all three 3-tuples are distinct. For example, in the substring, $(0,0,1,1,0)$, all three 3-tuples, $(0,0,1), (0,1,1), (1,1,0)$, are distinct.
- The sequence $\boldsymbol{s}_2 = (0,0,0,0,1,1,1) \in \{0,1\}^7$ is not a $(3,3)$-locally-constrained de Bruijn sequence since the first two 3-tuples are the same, $(0,0,0)$.
- The cyclic sequence $\boldsymbol{s}_1 = [0,0,1,1,0,1,0]$ is a cyclic $(3,3)$-locally-constrained de Bruijn sequence since we can check to see that each cyclic shift of $\boldsymbol{s}_1$ is also a $(3,3)$-locally-constrained de Bruijn sequence.

de Bruijn sequences form a special case of locally-constrained de Bruijn sequences as asserted in the following theorem which is readily verified by definition.

Theorem 3:

- The cyclic sequence $\boldsymbol{s}$ of length q^k over $\mathbb{F}_q$ is a de Bruijn sequence if and only if $\boldsymbol{s}$ is a cyclic (q^k, k)-locally-constrained de Bruijn sequence.
- The acyclic sequence $\boldsymbol{s}$ of length $q^k + k - 1$ over $\mathbb{F}_q$ is a de Bruijn sequence if and only if $\boldsymbol{s}$ is a (q^k, k)-locally-constrained de Bruijn sequence.

Theorem 3 introduce the connection between de Bruijn sequences and locally-constrained de Bruijn sequences. But, there are a few main differences between de Bruijn sequences of order k and (b,k)-locally-constrained de Bruijn sequences.

- A de Bruijn sequence is usually considered and is used as a cyclic sequence while a locally-constrained de Bruijn sequence will be generally used as an acyclic sequence.
- A de Bruijn sequence contains each possible k-tuple exactly once as a window of length k, while in a (b,k)-locally-constrained de Bruijn sequence, each possible k-tuple can be repeated several times or might not appear at all.
- A de Bruijn sequence contains each possible k-tuple exactly once in one period of the sequence, while in a (b,k)-locally-constrained de Bruijn sequence, each possible k-tuple can appear at most once among any b consecutive k-tuples in the sequence.
- The length (or equivalently period in this case) of a de Bruijn sequence is strictly q^k, while there is no constraint on the length and the period of a (b,k)-locally-constrained de Bruijn sequence.

These differences between de Bruijn sequences and locally-constrained de Bruijn sequences are important in the characterization, enumeration, constructions, and applications of the two types of sequences.

B. Periods and Forbidden Subsequences

There are two concepts which are closely related to locally-constrained de Bruijn sequences, the *period* of an acyclic sequence and *avoided* patterns. Let $\boldsymbol{u}$ and $\boldsymbol{s}$ be two sequences over Σ. The sequence $\boldsymbol{s}$ *avoids* $\boldsymbol{u}$ (or $\boldsymbol{s}$ is $\boldsymbol{u}$*-avoiding*) if $\boldsymbol{u}$ is not a substring of $\boldsymbol{s}$. Let $\mathcal{F}$ be a finite set of sequences over Σ and $\boldsymbol{s}$ a sequence over Σ. The sequence $\boldsymbol{s}$ *avoids* $\mathcal{F}$ (or $\boldsymbol{s}$ is $\mathcal{F}$*-avoiding*) if no sequence in $\mathcal{F}$ is a substring of $\boldsymbol{s}$. Let $\mathcal{A}(n;\mathcal{F})$ denote the set of all σ-ary sequences of length n which avoid $\mathcal{F}$. A subset of $\mathcal{A}(n;\mathcal{F})$, i.e., a set of $\mathcal{F}$-avoiding sequences of length n, is called an $\mathcal{F}$*-avoiding code* of length n.

The second concept is the *period* of a sequence. For cyclic sequences the length and the period of the sequence either coincide or have a very strict relation, where the period of the sequence divides its length. This is not the case for acyclic sequences.

Definition 3:

- A sequence $\boldsymbol{s} = (s_1, s_2, \ldots, s_n) \in \Sigma^n$ is a ***period-***p ***sequence*** if it satisfies $s_i = s_{i+p}$ for all $1 \leq i \leq n-p$. Note, that the definition for the period of a cyclic sequence coincides with this definition.
- A sequence $\boldsymbol{s} \in \Sigma^n$ is called an m***-limited period-***p ***substrings*** if any period-p substring of $\boldsymbol{s}$ has length at most m.

- A set of m-limited period-p substrings sequences from Σ^n is called a ***m-limited period-p substrings code***. The set of all such m-limited period-p substrings sequences is denoted by $\mathbb{C}_{LP}(n, m, p)$.

Example 2:

- The sequence $\boldsymbol{s}_3 = (0,1,0,1,0,1,0,1) \in \{0,1\}^8$ is a period-2 sequence of length 8.
- The sequence $\boldsymbol{s}_4 = (0,0,1,0,1,0,0,1) \in \{0,1\}^8$ is a 5-limited period-2 substrings sequence since the longest substring with period-2 of $\boldsymbol{s}_2$, $(0,1,0,1,0)$, is of length 5.
- The sequence $\boldsymbol{s}_5 = (0,0,1,0,1,0,1,1) \in \{0,1\}^8$ is not a 5-limited period-2 substrings sequence since the longest substring with period-2 of $\boldsymbol{s}_5$, $(0,1,0,1,0,1)$, is of length $6 > 5$.

The first lemma is a straightforward observation.

Lemma 1: If $\mathcal{F}$ is the set of all sequences of length $m+1$ and period-p, then for each i, $1 \leq i \leq m$, the set $\mathbb{C}_{LP}(n, i, p)$ is an $\mathcal{F}$-avoiding code.

Proof: Recall, that for each i, the set of sequences in $\mathbb{C}_{LP}(n, i, p)$, have length n. If $\boldsymbol{s}$ is a sequence in $\mathbb{C}_{LP}(n, i, p)$, $1 \leq i \leq m$, then each substring of $\boldsymbol{s}$ with period-p has length at most i, where i is a positive integer strictly smaller than $m+1$. The set $\mathcal{F}$ contains the sequences of length $m+1$ and period-p and hence a sequence of $\mathcal{F}$ cannot be a substring of $\boldsymbol{s}$. Thus, $\mathbb{C}_{LP}(n, i, p)$ is an $\mathcal{F}$-avoiding code. □

C. Equivalence Between the Three Types of Codes

Let $\mathcal{F}_{p,p+k}$ be the set of all period-p sequences of length $p+k$ for any given $1 \leq p \leq b-1$ and let $\mathcal{F}^{b,k} = \cup_{p=1}^{b-1} \mathcal{F}_{p,p+k}$.

Example 3: When $b = k = 3$, we have

$$\mathcal{F}^{3,3} = \mathcal{F}_{1,4} \cup \mathcal{F}_{2,5} = \{(0,0,0,0), (1,1,1,1), (0,0,0,0,0), (0,1,0,1,0), (1,0,1,0,1), (1,1,1,1,1)\}.$$

We observe that the $(3,3)$-locally-constrained de Bruijn sequence is also $\mathcal{F}^{3,3}$-avoiding.

The following result implies a strong relation between locally-constrained de Bruijn codes, codes of limited period p substrings, and $\mathcal{F}$-avoiding codes. By the related definitions of these concepts we have the following theorem.

Theorem 4: For all given admissible $n,\ b,\ k$, and any code $\mathbb{C} \subset \Sigma^n$,

$$\mathcal{A}(n; \mathcal{F}^{b,k}) = \mathbb{C}_{dB}(n, b, k) = \bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i).$$

Proof: We will prove that $\mathcal{A}(n; \mathcal{F}^{b,k}) \subseteq \mathbb{C}_{DB}(n, b, k)$, $\mathbb{C}_{dB}(n, b, k) \subseteq \bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)$, and $\bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i) \subseteq \mathcal{A}(n; \mathcal{F}^{b,k})$. These three containment proofs will imply the claim of the theorem.

First, we prove that if $\boldsymbol{c}$ is a sequence in $\mathcal{A}(n; \mathcal{F}^{b,k})$ then $\boldsymbol{c}$ is a sequence in $\mathbb{C}_{DB}(n, b, k)$. Let $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ be any sequence in $\mathcal{A}(n, \mathcal{F}^{b,k})$, and assume to the contrary that $\boldsymbol{c} \notin \mathbb{C}_{DB}(n, b, k)$. This implies that there exist integers $i, j \in [n-k+1]$ such that $1 \leq p = j - i \leq b-1$ and $\boldsymbol{c}[i, i+k-1] = \boldsymbol{c}[j, j+k-1]$. Hence, $\boldsymbol{c}[i, j+k-1] = (c_i, c_{i+1}, \ldots, c_{j+k-1})$ is a substring with period-p and length $p+k$. Therefore $\boldsymbol{c}[i, j+k-1] \in \mathcal{F}^{b,k}$, a contradiction since $\boldsymbol{c} \in \mathcal{A}(n; \mathcal{F}^{b,k})$. Thus, $\boldsymbol{c} \in \mathbb{C}_{DB}(n, b, k)$ which implies that $\mathcal{A}(n; \mathcal{F}^{b,k}) \subseteq \mathbb{C}_{DB}(n, b, k)$.

Next, we prove that if $\boldsymbol{c}$ is a sequence in $\mathbb{C}_{DB}(n, b, k)$ then $\boldsymbol{c}$ is a sequence of $\bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)$. Let $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ be any sequence in $\mathbb{C}_{DB}(n, b, k)$, and assume to the contrary that $\boldsymbol{c} \notin \bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)$. Hence, there exists $p \in [b-1]$ such that $\boldsymbol{c} \notin \mathbb{C}_{LP}(n, p+k-1, p)$. Therefore, $\boldsymbol{c}$ contains a period-p substring of length $p+k$. Let $\boldsymbol{c}[i, i+p+k-1]$ be such a period-p substring. Hence, $\boldsymbol{c}[i, i+k-1] = \boldsymbol{c}[i+p, i+p+k-1]$, a contradiction since $\boldsymbol{c} \in \mathbb{C}_{DB}(n, b, k)$. Thus, $\mathbb{C} \subseteq \bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)$ which implies that $\mathbb{C}_{DB}(n, b, k) \subseteq \bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)$.

Finally, if $\boldsymbol{c}$ is a sequence in $\bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)$, then by Lemma 1 we have that $\boldsymbol{c}$ is a sequence in $\mathcal{A}(n; \mathcal{F}^{b,k})$. Thus, $\bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i) \subseteq \mathcal{A}(n; \mathcal{F}^{b,k})$. □

Corollary 1: For all given admissible n, b, k, and any code $\mathbb{C} \subset \Sigma^n$, the following three statements are equivalent

1) $\mathbb{C}$ is a subset of $\mathcal{A}(n; \mathcal{F}^{b,k})$.
2) $\mathbb{C}$ is a subset of $\mathbb{C}_{DB}(n, b, k)$.
3) $\mathbb{C}$ is a subset of $\bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)$.

The set $\mathcal{F}$ which contains **all** the forbidden patterns of an $\mathcal{F}$-avoiding code $\mathcal{C}$ is never a minimum size set of forbidden patterns, i.e., there exists another set $\mathcal{F}'$ such that $|\mathcal{F}'| < |\mathcal{F}|$ and $\mathcal{C}$ is also an $\mathcal{F}'$-avoiding code. This set $\mathcal{F}'$ contains shorter patterns than the ones which are contained in $\mathcal{F}$. For example, if $\mathcal{F} = \{00, 01\}$ then $\mathcal{F}' = \{0\}$ is a smaller size set which for which $\mathcal{C}$ is an $\mathcal{F}$-avoiding code if and only if $\mathcal{C}$ is an $\mathcal{F}'$-avoiding code. This implies that there always exist two distinct sets, $\mathcal{F}_1$ and $\mathcal{F}_2$, where $|\mathcal{F}_1| < |\mathcal{F}_2|$, such that $\mathcal{A}(n; \mathcal{F}_1) = \mathcal{A}(n; \mathcal{F}_2)$. But, there always exists one such set $\mathcal{F}$ of minimum size. To see that, let $\mathcal{F}$ be a set of forbidden sequences over Σ. As long as $\mathcal{F}$ contains σ sequences of length $\ell+1$ which form the set $S = \{\boldsymbol{u}\alpha \ : \ \alpha \in \Sigma\}$, where $\boldsymbol{u}$ is a string of length ℓ, these σ sequences of $\mathcal{F}$ which are contained in S can be replaced by the sequence $\boldsymbol{u}$. When this process comes to its end, instead of the original set $\mathcal{F}$ of forbidden patterns we have a set $\mathcal{F}'$ of forbidden patterns for the same code. The set of sequences $\mathcal{F}'$ will be called the *forbidden reduced set* of $\mathcal{F}$.

Lemma 2: If $\mathcal{F}$ is a set of forbidden sequences and $\mathcal{F}'$ is its forbidden reduced set, then $\mathcal{A}(n; \mathcal{F}) = \mathcal{A}(n; \mathcal{F}')$.

Proof: The proof is an immediate observation from the fact that all the sequences of length n which do not contain the patterns in $S = \{\boldsymbol{u}\alpha \ : \ \alpha \in \Sigma\}$, where n is greater than the length of $\boldsymbol{u}$, as substrings do not contain the pattern $\boldsymbol{u}$ as a substring. Hence, $\mathcal{A}(n; \mathcal{F}) \subseteq \mathcal{A}(n; \mathcal{F}')$.

Clearly, all the sequences of length n which do not contain $\boldsymbol{u}$ as a substring do not contain any pattern from S as a substring. Hence, $\mathcal{A}(n; \mathcal{F}') \subseteq \mathcal{A}(n; \mathcal{F})$.

Thus, $\mathcal{A}(n; \mathcal{F}) = \mathcal{A}(n; \mathcal{F}')$. □

IV. Enumeration of Locally-Constrained de Bruijn Sequences

In this section we consider enumeration of the number of sequences in $\mathbb{C}_{DB}(n, b, k)$. Note, that we are considering only acyclic sequences. A σ-ary code $\mathbb{C}$ of length n is a

set of σ-ary sequences of length n, that is $\mathbb{C} \subseteq \Sigma^n$. For each code $\mathbb{C}$ of length n, we define the rate of the code $\mathbb{C}$ to be $R(\mathbb{C}) = \log_\sigma(|\mathbb{C}|)/n$, and the redundancy of the code $\mathbb{C}$ to be $r(\mathbb{C}) = n - \log_\sigma(|\mathbb{C}|)$, where $|\mathbb{C}|$ is the size of the code $\mathbb{C}$. We define the *maximum asymptotic rate* of (b,k)-locally-constrained de Bruijn codes to be $R_{DB}(b,k) = \limsup_{n\to\infty} \frac{\log_\sigma |\mathbb{C}_{DB}(n,b,k)|}{n}$.[1]

A. Trivial Bounds

In this section trivial bounds on the number of locally-constrained de Bruijn sequences are considered. The number of (cyclic) de Bruijn sequences of length σ^n over an alphabet of size σ is $(\sigma!)^{\sigma^{n-1}}/\sigma^n$ [1] which implies the following simple result (for acyclic sequences).

Theorem 5: For any given positive integer $\sigma \geq 2$, positive integer k, and $n \geq \sigma^k + k - 1$,

$$|\mathbb{C}_{DB}(n, \sigma^k, k)| = (\sigma!)^{\sigma^{k-1}}.$$

Corollary 2: For any given positive integer $\sigma \geq 2$ and a positive integer k,

$$R_{DB}(\sigma^k, k) = 0.$$

Corollary 2 can be generalized for $R_{DB}(b,k)$, where $\sigma^k \geq b \geq b_0$, where b_0 is an integer whose value depends on σ and k. The reason that the rates are zero for so many values is that once we have a long simple path of length b_0 (where the smallest value of b_0 has to be determined) in $G_{\sigma,k}$ (and the number of such paths is very large [53]), to continue the path for a long sequence which is a (b,k)-locally-constrained de Bruijn sequence, the sequence will be almost periodic (with a possibility of some small local changes). In the case of Theorem 5, where the path is a de Bruijn sequence, there is only one way to continue the path without violating the constraint. There are some intriguing questions in this context. The first question is to be specific in the value of b_0. Another question is to find a good bound on $R_{DB}(b,k)$, where b is large compared to k and $R_{DB}(b,k) > 0$. Such a bound will be given in Section V, but we have no indication how good it is. The other extreme case is when $b = 1$ and hence the sequence is not constrained and we have the following trivial result.

Theorem 6: For any given positive integer $\sigma \geq 2$, a positive integer k, and $n \geq k$, $|\mathbb{C}_{DB}(n,1,k)| = \sigma^n$ and $R_{DB}(1,k) = 1$.

Except for these cases, to find the rates of locally-constrained de Bruijn sequences, where $b > 1$, is not an easy task. When the rate is approaching one, we are interested in the redundancy of $\mathbb{C}_{DB}(n,b,k)$. Unfortunately, finding the redundancy of $\mathbb{C}_{DB}(n,b,k)$ is even more difficult than to find the rate. Fortunately, for small values of b we can use the theory of constrained coding, and for k large enough we can even show that the redundancy is one symbol.

The last trivial case is the $(b,1)$-locally-constrained de Bruijn codes. The sequences of such a code will be used for error correction of synchronization errors in racetrack memories in Section VII-A. Since this constraint implies that any b consecutive elements in the sequence will be distinct, we must have that $\sigma \geq b$ to obtain any valid sequence. If $\sigma \geq b$, then we have to choose b elements from the σ alphabet letters to start the sequence and in any other position we can choose any of the alphabet letters, except for the previous $b-1$ positions in the sequence. Hence, we have

Theorem 7: For any $b \geq 1$ and any alphabet of size σ we have

$$|\mathbb{C}_{DB}(n,b,1)| = \begin{cases} 0 & \text{if } b > \sigma \\ \binom{\sigma}{b} b!(\sigma - b + 1)^{n-b} & \text{if } \sigma \geq b. \end{cases}$$

Corollary 3: For any $b \geq 1$ and any alphabet σ we have that $R_{DB}(b,1) = 0$ if $\sigma \leq b$ and $R_{DB}(b,1) = \log_\sigma(\sigma - b + 1)$ if $\sigma > b$.

B. (b,k)-Locally-Constrained de Bruijn Codes With Redundancy 1

When b is fixed and k tends to infinity the number of forbidden patterns can be neglected compared with the total number of patterns. This will lead to asymptotic rate $R_{DB}(b,k)$ getting close to 1. This can be formally seen for example from Theorem 8 which provides a stronger statement on a range for which the redundancy is only one bit. Therefore, in this case of fixed b and k tending to infinity we are interested in the redundancy of the code.

For a subset A of a set U, the complement of A, A^c, is the subset which consists of all the elements of U which are not contained in A. For a code $\mathbb{C}$ of length n over Σ, $\mathbb{C}^c = \Sigma^n \setminus \mathbb{C}$. The following simple lemma will be used in the next theorem.

Lemma 3: If A_i, $1 \leq i \leq m$, are m sets over the same domain then

$$|(\cap_{i=1}^m A_i)^c| \leq \sum_{i=1}^m |A_i^c|.$$

Proof: It is well known from set theory that

$$(\cap_{i=1}^m A_i)^c = \cup_{i=1}^m A_i^c.$$

Combining this with the trivial assertion that for any two sets A and B, $|A \cup B| \leq |A| + |B|$ we have that

$$|\,(\cap_{i=1}^m A_i)^c\,| = |\,\cup_{i=1}^m A_i^c| \leq \sum_{i=1}^m |A_i^c|.$$

□

Theorem 8: For all σ, n and $b < k$,

$$|\mathbb{C}_{DB}(n,b,k)| \geq \sigma^n \left(1 - (b-1)n \cdot \left(\frac{1}{\sigma}\right)^k\right).$$

In particular, for $k \geq \lceil \log_\sigma n + \log_\sigma(b-1) \rceil + 1$, the redundancy of $\mathbb{C}_{DB}(n,b,k)$ is at most a single symbol.

Proof: By Theorem 4, $\mathbb{C}_{DB}(n,b,k) = \bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)$. Combining this fact with Lemma 3 we have that

$$\begin{aligned} |\mathbb{C}_{DB}(n,b,k)| &= \sigma^n - |\mathbb{C}_{DB}(n,b,k)^c| \\ &= \sigma^n - |\left(\cap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+k-1, i)\right)^c| \\ &\geq \sigma^n - \sum_{i=1}^{b-1} |\mathbb{C}_{LP}(n, i+k-1, i)^c|. \end{aligned}$$

[1] The lim sup can indeed be replaced by a proper lim [52].

For each i, $1 \leq i \leq b-1$, a word $\boldsymbol{c}$ is contained in $\mathbb{C}_{LP}(n, i+k-1, i)^c$ if and only if it has length n and a substring of period-i whose length is at least $i+k$. There are at most $n-(i+k-1)$ possible starting positions for such a substring. Once the first i symbols in this substring, of length $i+k$, are known, its other k symbols are uniquely determined. The remaining $n-(i+k)$ symbols in the word of length n can be chosen arbitrarily (note that some choices cause other substrings of the same period and larger length in this word. They might cause other substrings with larger period than i, so the computation which follows has many repetitions and some sequences which do not have to be computed.). Hence, the number of words in $\mathbb{C}_{LP}(n, i+k-1, i)^c$ is upper bounded by

$$|\left(\mathbb{C}_{LP}(n, i+k-1, i)\right)^c| \leq (n-i-k+1) \cdot \sigma^i \cdot \sigma^{n-(i+k)} < \sigma^n \cdot n \cdot \left(\frac{1}{\sigma}\right)^k.$$

Therefore,

$$\sum_{i=1}^{b-1} |\left(\mathbb{C}_{LP}(n, i+k-1, i)\right)^c| \leq (b-1) \cdot \sigma^n \cdot n \cdot \left(\frac{1}{\sigma}\right)^k,$$

and hence,

$$|\mathbb{C}_{DB}(n,b,k)| \geq \sigma^n - (b-1) \cdot \sigma^n \cdot n \cdot \left(\frac{1}{\sigma}\right)^k = \sigma^n \left(1 - (b-1) \cdot n \cdot \left(\frac{1}{\sigma}\right)^k\right).$$

In particular, for $k \geq \lceil \log_\sigma(n) + \log_\sigma(b-1) \rceil + 1$, we obtain

$$|\mathbb{C}_{DB}(n,b,k)| \geq \sigma^n \left(1 - (b-1) \cdot n \cdot \left(\frac{1}{\sigma}\right)^{\log_\sigma n + \log_\sigma(b-1)+1}\right) \geq \sigma^n \cdot \frac{\sigma-1}{\sigma} \geq (\sigma-1) \cdot \sigma^{n-1}.$$

Thus, the redundancy of $\mathbb{C}_{DB}(n,b,k)$ is at most a single symbol. □

A weaker bound than the one in Theorem 8 for $\sigma = 2$ was given in [15] (Theorem 13). Finally, to encode the (b,k)-locally-constrained de Bruijn code efficiently with only a single symbol of redundancy, we may use *sequence replacement techniques* [66].

C. Representation as a Graph Problem

The key for enumeration of the number of (b,k)-locally-constrained de Bruijn sequences of length n is a representation of the enumeration as a graph problem. The de Bruijn graph $G_{\sigma,k}$ was the key for the enumeration of the cyclic and acyclic de Bruijn sequences length σ^k. This in turn gives the exact number of (σ^k, k)-locally-constrained de Bruijn sequences of length $n \geq \sigma^k + k - 1$. This number is also equal to the number of Hamiltonian cycles in the graph and also the number of Eulerian cycles in $G_{\sigma,k-1}$. There are a few ways to compute this number, one way for example was to show that this number is equal to the number of spanning trees in the graph [27]. The paths in $G_{\sigma,k}$, which are not necessarily simple, where vertices are considered as the k-tuples, represent the $(1,k)$-locally-constrained de Bruijn sequences. These sequences have no constraints. Can the de Bruijn graph or a slight modification of it represent other (b,k)-locally-constraints? The answer is yes. For this we have to apply a state-splitting algorithm [52] on the de Bruijn graph. The main idea of the method is to eliminate cycles of length $b-1$ or less by splitting vertices and edges to a few vertices and edges. This method, which is done in constrained coding, will be related to the code $\mathbb{C}_{DB}(n,b,k)$.

A much simpler representation is related to the code $\mathcal{A}(n;\mathcal{F})$, where $\mathcal{F}$ is the forbidden reduced set of all the substrings which are forbidden in a (b,k)-locally-constrained sequence. This representation is also using the theory of constrained coding [52]. One has to construct the state diagram of the constrained system. From the state diagram one has to generate the related adjacency matrix to compute the rate of the locally-constrained de Bruijn code by computing the largest eigenvalue of the adjacency matrix. If λ is the largest eigenvalue of the adjacency matrix then asymptotically there are λ^n sequences and the rate of the code is $\log_\sigma \lambda$. An example will be given in the next subsection. By Theorem 4 the two codes are equal, so the simpler representation with the state diagram of the constrained system is preferred. Using a state diagram, each vertex in the state diagram of the constrained system is labelled by a substring, which is not in $\mathcal{F}$. Moreover, each vertex is labelled by a substring whose length is at most $b+k-2$ which can be completed to a forbidden substring of length $b+k-1$. Each vertex, labelled by a substring s, has an out-degree at most σ. If α is a symbol in Σ and $s\alpha$ is not a forbidden substring in $\mathcal{F}$, then there is an edge, labelled by α, from v to a vertex u. The vertex u is labelled by the longest suffix of $s\alpha$ which can be completed to a forbidden substring of length $b+k-1$. There is also an initial vertex x which represents the empty substring. Each one of the σ out-edges exists in a vertex v if it does not lead to a vertex labelled by a forbidden substring in $\mathcal{F}$. To find the rate of the code more efficiently, one can use a reduction of the state diagram [52]. An example of the state diagram for the $(3,3)$-locally-constrained de Bruijn sequences via the forbidden patterns in the set {0000, 1111, 01010, 10101} is depicted in Figure 1. Note, this set of forbidden pattern is equivalent to the set $\mathcal{F}^{3,3}$ introduced in Example 3.

D. State Diagrams and Rates Based on the Forbidden Subsequences

As a sequence of a constrained code, the (b,k)-locally-constrained de Bruijn sequence has several forbidden patterns, $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_\ell$. W.l.o.g. we assume that $\boldsymbol{s}_i$ is not a prefix of $\boldsymbol{s}_j$ for $i \neq j$. The state diagram has an initial vertex x, from which there are ℓ paths. The ith path has length smaller by one than the length of the ith forbidden pattern $\boldsymbol{s}_i$. Paths share a prefix if their related sequences share a corresponding prefix. Each vertex in the state diagram, except for the ones at the end of the ℓ paths, has out-degree σ, one edge for each possible symbol of Σ that can be seen in the corresponding read point of the constrained sequence. The edges are labelled by the

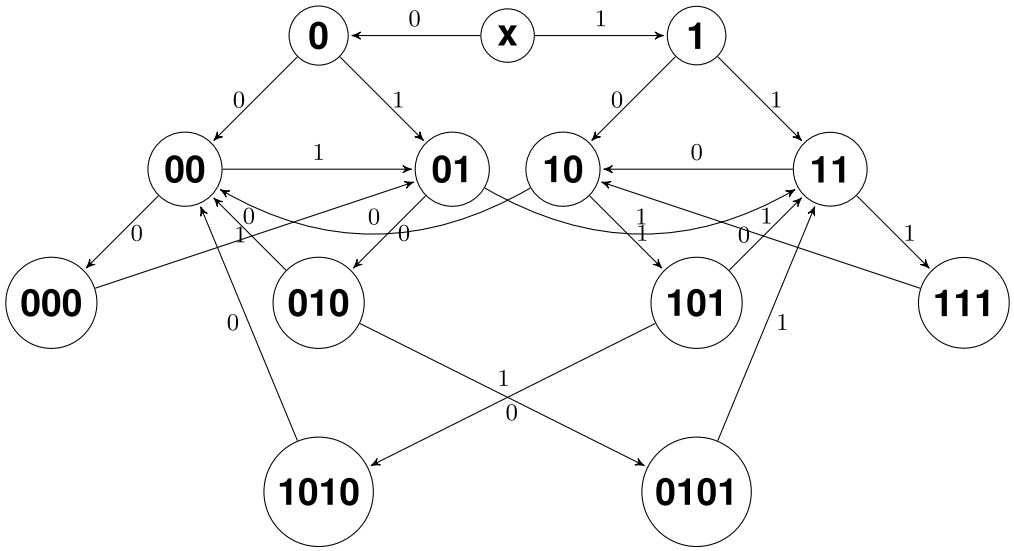

Fig. 1. State diagram for the $(3,3)$-locally-constrained de Bruijn sequences, via the forbidden patterns {0000, 1111, 01010, 10101}.

related symbols. The last symbol of $\boldsymbol{s}_i$ does not appear on an edge from the last vertex of the related path. A vertex in the state diagram is labelled by the prefix of the path which it represents. The initial vertex x is labelled with the sequence of length zero. Thus, to construct the state diagram we have to determine all the forbidden substrings in the (b,k)-locally-constrained de Bruijn sequence. This representation coincides with the representation of the constrained system and the related example for the $(3,3)$-locally-constrained de Bruijn sequences is depicted in Figure 1.

To determine exactly the maximum asymptotic rates of (b,k)-locally-constrained de Bruijn codes, we use the well-known Perron-Frobenius theory [52]. When b and k are given, by using the related state diagram, we can build a finite directed graph with labelled edges such that paths in the graph generate exactly all (b,k)-locally-constrained de Bruijn sequences. For example, when $(b,k)=(3,3)$ the 10×10 adjacency matrix A_G is

$s \ \backslash r$	000	01	00	010	1010	0101	101	11	10	111
000	0	1	0	0	0	0	0	0	0	0
01	0	0	0	1	0	0	0	1	0	0
00	1	1	0	0	0	0	0	0	0	0
010	0	0	1	0	0	1	0	0	0	0
1010	0	0	1	0	0	0	0	0	0	0
0101	0	0	0	0	0	0	0	1	0	0
101	0	0	0	0	1	0	0	1	0	0
11	0	0	0	0	0	0	0	0	1	1
10	0	0	1	0	0	0	1	0	0	0
111	0	0	0	0	0	0	0	0	1	0

,

where the entry in row s, column r of A_G is one if there is an edge from state s to state r in the state diagram. Note, that states x, 0, and 1 are omitted since no path lead to these vertices after they were accessed one time. The largest eigenvalue of A_G is $\lambda\approx 1.73459$. Hence, the capacity of this constrained system, which is the maximum asymptotic rate of $(3,3)$-locally-constrained de Bruijn code, is $\log\lambda=0.7946$. Similarly, we can compute the maximum asymptotic rates of (b,k)-locally-constrained de Bruijn codes for other values of b,k. Table I presents some values for the asymptotic rates of the constrained systems for small parameters. The asymptotic rate can be evaluated for infinite pairs of (b,k) as proved in the next theorem.

Theorem 9: For any positive integer $k>1$, the maximum asymptotic rate of a binary $(3,k)$-locally-constrained de Bruijn code is $\log_2\lambda$, where λ is the largest root of the polynomial equation

$$x^{2k-1}=x^{2k-3}+2x^{2k-4}+\cdots+(k-2)x^{k}+(k-1)x^{k-1}+(k-1)x^{k-2}+(k-2)x^{k-3}+\cdots+3\ x^2+2x+1.$$

Proof: Recall that $\mathcal{F}_{p,p+k}$ is the set of all period-p sequences of length $p+k$. Let $\mathcal{F}_1=\mathcal{F}_{1,k+1}\cup\mathcal{F}_{2,k+2}$, i.e., $\mathcal{F}_1$ contains the all-zero word of length $k+1$, the all-one word of length $k+1$, and the two words of length $k+2$ in which any two consecutive positions have distinct symbols. By Corollary 1, $\mathbb{C}_{DB}(n,3,k)=\mathcal{A}(n;\mathcal{F}_1)$, i.e., binary $(3,k)$-locally-constrained de Bruijn code of length n is an $\mathcal{F}_1$-avoiding code of length n.

Let $\mathcal{F}_2$ be the set which contains the all-zero word of length k and the all-one word of length $k+1$. Consider the $\mathbf{D}$-morphism defined first in [43], $\mathbf{D}: B^n\mapsto B^{n-1}$, $B=\{0,1\}$, where $\mathbf{D}(\boldsymbol{x})=\mathbf{D}(x_1,x_2,\ldots,x_n)=\boldsymbol{y}=(y_2,\ldots,y_n)$, with $y_i=x_i+x_{i-1}$, $2\le i\le n$. It was proved in [43] that the mapping $\mathbf{D}$ is a 2-to-1 mapping. Furthermore, $\boldsymbol{x}\in\mathcal{A}(n;\mathcal{F}_1)$ if and only if $\boldsymbol{y}\in\mathcal{A}(n-1;\mathcal{F}_2)$. Hence, $|\mathcal{A}(n;\mathcal{F}_1)|=2|\mathcal{A}(n-1;\mathcal{F}_2)|$. This implies that

$$\lim_{n\to\infty}\frac{\log_2|\mathcal{A}(n;\mathcal{F}_2)|}{n}=\lim_{n\to\infty}\frac{\log_2|\mathcal{A}(n;\mathcal{F}_1)|}{n}.$$

Hence, $\mathcal{A}(n;\mathcal{F}_1)$ and $\mathcal{A}(n;\mathcal{F}_2)$ have the same maximum asymptotic rate, and hence $\mathcal{A}(n;\mathcal{F}_2)$ can be computed instead of $\mathcal{A}(n;\mathcal{F}_1)$.

Let $\mathcal{A}_0(n;\mathcal{F}_2)$ be the set of all $\mathcal{F}_2$-avoiding words of length n which start with a *zero* and let $\mathcal{A}_1(n;\mathcal{F}_2)$ be the set of all $\mathcal{F}_2$-avoiding words of length n which start with a *one*. Clearly, the asymptotic rates of $\mathcal{A}(n;\mathcal{F}_2)$, $\mathcal{A}_0(n;\mathcal{F}_2)$, $\mathcal{A}_1(n;\mathcal{F}_2)$ are equal.

TABLE I
THE MAXIMAL ASYMPTOTIC RATES OF (b, k)-LOCALLY-CONSTRAINED DE BRUIJN CODES

	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
$b=2$	0.6942	0.8791	0.9468	0.9752	0.9881	0.9942	0.9971	0.9986	0.9993
$b=3$	0.4056	0.7946	0.9146	0.9614	0.9817	0.9912	0.9957	0.9978	0.9989
$b=4$	0	0.6341	0.8600	0.9392	0.9719	0.9865	0.9934	0.9966	0.9978
$b=5$	0	0.4709	0.7973	0.9150	0.9615	0.9818	0.9912	0.9957	0.9978
$b=6$	0	0.4517	0.7289	0.88412	0.94815	0.97574	?	?	?

Let

$$\Phi_1 : \mathcal{A}_0(n; \mathcal{F}_2) \mapsto \cup_{i=1}^{k-1} \mathcal{A}_1(n-i; \mathcal{F}_2)$$

be the mapping for which $\Phi_1(\boldsymbol{x} = (0, x_2, \ldots, x_n)) = (x_{i+1}, \ldots, x_n) \in \mathcal{A}_1(n-i; \mathcal{F}_2)$, where i is the smallest index that $x_{i+1} = 1$. Since $\mathcal{A}_0(n; \mathcal{F}_2)$ avoids the all-zero sequence of length k, it follows that the mapping Φ_1 is a well-defined bijection. Therefore,

$$|\mathcal{A}_0(n; \mathcal{F}_2)| = \sum_{i=1}^{k-1} |\mathcal{A}_1(n-i; \mathcal{F}_2)|. \tag{1}$$

Similarly, we can define the bijection

$$\Phi_2 : \mathcal{A}_1(n; \mathcal{F}_2) \mapsto \cup_{i=1}^{k} \mathcal{A}_0(n-i; \mathcal{F}_2)$$

and obtain the equality

$$|\mathcal{A}_1(n; \mathcal{F}_2)| = \sum_{i=1}^{k} |\mathcal{A}_0(n-i; \mathcal{F}_2)|. \tag{2}$$

Equations (1) and (2) imply that

$$\begin{aligned}
|\mathcal{A}_0(n; \mathcal{F}_2)| &= \sum_{i=1}^{k-1} |\mathcal{A}_1(n-i; \mathcal{F}_2)| \\
&= \sum_{i=1}^{k-1} \sum_{j=1}^{k} |\mathcal{A}_0(n-i-j; \mathcal{F}_2)| \\
&= \sum_{\ell=2}^{k} (\ell-1) |\mathcal{A}_0(n-\ell; \mathcal{F}_2)| + \sum_{\ell=1}^{k-1} \ell |\mathcal{A}_0(n-2h+\ell; \mathcal{F}_2)| \\
&= |\mathcal{A}_0(n-2; \mathcal{F}_2)| + \cdots + (k-1)|\mathcal{A}_0(n-k; \mathcal{F}_2)| \\
&+ (k-1)|\mathcal{A}_0(n-k-1; \mathcal{F}_2)| + \cdots + |\mathbb{A}_0(n-2k+1; \mathcal{F}_2)|.
\end{aligned}$$

It is again easy to verify that the maximum asymptotic rates of $\mathcal{A}_0(n-\ell; \mathcal{F}_2)$ for all $2 \leq \ell \leq 2k-1$ are equal. Let λ^n be this maximum asymptotic rate. The recursive formula can be solved now for λ and the maximum asymptotic rate of $\mathcal{A}_0(n; \mathcal{F}_2)$ will be $\log_2 \lambda$, where λ is computed as the largest root of the polynomial equation

$$\begin{aligned}
x^{2k-1} = {} & x^{2k-3} + 2x^{2k-4} + \cdots + (k-2)x^k + (k-1)x^{k-1} \\
& + (k-1)x^{k-2} + (k-2)x^{k-3} + \cdots + 2x + 1.
\end{aligned}$$

□

Using recursive formulas in Equations (1) and (2), we can compute the exact size of $\mathcal{A}(n; \mathcal{F}_2)$ efficiently. Hence, we can rank/unrank all words in $\mathcal{A}(n; \mathcal{F}_2)$ efficiently using enumerative techniques [19].

Computing the largest eigenvalue of the adjacency matrix is the key to computing the asymptotic rate. The size of the matrix and the number of its nonzero entries is important for reducing the complexity of the computation. Fortunately we can evaluate some of these parameters to get some idea for which parameters it is feasible to compute the largest eigenvalue. Since this is mainly a combinatorial problem we omit this messy computation. For some practical values of the parameters b and k, we computed the asymptotic rates and tabulated them in Table I. For large values of the parameters b and k, there are some difficulties to compute the largest eigenvalue in term of computation complexity. Fortunately, we can use some known techniques to find the bounds of these values [37], [46]. Also some probabilistic methods might be helpful to estimate the asymptotic rates. Finally, we note that we can use the idea in the proof of Theorem 9 to reduce the number of forbidden patterns by half, while keeping the same asymptotic rate. This is done in the following example.

Example 4: Consider the $(3, 3)$-locally-constrained de Bruijn sequences. The set of patterns which should be avoided is $\{0000, 1111, 01010, 10101\}$. As was illustrated in Figure 1, 10 nodes are required to represent a graph of constrained sequences avoiding these four patterns (the nodes labelled by x, 0, and 1, cannot be reached from the other nodes). Using Theorem 9, it can be observed that the asymptotic size of this code is the same as the code avoiding all patterns in set $\{000, 1111\}$. Hence, only 5 nodes are required to represent the state diagram of the constrained sequences avoiding these two patterns as depicted in Figure 2.

V. CODES WITH A LARGE CONSTRAINED SEGMENT

After considering in Section IV enumerations for (b, k)-locally-constrained de Bruijn sequences, based on constrained coding, which are mainly efficient for small b compared to k, we will use the theory of shift registers for a construction which is applied for considerably larger b compared to k. Such constructions for (σ^k, k)-locally-constrained de Bruijn codes are relatively simple as each sequence in the code is formed by concatenations of the same de Bruijn sequence. We can use for this construction any known efficient construction for de Bruijn sequences, e.g., [23]. The disadvantage is that the rate of the related code is zero. In this section we present two constructions of codes in which b is relatively large compared to k and the rates of the codes are greater than zero.

Let $\mathcal{P}_k$ be the set of all primitive polynomials of degree k over $\mathbb{F}_q$. By Theorem 1, there are $\frac{\phi(q^k-1)}{k}$ polynomials in $\mathcal{P}_k$. Each polynomial is associated with a linear feedback shift

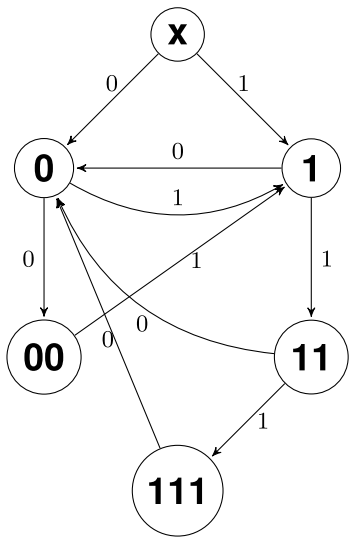

Fig. 2. Reduced state diagram to count the number of $(3,3)$-locally-constrained de Bruijn sequences.

register and a related m-sequence of length q^k-1. Let $\mathcal{S}_{\mathcal{P}_k}$ be this set of m-sequences. In each such sequence, each nonzero k-tuple over $\mathbb{F}_q$ appears exactly once as a window in one period of the sequence. In each such sequence, there is a unique run of length k for each nonzero symbol. This run of the same symbol is the longest such run in each sequence. But, there is another window property for all the sequences in $\mathcal{S}_{\mathcal{P}_k}$.

Lemma 4: Each $(2k)$-tuple over $\mathbb{F}_q$ appears at most once as a window in one of the sequences of $\mathcal{S}_{\mathcal{P}_k}$.

Proof: Let $f_1(x)$ and $f_2(x)$ be two distinct primitive polynomials in $\mathcal{P}_k$, whose state diagrams contain the cycles of length q^k-1, $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. By Theorem 2, the polynomial $f_1(x)f_2(x)$ has degree $2k$ and its state diagram contains the two cycles $\mathcal{C}_1$ and $\mathcal{C}_2$. Each $(2k)$-tuple appears exactly once in a window of length $2k$ in one of the cycles of the state diagram related to $f_1(x)f_2(x)$. Hence, each such $(2k)$-tuple appears at most once in either $\mathcal{C}_1$ or $\mathcal{C}_2$.

The windows of length k are distinct within each sequence and the windows of length $2k$ are distinct between any two sequences which implies the claim of the lemma. □

We are now in a position to describe our construction of locally-constrained de Bruijn codes with a large constrained segment.

Construction 1: Let $\mathcal{S}_{\mathcal{P}_k}$ be the set of all m-sequences of order k over $\mathbb{F}_q$. Each sequence will be considered as an acyclic sequence of length q^k-1, where the unique run of k *ones* is at the end of the sequence. We construct the following code

$$\mathbb{C} \triangleq \{(1^{\epsilon_0=0}\boldsymbol{s}_1 1^{\epsilon_1}, \boldsymbol{s}_2 1^{\epsilon_2}, \ldots, \boldsymbol{s}_{\ell-1}1^{\epsilon_{\ell-1}}, \boldsymbol{s}_\ell 1^{\epsilon_\ell}) \ : \\ \boldsymbol{s}_i \in \mathcal{S}_{\mathcal{P}_k},\ 0 \le \epsilon_i \le k,\ 1 \le i \le \ell\} \ .$$

Theorem 10: The code $\mathbb{C}$ contains $(q^k-1,2k)$-locally-constrained de Bruijn sequences, each one of length at least $\ell(q^k-1)$ and at most $\ell(q^k+k-1)$. The size of $\mathbb{C}$ is M^ℓ, where $M=\frac{\phi(q^k-1)(k+2)}{k}$ and $k \ge 3$.

Proof: We first prove that each codeword of $\mathbb{C}$ is a $(q^k-1,2k)$-locally-constrained de Bruijn sequence. Let $\boldsymbol{s}=1^{\epsilon_{i-1}}\boldsymbol{s}_i 1^{\epsilon_i}$, $1 \le i \le \ell$, be a substring of a codeword in $\mathbb{C}$. We first note that except for the windows of length $2k$ having more than k *ones* at the start or at the end of $\boldsymbol{s}$, all the other windows of length $2k$ contained in $\boldsymbol{s}$ appear in the cyclic sequence $\boldsymbol{s}_i$. These new windows (having more than k *ones*) as well as the windows which contain the run of *ones* 1^{ϵ_i+k} appear only between two sequences $\boldsymbol{s}_j$ and $\boldsymbol{s}_{j+1}$ and as such they are separated by at least q^k-k-1 symbols. Moreover, each window of length $2k$ containing *ones* from one such run is clearly unique. This implies that each sequence of $\mathbb{C}$ is a $(q^k-1,2k)$-locally-constrained de Bruijn sequence (as repeated windows of length $2k$ can occur only between different s_i's). Since each $\boldsymbol{s}_i$ has length q^k-1 and $0 \le \epsilon_i \le k$, it follows that the length of a codeword is at least $\ell(q^k-1)$ and at most $\ell(q^k+k-1)$. The number of sequences that can be used for the substring $\boldsymbol{s}_i 1^{\epsilon_i}$ is $\frac{\phi(q^k-1)(k+2)}{k}$ since $\boldsymbol{s}_i$ can be chosen in $\frac{\phi(q^k-1)}{k}$ ways (see Theorem 1) and 1^{ϵ_i} can be chosen in $k+2$ distinct ways. It implies that $|\mathbb{C}|=M^\ell$. □

The codewords in the code $\mathbb{C}$ obtained via Construction 1 can be of different lengths. We will now construct a similar code in which all codewords have the same length. Let $\mathbb{C}'$ be a code which contains all the prefixes of codewords from $\mathbb{C}$. Let

$$\mathbb{C}_1 \stackrel{\text{def}}{=} \{(\boldsymbol{s}_1\boldsymbol{s}_2) : \boldsymbol{s}_1 \in \mathbb{C}, \boldsymbol{s}_2 \in \mathbb{C}', \text{length}(\boldsymbol{s}_1\boldsymbol{s}_2) = \ell(q^k+k)\}.$$

In other words we take the sequences of the code $\mathbb{C}$ to be of a length large enough and after that we truncate them to have a given fixed length.

Theorem 11: The code $\mathbb{C}_1$ contains $(q^k-1,2k)$-locally-constrained de Bruijn sequences, of length $n=\ell(q^k+k)$ and its size is at least M^ℓ, where $M=\frac{\phi(q^k-1)(k+2)}{k}$.

Proof: The codewords in $\mathbb{C}_1$ are formed from the codewords in $\mathbb{C}$ by lengthening them to the required length with prefixes of codewords in $\mathbb{C}$. The lengthening does not change the structure of the codewords, as it only changes their length. Hence, the proof that the codewords are $(q^k-1,2k)$-locally-constrained de Bruijn sequence is the same as in the proof of Theorem 10. The number of codewords is the same as in $\mathbb{C}$ if there is exactly one way to lengthen each codeword of $\mathbb{C}$. Since there is usually more than one such way, it follows that the code will be of a larger size. □

Corollary 4: The rate of the code $\mathbb{C}_1$ is at least $\frac{k}{q^k+k}$.

Construction 1 yields acyclic sequences. Can we find a related construction for cyclic locally-constrained de Bruijn sequences with similar parameters? The answer is yes. Construction 1 can be viewed as a construction for cyclic sequences of different length. To have a code with cyclic $(q^k-1,2k)$-locally-constrained de Bruijn sequences of the same length (as the acyclic sequences in $\mathbb{C}_1$), we can restrict the values in the ϵ_i's. For example, we can require that $\sum_{i=1}^{\ell}\epsilon_i = \lfloor\frac{\ell(k+1)}{2}\rfloor$. This will imply similar results for the cyclic code as for the acyclic code.

The code $\mathbb{C}_1$ can be slightly improved in size, but as it does not make a significant increase in the rate we ignore the possible improvements. The same construction can be applied

with a larger number of cycles whose length is $\sigma^k - 1$ (σ not necessarily a prime power) and with a similar window length as explained in the next paragraph. We do not have a general construction for such cycles, but we outline a simple method to search for them.

Let $\mathcal{D}(\sigma, k)$ be the set of de Bruijn sequences in $G_{\sigma,k}$. Let δ be a positive integer. Construct a graph whose vertices are all the de Bruijn sequences in $\mathcal{D}(\sigma, k)$. Two vertices (de Bruijn sequences) are connected by an edge if they have the same window of length greater than $k + \delta$. Now let $\mathcal{T}$ be an independent set in the graph. Two sequences related to this independent set do not share a window of length $k + \delta$ or less, so we can apply Construction 1 and its variants, where $0 \leq \epsilon_i \leq \delta$. Note that in Construction 1, the set of sequences in $\mathcal{S}_{\mathcal{P}_k}$ form an independent set with $\delta = k$.

A computer search for the 2048 binary de Bruijn sequences of length 32 was performed. An independent set of size 8 was found for $\delta = 5$ compared to the 6 m-sequences of length 31. Furthermore, for $\delta = 4$, an independent set of size 4 was found, and for $\delta = 2$ the size of the independent set that was found was only 2. This implies that this direction of research can be quite promising.

Next, we present another construction of (b, k)-locally-constrained de Bruijn sequences for a large constrained segment. This construction yields $(q^k, 2k+1)$-locally-constrained de Bruijn sequences, and uses slightly larger window length and segment length compared to Construction 1. However, it constructs codes with asymptotic rate that approaches 1 as $k \to \infty$.

For a sequence $\boldsymbol{s} = (s_1, \ldots, s_n)$ and integer ℓ, let $\text{Suff}_\ell(\boldsymbol{s})$ denote its ℓ-suffix, i.e., $\boldsymbol{s}[n - \ell + 1, n]$ if $\ell \leq n$, and $\boldsymbol{s}[1, n]$ otherwise. Let $CP_\ell(\boldsymbol{s})$ denote a string of length ℓ which is created by repeatedly concatenating $\boldsymbol{s}$ to itself and taking the ℓ-prefix, i.e., $CP_\ell(\boldsymbol{s}) = \boldsymbol{s}^\ell[1, \ell]$. When $n \geq \ell$ the set of *denied ℓ-substrings of $\boldsymbol{s}$*, $DS_\ell(\boldsymbol{s})$, consists of the following sequences of length ℓ,

$$\begin{aligned} DS_\ell(\boldsymbol{s}) \triangleq &\{\boldsymbol{s}[i, i + \ell - 1] \ : \ i \in [n - \ell + 1]\} \\ &\cup \{CP_\ell(\boldsymbol{s}[i, n]) \ : \ i \in [n - \ell + 2, n]\}. \end{aligned}$$

The *allowed ℓ-substrings of $\boldsymbol{s}$*, denoted as $AS_\ell(\boldsymbol{s})$ are the complementary set $AS_\ell(\boldsymbol{s}) \triangleq DS_\ell(\boldsymbol{s})^c$.

Lemma 5: For every sequence $\boldsymbol{u} \in AS_\ell(\boldsymbol{s})$, $\boldsymbol{u}$ appears exactly once as a substring of $\boldsymbol{su}$, at its end.

Proof: By the definition of $AS_\ell(\boldsymbol{s})$, the sequence $\boldsymbol{u}$ can appear in $\boldsymbol{su}$ an additional time as a substring only at position $i \in [n - \ell + 2, n]$. Assume to the contrary that such index i exists and $(\boldsymbol{su})[i, i + \ell - 1] = \boldsymbol{u}$. However, this implies that $\boldsymbol{u} = CP_\ell(\boldsymbol{s}[i, n])$ which is a contradiction to the definition of $AS_\ell(\boldsymbol{s})$. □

The next step is to present a code $\mathbb{C}_2$, which is a $(q^k, 2k + 1)$-locally-constrained de Bruijn code. Each codeword of $\mathbb{C}_2$ is constructed by an iterative concatenation of m sequences $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m \in \mathbb{F}_q^{k+1}$. Each tuple is picked from the set of allowed ℓ-substrings of its preceding segment of length q^k. Therefore, it is ensured that throughout the concatenation process the constructed sequence is always a $(q^k, 2k + 1)$-locally-constrained de Bruijn sequence, as new repeats cannot be created within any segment of length q^k.

Construction 2: Let m be an integer. We construct the following code,

$$\begin{aligned} \mathbb{C}_2 \triangleq \{&\boldsymbol{u}_1 \cdots \boldsymbol{u}_m \ : \ \boldsymbol{u}_1 \in \mathbb{F}_q^{k+1}, \forall_{i \in [2,m]}, \\ &\boldsymbol{u}_i \in AS_{k+1}(\text{Suff}_{q^k}(\boldsymbol{u}_1 \cdots \boldsymbol{u}_{i-1}))\}. \end{aligned}$$

Lemma 6: The code $\mathbb{C}_2$ contains $(q^k, 2k + 1)$-locally-constrained de Bruijn sequences of length $n = m(k + 1)$. The size of $\mathbb{C}_2$ is at least $(q^k(q - 1))^m$.

Proof: Let $\boldsymbol{s} \in \mathcal{C}$. Assume to the contrary that there exist indices $i, j \in [n - 2k]$ with $0 < j - i < q^k$ such that $\boldsymbol{s}[i, i + 2k] = \boldsymbol{s}[j, j + 2k]$. By the definition of Construction 2, the substring $\boldsymbol{s}[j, j + 2k]$ contains a tuple $\boldsymbol{u} \in \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m\}$ in its entirety, at some position $r \leq k$. However, it follows that $\boldsymbol{s}[i + r, i + r + k] = \boldsymbol{u}$ as well which is a contradiction to Lemma 5 as $\boldsymbol{u}$ appears only once in $\boldsymbol{s}[j + r - q^k, j + r + k]$, at its end.

For the size of $\mathbb{C}_2$, let $\boldsymbol{s}_i$ denote the sequence $\boldsymbol{s}_i = \text{Suff}_{q^k}(\boldsymbol{u}_1 \cdots \boldsymbol{u}_{i-1})$ for every $i \in [2, m]$. Note that the size of $AS_{k+1}(\boldsymbol{s}_i)$ satisfies

$$|AS_{k+1}(\boldsymbol{s}_i)| \geq q^{k+1} - q^k = q^k(q - 1).$$

Hence, for every $i \in [m]$, there are at least $q^k(q-1)$ possible values for $\boldsymbol{u}_i$, and the lemma statement follows immediately. □

Corollary 5: The rate of the code $\mathbb{C}_2$ is at least $\frac{k + \log_q(q-1)}{k+1}$.

VI. Application to the ℓ-Symbol Read Channel

In this section, we show that cyclic (b, k)-locally-constrained de Bruijn codes can be used to correct synchronization errors in the ℓ-symbol read channel [8], [16], [67]. Previously, only substitution errors were considered in such a channel. Each cyclic (b, k)-locally-constrained de Bruijn sequence which forms a codeword in the channel can be used to correct a number of limited synchronization errors which might occur. The correction does not depend on the other codewords of the code. The mechanism used in this section will be of help in correcting such errors in racetrack memory as discussed in the next section. We will consider now $\mathbb{F}_q$ as our alphabet although the method can be applied to alphabets of any size. The reason is that other error-correcting codes, for this purpose, are defined over $\mathbb{F}_q$.

Definition 4: Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$ be a q-ary sequence of length n. In the *ℓ-symbol read channel*, if $\boldsymbol{x}$ is a codeword then the corresponding *ℓ-symbol read vector* of $\boldsymbol{x}$ is

$$\pi_\ell(\boldsymbol{x}) = ((x_1, \ldots, x_\ell), (x_2, \ldots, x_{\ell+1}), \ldots, (x_n, x_1 \ldots, x_{\ell-1})).$$

Note, that $\boldsymbol{x}$ might not be a cyclic sequence, but the channel read the symbols cyclically, i.e., after the last symbol, the first symbol, the second symbol and so on are read to complete an ℓ-symbol read in each position of $\pi_\ell(\boldsymbol{x})$. In this channel, if the ℓ-symbol read sequence $\pi_\ell(\boldsymbol{x})$ is received with no error, it is simple to recover the codeword $\boldsymbol{x}$. However, several types of errors, such as substitution errors and synchronization errors, can occur. Substitution errors were considered in [8],

[16], [67], but a synchronization error might be a more common one. We focus now on the ℓ-symbol read channel with only synchronization errors which are deletions and sticky-insertions. A *sticky-insertion* is the error event when an ℓ-tuple $(x_i, \ldots, x_{i+\ell-1})$ is repeated, i.e., read more than once. A deletion is the error event when an ℓ-tuple is deleted. A burst of deletions of length at most b is an error in which at most b consecutive ℓ-tuples are deleted. In this work, when we consider multiple bursts of deletions of length at most b, any two bursts are not adjacent. Otherwise, two adjacent bursts will be merged to be one larger burst.

Example 5: Let $\boldsymbol{x} = (0,1,0,0,1,0,0,0)$ be a codeword. When $\ell = 2$, the 2-symbol read sequence of $\boldsymbol{x}$ is $\pi_2(\boldsymbol{x}) = ((0,1),(1,0),(0,0),(0,1),(1,0),(0,0),(0,0),(0,0))$. If there is a sticky-insertion at the second location and a burst of deletions of length two at the 6-th and 7-th positions, we obtain the 2-symbol read sequence $\pi_2(\boldsymbol{y}) = ((0,1),(1,0),(1,0),(0,0),(0,1),(1,0),(0,0))$.

The goal of this section is to show that a cyclic locally-constrained de Bruijn code is a code correcting sticky-insertions and bursts of deletions in the ℓ-symbol read channel.

Theorem 12: Let b and k be two positive integers such that $q^k \geq b \geq 2$ and $\ell = k + b - 2$. The code $\mathbb{C}_{DB}(n, b, k)$ can correct any number of sticky-insertions and any number of bursts of deletions (depending only on n and b) whose length is at most $b-2$ (including one possible cyclic burst of this size) in the ℓ-symbol read channel.

Proof: Let $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$ be a codeword and

$$\pi_\ell(\boldsymbol{c}) = ((c_0, \ldots, c_{\ell-1}), (c_1, \ldots, c_\ell), \ldots, (c_{n-1}, \ldots, c_{n+\ell-2})),$$

where indices are taken modulo n, be a corresponding ℓ-symbol read sequence of $\boldsymbol{c}$. Let

$$\pi_\ell(\boldsymbol{y}) = ((y_{1,1}, \ldots, y_{1,\ell}), (y_{2,1}, \ldots, y_{2,\ell}), \ldots, (y_{t,1}, \ldots, y_{t,\ell}))$$

be a received ℓ-symbol read sequence.

Clearly, since no more than $b-2$ consecutive deletions occurred, it follows that there exists an i, $1 \leq i \leq b-1$, such that $(c_i, c_{i+1}, \ldots, c_{i+\ell-1}) = (y_{1,1}, \ldots, y_{1,\ell})$. Similarly, there also exists a j, $0 \leq j \leq b-1$, such that $(c_{i+j}, c_{i+j+1}, \ldots, c_{i+j+\ell-1}) = (y_{2,1}, \ldots, y_{2,\ell})$.

Since $\ell = k + b - 2$ and at most $b-2$ deletions occurred between the first two consecutive ℓ-reads, it follows that either the substring $(y_{2,1}, \ldots, y_{2,k})$ is a substring of $(y_{1,1}, \ldots, y_{1,\ell})$ or $(y_{1,b}, y_{1,b+1}, \ldots, y_{1,\ell}) = (y_{2,1}, \ldots, y_{2,k-1})$. If $(y_{2,1}, \ldots, y_{2,k})$ is a substring of $(y_{1,1}, \ldots, y_{1,\ell})$, then since any b consecutive k-tuples of $\boldsymbol{c}$ are distinct, it follows that there exists a unique j, $0 \leq j \leq b-2$, such that $(c_{i+j}, c_{i+j+1}, \ldots, c_{i+j+k-1}) = (y_{2,1}, \ldots, y_{2,k})$. If $j = 0$, then a sticky-insertion has occurred and if $j = 1$ no error occurred in the second read, and if $2 \leq j \leq b-2$, then a burst of $j-1$ deletions has occurred. If a burst of $b-2$ deletions has occurred, then $(y_{1,b}, y_{1,b+1}, \ldots, y_{1,\ell}) = (y_{2,1}, \ldots, y_{2,k-1})$. The same process continues between the second read and the third read and so on until the last read and the first read which behave in the same way since also cyclically the longest run of deletions has at most length $b-2$. Thus, using the overlaps between consecutive reads we can recover the codeword $\boldsymbol{c}$. □

Note, that it is possible to assume that there are only sticky-insertion errors or only deletion errors in the ℓ-symbol read channel. In these cases, we can always choose an appropriate locally-constrained de Bruijn code to correct these errors. This implies the following consequences.

Corollary 6: The code $\mathbb{C}_{DB}(n, b, k)$ can correct any number of bursts of deletions whose length is at most $b-1$ in the ℓ-symbol read channel, where $\ell = k + b - 1$.

Corollary 7: The code $C_{DB}(n, 2, k)$ can correct any number of sticky-insertions in the ℓ-symbol read channel, where $k = \ell$.

We can consider an error event of an ℓ-tuple $(y_1, \ldots, y_\ell)$ which is added into the ℓ-symbol read sequence $\boldsymbol{c}$, where $(y_1, \ldots, y_\ell)$ may not a substring of $\boldsymbol{c}$. This is a general insertion. It is not difficult to use a locally-constrained de Bruijn code to correct a single insertion in the ℓ-symbol read channel. The proof follows with the same arguments as in the proof of Theorem 12.

Proposition 1: The code $C_{DB}(n, 2, k)$ can correct a single insertion in the ℓ-symbol read channel where $k = \ell - 1$.

However, if there are many such insertions, a locally-constrained de Bruijn code is not enough to correct these errors. For example, if $k = \ell = 3$ and

$$\pi_\ell(\boldsymbol{c}) = ((c_0, c_1, c_2), (c_1, c_2, c_3), \ldots, (c_{n-1}, c_0, c_1)).$$

Assume that there are three insertions and that the ℓ-symbol read sequence is

$$((c_0, c_1, c_2), (c_1, c_2, c_3'), (c_1, c_2, c_3), (c_2, c_3', c_4), (c_2, c_3, c_4), (c_3', c_4, c_5), (c_3, c_4, c_5), \ldots, (c_{n-1}, c_0, c_1)).$$

In this case, we cannot distinguish between c_3 and c_3' using a locally-constrained de Bruijn code.

VII. Applications to Racetrack Memories

Racetrack memory is an emerging non-volatile memory technology which has attracted significant attention in recent years due to its promising ultra-high storage density and low power consumption [56], [64]. The basic information storage element of a racetrack memory is called a *domain*, also known as a *cell*. The magnetization direction of each cell is programmed to store information. The reading mechanism is operated by one or more *read ports*, called *heads*. In order to read the information, each cell is shifted to its closest head by a *shift operation*. Once a cell is shifted, all other cells are also shifted in the same direction and in the same speed. Normally, along the racetrack strip, all heads are fixed and equally spaced [68]. Each head thus reads only a block of consecutive cells which is called a *data segment*.

A shift operation might not work perfectly. When the cells are not shifted (or under-shifted), the same cell is read again in the same head. This event causes a repetition (or sticky-insertion) error. When the cells are shifted by more than a single cell location (or over-shifted), one cell or a block of cells is not read in each head. This event causes a single deletion or a burst of consecutive deletions. We note that

the maximum number of consecutive deletions is limited or in other words, the burst of consecutive deletions has limited length. An experimental result shows that the cells are over-shifted by at most two locations with extremely high probability [68]. In this paper, we study both kinds of errors and refer to these errors as *limited-shift errors*.

Since limited-shift errors can be modeled as sticky-insertions and bursts of consecutive deletions with limited length, sticky-insertion/deletion-correcting codes can be applied to combat these limited-shift errors. Although there are several known *sticky-insertion-correcting codes* [20], [40], [50], *deletion-correcting codes* [5], [36], [47], *single-burst-deletion-correcting codes* [17], [61], and *multiple-burst-deletion-correcting codes* [35], there is a lack of knowledge on codes correcting a combination of multiple bursts of deletions and sticky-insertions. Correcting these type of errors are especially important in the racetrack memories. In this section, motivated by the special structure of having multiple heads in racetrack memories, we study codes correcting multiple bursts of deletions and sticky-insertions. To correct shift errors in racetrack memories with only a single head, Vahid *et al.* [51] recently studied codes correcting two shift errors of deletions and/or insertions.

Another approach to combat limited-shift errors is to leverage the special feature of racetrack memories where it is possible to add some extra heads to read cells. If there is no error, the information read in these extra heads is redundant. However, if there are limited-shift errors, this information is useful to correct these errors. Recently, several schemes have been proposed to leverage this feature [13], [15], [68] in order to tackle this problem. However, in [13], [15], each head needs to read all the cells while in this model, each head only needs to read a single data segment. Our goal in this section is to present several schemes to correct synchronization errors in racetrack memories, all of which are based on locally-constrained de Bruijn sequences and codes. In some of these schemes we add extra heads and some are without adding extra heads. Let N, n, m be three positive integers such that $N = n \cdot m$. The racetrack memory comprises N cells and m heads which are equally spaced. Each head will read a segment of n cells. For example, in Fig. 3, the racetrack memory contains 15 data cells and three heads are placed initially at the positions of cells $c_{1,1}$, $c_{2,1}$, and $c_{3,1}$, respectively. Each head reads a data segment of length 5.

In general, if $\boldsymbol{c} = (c_1, c_2, \ldots, c_N)$ is the stored data then the output of the i-th head is $\boldsymbol{c}^i = (c_{i,1}, \ldots, c_{i,\ell})$ where $c_{i,j} = c_{(i-1)\cdot n+j}$ for $1 \leq i \leq m$ and $1 \leq j \leq n$. Hence, an output matrix describing the output from all m heads (without error) is:

$$\begin{pmatrix} \boldsymbol{c}^1 \\ \boldsymbol{c}^2 \\ \vdots \\ \boldsymbol{c}^m \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,n} \end{pmatrix}.$$

When an under-shift error (a sticky-insertion error) occurs, one column is added to the above matrix by repeating a related column of the matrix. When over-shift errors (a deletion error or a burst of b consecutive deletions) occur, one or a few consecutive columns in the matrix are deleted. Our goal is to combat these shift errors in racetrack memories. We note that each column in the above matrix can be viewed as a symbol in an alphabet of size $q = 2^m$. In particular, let $\Phi_m : \mathbb{F}_2^m \mapsto \mathbb{F}_q$ be any bijection. For each column $\widehat{\boldsymbol{c}}_j = (c_{2,j}, \ldots, c_{m,j})^T$, $\Phi_m(\widehat{\boldsymbol{c}}_j) = v_j \in \mathbb{F}_q$. Hence, to correct under-shift and over-shift errors in racetrack memories, we construct a q-ary code correcting sticky-insertion errors and bursts of deletion errors.

A. Correcting Errors in Racetrack Memories Without Extra Heads

Our main goal in this subsection is to study q-ary codes correcting a combination of sticky-insertions and bursts of deletions to combat synchronization errors in racetrack memories. The first goal is to construct q-ary b-limited t_1-burst-deletions-correcting codes. Such a code can correct t_1 bursts of deletions if the length of each such burst is at most b, i.e., at most b deletions occurred in each such burst and each pair of these bursts are separated by symbols which are not in any error.

Lemma 7: Let $\boldsymbol{s}$ be a $(b, 1)$-locally-constrained de Bruijn sequence over an alphabet of size q. Let $\boldsymbol{s}(\Delta^-)$ be the sequence obtained from $\boldsymbol{s}$ after deleting all symbols specified by the locations in the set Δ^- such that the number of consecutive deletions is at most $b-1$. Then the set Δ^- is uniquely determined from $\boldsymbol{s}$ and $\boldsymbol{s}(\Delta^-)$.

Proof: Assume that $\Delta^- = \{i_1, i_2, \ldots, i_t\}$ where $i_1 < i_2 < \cdots < i_t$ is the set of t locations of all t deleted symbols. Since i_1 is the leftmost index in which a deletion has occurred, it follows that $\boldsymbol{s}[1, i_1 - 1] = \boldsymbol{s}(\Delta^-)[1, i_1 - 1]$. Furthermore, since $\boldsymbol{s}$ is a $(b, 1)$-locally-constrained de Bruijn sequence, it follows that the symbols $\boldsymbol{s}[i]$ for $i_1 \leq i \leq i_1 + b - 1$ are distinct. Since $\boldsymbol{s}[i_1]$ was deleted and the maximum number of consecutive deletions is at most $b-1$, it follows that $\boldsymbol{s}(\Delta^-)[i_1] \neq \boldsymbol{s}[i_1]$. So, i_1 is the leftmost index in which $\boldsymbol{s}$ and $\boldsymbol{s}(\Delta^-)$ differ. Therefore, we can determine i_1 from the two vectors $\boldsymbol{s}$ and $\boldsymbol{s}(\Delta^-)$. Let $\Delta_1^- \triangleq \Delta^- \setminus \{i_1\} = \{i_2, \ldots, i_t\}$. To correct the first error, we insert the symbol $\boldsymbol{s}[i_1]$ into the i_1-th position of $\boldsymbol{s}(\Delta^-)$ and obtain the vector $\boldsymbol{s}(\Delta_1^-)$. Similarly, we can continue now to determine, one by one, the other positions where deletions have occurred using words $\boldsymbol{s}$ and $\boldsymbol{s}(\Delta_1^-)$.

Thus, the set Δ^- can be determined from $\boldsymbol{s}$ and $\boldsymbol{s}(\Delta^-)$ and the lemma is proved. □

We are now ready to present a construction of q-ary b-limited t_1-burst-deletion-correcting codes. We will show that the maximum rate of these codes is close to the maximum rate of codes correcting multiple erasures, especially when q is large. A q-ary *t-erasure-correcting code of length ℓ* is a set of q-ary words of length ℓ in which one can correct any set of t-erasures, i.e., t known positions whose values are not known.

Construction 3: Let $\boldsymbol{s} = (s_1, s_2, \ldots, s_\ell)$ be a $(b, 1)$-locally-constrained de Bruijn sequence over an alphabet of size q_1. Let $\mathbb{C}_{q_2}(\ell, t)$ be a q_2-ary t-erasure-correcting code of length ℓ and let $q = q_1 \cdot q_2$. For each word $\boldsymbol{c} = (c_1, c_2, \ldots, c_\ell) \in \mathbb{C}_{q_2}(\ell, t)$,

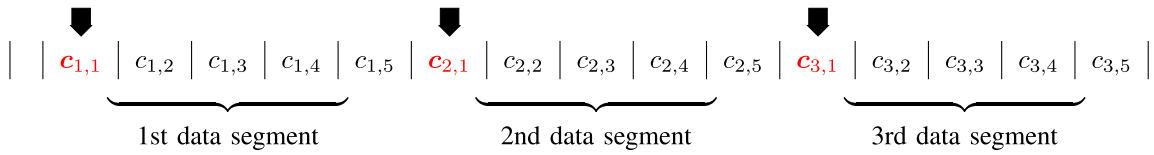

Fig. 3. Racetrack memory with three heads and 15 domains.

we define $\boldsymbol{f}(\boldsymbol{c},\boldsymbol{s}) = (f_1, f_2, \ldots, f_\ell)$, where $f_i = (c_i, s_i)$ for all $1 \le i \le \ell$. Construct the q-ary code of length ℓ which is denoted by $\mathbb{C}_q(b,\ell,t)$

$$\mathbb{C}_q(b,\ell,t) \triangleq \{\boldsymbol{f}(\boldsymbol{c},\boldsymbol{s}) : \boldsymbol{c} \in \mathbb{C}_{q_2}(\ell,t)\}.$$

Theorem 13: The code $\mathbb{C}_q(b,\ell,t)$ obtained in Construction 3 is a q-ary $(b-1)$-limited t_1-burst-deletions-correcting code where $t = t_1 \cdot (b-1)$.

Proof: Let $\boldsymbol{f} = (f_1, \ldots, f_\ell) \in \mathbb{C}_q(b,\ell,t)$ be a codeword of length ℓ. Let $\Delta^- = \{i_1, \ldots, i_t\}$ be the set of all deleted positions such that $i_1 < \cdots < i_t$ and let $\boldsymbol{f}(\Delta^-)$ denote the received word. Since there are at most t_1 bursts of deletions whose length is at most $b-1$, it follows that $|\Delta^-| = t \le t_1 \cdot (b-1)$ and there are at most $b-1$ consecutive integers in Δ^-. From the received word $\boldsymbol{f}(\Delta^-)$, we can extract the unique pair of words $(\boldsymbol{c}(\Delta^-), \boldsymbol{s}(\Delta^-))$, where $\boldsymbol{c}(\Delta^-)$ and $\boldsymbol{s}(\Delta^-)$ are two words obtained from $\boldsymbol{c}$ and $\boldsymbol{s}$, respectively, after deleting all the symbols specified by the locations in the set Δ^-. By Lemma 7, the set Δ^- can be determined from the words $\boldsymbol{s}(\Delta^-)$ and $\boldsymbol{s}$ since $\boldsymbol{s}$ is a $(b,1)$-locally-constrained de Bruijn sequence. This implies that now we have a word of length ℓ in which t positions have unknown values, i.e., t erasures. Moreover, $\mathbb{C}_{q_2}(\ell,t)$ can correct up to t erasures. Hence, using the decoder of $\mathbb{C}_{q_2}(\ell,t)$, the codeword $\boldsymbol{c}$ can be recovered from $\boldsymbol{c}(\Delta^-)$ and Δ^-.

Thus, the codeword $\boldsymbol{f} = (f_1, \ldots, f_\ell)$ such that $f_i = (c_i, s_i)$ for $1 \le i \le \ell$, can be recovered from $\boldsymbol{c}$ and $\boldsymbol{s}$, and the theorem is proved. □

The proof of Theorem 13 implies a simple decoding algorithm to recover $\boldsymbol{f}$. Let

$$R(\mathbb{C}_{q_2}(\ell,t)) = \frac{\log_{q_2}|\mathbb{C}_{q_2}(\ell,t)|}{\ell}$$

denote the rate of the code $\mathbb{C}_{q_2}(\ell,t)$. By Theorem 7 there exists a $(b,1)$-locally-constrained de Bruijn sequences π over an alphabet of size q_1 if $q_1 \ge b$. By Construction 3, if $\boldsymbol{s}$ is a $(b,1)$-locally-constrained de Bruijn sequence of length ℓ and there exists a q_2-ary t-erasure-correcting code of length ℓ, then $|\mathbb{C}_q(b,\ell,t)| = |\mathbb{C}_{q_2}(\ell,t)|$. If $q_1 = q/q_2 = b$, then the rate of the q-ary code $\mathbb{C}_q(b,\ell,t)$ is

$$R = \frac{\log_q|\mathbb{C}_q(b,\ell,t)|}{\ell} = \frac{\log_q q_2 \cdot \log_{q_2}|\mathbb{C}_{q_2}(\ell,t)|}{\ell}$$
$$= \log_q(q/q_1) \cdot R(\mathbb{C}_{q_2}(\ell,t)) = (1-\log_q b) \cdot R(\mathbb{C}_{q_2}(\ell,t)).$$

It is well-known that a code with minimum Hamming distance $t+1$ can correct t erasure errors. Moreover, for any $0 < \epsilon, \delta < 1$, there exists a "near MDS" code [2] of length ℓ with minimum Hamming distance $t = \delta \cdot \ell$ and rate $R(\mathbb{C}_{q_2}(\ell,t)) \ge 1 - \delta - \epsilon$ [34]. Hence, there exists a code of length ℓ correcting $t = \delta \cdot \ell$ erasures whose rate is $R(\mathbb{C}_{q_2}(\ell,t)) \ge 1 - \delta - \epsilon$. Therefore, we have the following theorem.

Theorem 14: Given $0 < \delta, \epsilon < 1$, there exists a q-ary b-limited t_1-burst-deletion-correcting code of length ℓ such that its rate R satisfies

$$R \ge (1 - \log_q(b+1)) \cdot (1 - \delta - \epsilon),$$

where $t_1 \cdot b = \delta \cdot \ell$.

The next goal is to study error-correcting codes to combat both under-shift and limited-over-shift errors. For this purpose we start by generalizing Lemma 7.

Lemma 8: Let $\boldsymbol{s}$ be a $(b,1)$-locally-constrained de Bruijn sequence over an alphabet of size q. Let $\boldsymbol{s}(\Delta^-, \Delta^+)$ be the sequence obtained from $\boldsymbol{s}$ after deleting symbols in locations specified by $\Delta^- = \{i_1, \ldots, i_t\}$ and insertion of symbols in locations specified by Δ^+. Assume further that $i_1 < \cdots < i_t$ and there are at most $b-2$ consecutive numbers in Δ^-. Then the sets Δ^- and Δ^+ are uniquely determined from $\boldsymbol{s}$ and $\boldsymbol{s}(\Delta^-, \Delta^+)$.

Proof: Since $\boldsymbol{s}$ is a $(b,1)$-locally-constrained de Bruijn sequence, it follows that between any two equal symbols there are at least $b-1$ different symbols. Hence, all the sticky-insertions can be located and corrected. Now, Lemma 7 can be applied to find the positions of the deletions, i.e., to determine Δ^-. Since the locations of the sticky-insertions are already known, it follows that Δ^- can be now determined. □

It is now straightforward to generalize Theorem 13. A q-ary b-limited t_1-burst-deletions sticky-insertions correcting code is a code over $\mathbb{F}_q$ which corrects any number of sticky-insertions and t_1 bursts of deletions, where each burst has length at most b.

Theorem 15: The code $\mathbb{C}_q(b,\ell,t)$ obtained in Construction 3 is a q-ary $(b-2)$-limited t_1-burst-deletions t_2-sticky-insertions correcting code where $t = t_1 \cdot (b-2)$ and t_2 is arbitrary.

Corollary 8: Given $0 < \delta, \epsilon < 1$, there exists a q-ary b-limited t_1-burst-deletions t_2-sticky-insertions correcting code of length ℓ, where $t_1 \cdot b = \delta \cdot \ell$ and t_2 is arbitrary, whose rate R satisfies

$$R \ge (1 - \log_q(b+2)) \cdot (1 - \delta - \epsilon).$$

It is clear that an upper bound on the maximum rate of our codes is at most $1-\delta$, Since ϵ is arbitrarily small, when b is small and $q = 2^m$ is large, it follows that the rates of our codes are close to the upper bounds and hence they are asymptotically optimal. A straightforward consequence from Corollary 8 is the following result.

Corollary 9: Given $0 < \delta, \epsilon < 1$, there exists a code correcting any number of under-shift errors and t_1 over-shift

errors, each of length at most b, with rate $R \geq \frac{m-\log_2(b+2)}{m}(1-\delta-\epsilon)$.

We finish this subsection with some remarks on the code $\mathbb{C}_q(b,\ell,t)$.

Remark 1:

- The code can correct not only a large (portion of the length n) number of deletions, but it can correct also an arbitrary (not depending on other parameters) number of sticky-insertions. It was proved by [47] that a code which can correct t deletions, can also correct any combination of up to t deletions and insertions. The rate of these codes correcting deletion/insertion errors is restricted by the number of errors. In the model we are considering, which includes sticky-insertion errors and bursts of deletions we construct an asymptotically optimal code whose rate does not depend on the number of sticky-insertions.
- Although, there is some prior research on codes correcting multiple deletions and codes correcting an arbitrary number of sticky-insertions, the current work is the first that studies codes correcting a combination of multiple deletions and an arbitrary number of sticky-insertions.

B. An Acyclic ℓ-Symbol Read Channel

In this subsection, we consider a slight modification of the ℓ-symbol read channel, where the symbols are not read cyclically. This acyclic ℓ-symbol read channel will be used in Section VII-C to correct synchronization errors in the racetrack memories with extra heads.

Definition 5: Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$ be a q-ary sequence of length n. In the *acyclic ℓ-symbol read channel*, if $\boldsymbol{x}$ is the codeword then the corresponding ℓ-symbol read sequence of $\boldsymbol{x}$ is

$$\pi_\ell(\boldsymbol{x}) = ((x_1, \ldots, x_\ell), (x_2, \ldots, x_{\ell+1}), \ldots, (x_n, \ldots, x_{n+\ell-1})),$$

where x_i is chosen arbitrarily for $i > n$.

We note that for each codeword $\boldsymbol{x}$, there are $q^{\ell-1}$ acyclic ℓ-symbol read sequences of $\boldsymbol{x}$ since for $n < i < n+\ell$, x_i can be any value in $\mathbb{F}_q$. The value of x_i for $i > n$ is not important since from any acyclic ℓ-symbol read sequence $\pi_\ell(\boldsymbol{x})$, we can recover the unique codeword $\boldsymbol{x}$. Hence, in this model, we may denote $x_i = *$ for $i > n$ where $*$ can be any value. In this channel, as in the cyclic ℓ-symbol read channel, two types of synchronization errors can occur: bursts of deletions where the length of each one is restricted to at most $b-2$ deletions and sticky-insertions.

Example 6: Let $\boldsymbol{x} = (0,1,0,0,1,0,0,0)$ be a stored information word. When $\ell = 2$, the 2-symbol read sequence of $\boldsymbol{x}$ is

$$\pi_2(\boldsymbol{x}) = ((0,1),(1,0),(0,0),(0,1),(1,0),(0,0),(0,0),(0,*)),$$

where $*$ can be any value. If there is a sticky-insertion at the second location and a burst of deletions of length two at the 6-th and 7th locations, we obtain the 2-symbol read sequence

$$\pi_2(\boldsymbol{y}) = ((0,1),(1,0),(1,0),(0,0),(0,1),(1,0),(0,*)).$$

Theorem 16: Let b and k be two positive integers such that $q^k \geq b \geq 2$, where $\ell = k+b-2$. If the first ℓ-symbol read is not deleted, then the code $\mathbb{C}_{DB}(n,b,k)$ can correct any number of sticky-insertions and any number of bursts of deletions of length at most $b-2$ in the acyclic ℓ-symbol read channel.

Proof: The proof is similar to the one of Theorem 12. Let $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$ be a codeword and

$$\pi_\ell(\boldsymbol{c}) = ((c_0, \ldots, c_{\ell-1}), (c_1, \ldots, c_\ell), \ldots, (c_{n-1}, \ldots, c_{n+\ell-2})),$$

be a corresponding ℓ-symbol read sequence of $\boldsymbol{c}$, where c_i can be arbitrary for $i \geq n$. Let

$$\pi_\ell(\boldsymbol{y}) = ((y_{1,1}, \ldots, y_{1,\ell}), (y_{2,1}, \ldots, y_{2,\ell}), \ldots, (y_{t,1}, \ldots, y_{t,\ell}))$$

be a received ℓ-symbol read sequence. We note that the first ℓ-tuple is correct, that is $(y_{1,1}, \ldots, y_{1,\ell}) = (c_0, \ldots, c_{\ell-1})$. For the second ℓ-tuple in the output $(y_{2,1}, \ldots, y_{2,\ell})$, we can determine the unique index $0 \leq i \leq b-1$ such that $(c_i, c_{i+1}, \ldots, c_{i+\ell-1}) = (y_{2,1}, \ldots, y_{2,\ell})$. Hence, we can recover the substring $(c_0, c_1, \ldots, c_{i+\ell-1})$. The same process can continue for the third ℓ-tuple and so on. For $0 \leq i \leq b-2$, there exists at least one ℓ-tuple $(c_{n-\ell+i}, \ldots, c_{n-1+i})$ which is not deleted. That is, there exists an index j such that $(c_{n-\ell+i}, \ldots, c_{n+i-1}) = (y_{j,1}, \ldots, y_{j,\ell})$. The above process can continue until $(y_{j,1}, \ldots, y_{j,\ell})$. Thus, we can obtain the sequence $(c_0, c_1, \ldots, c_{n+i-1})$. Since $i \geq 0$, we recover the original vector $\boldsymbol{c} = (c_0, \ldots, c_{n-1})$. Hence, the theorem is proved. □

Note, that from the proof of Theorem 16, we can find a simple decoding algorithm to recover the original sequence.

C. Correcting Errors in Racetrack Memories With Extra Heads

In this subsection, we present our last application for (b,k)-locally-constrained de Bruijn codes in the construction of codes correcting shift-errors in racetrack memories.

We present another way to combat under-shift and over-shift errors in racetrack memories by adding some consecutive extra heads next to the first head. For example, in Fig. 4, there are two extra heads next to the first head. We assume in this section that there are $\ell - 1$ extra heads. Since there are two types of heads, we call the $\ell - 1$ extra heads *secondary heads*, while the first m equally spaced heads are the *primary heads*. Hence, there are ℓ heads which read the first data segment together, the first primary head and all the $\ell - 1$ secondary heads. For $2 \leq i \leq m$, each other primary head will read one data segment individually. The output from the last $(m-1)$ primary heads is $\boldsymbol{c}[n+1, N] = (\boldsymbol{c}^2, \ldots, \boldsymbol{c}^m)$, where

$$\begin{pmatrix} \boldsymbol{c}^2 \\ \vdots \\ \boldsymbol{c}^m \end{pmatrix} = \begin{pmatrix} c_{2,1} & c_{2,2} & \cdots & c_{2,n-1} & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,n-1} & c_{m,n} \end{pmatrix}.$$

This matrix can be viewed as a q_2-ary word of length n where each column is a symbol in the alphabet of size $q_2 = 2^{m-1}$. In particular, let $\Phi_{m-1} : \mathbb{F}_2^{m-1} \mapsto \mathbb{F}_{q_2}$ be any bijection. For each column $\widehat{\boldsymbol{c}}_j = (c_{2,j}, \ldots, c_{m,j})^T$, $\Phi_{m-1}(\widehat{\boldsymbol{c}}_j) = v_j \in \mathbb{F}_{q_2}$, and define

$$\begin{aligned} \Phi(\boldsymbol{c}[n+1, N]) &\triangleq (\Phi_{m-1}(\widehat{\boldsymbol{c}}_1), \Phi_{m-1}(\widehat{\boldsymbol{c}}_2), \ldots, \Phi_{m-1}(\widehat{\boldsymbol{c}}_n)) \\ &= (v_1, \ldots, v_n) = \boldsymbol{v} \in \mathbb{F}_{q_2}^n . \end{aligned}$$

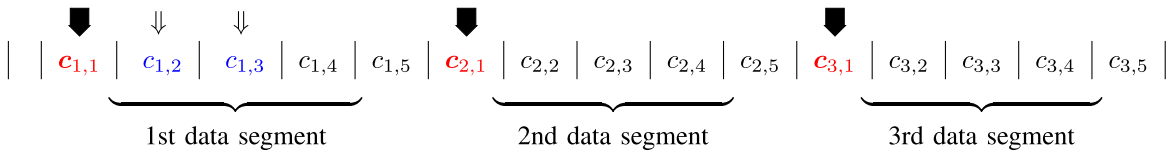

Fig. 4. Racetrack memory with two extra heads.

The output from all the ℓ heads in the first segment is

$$\begin{pmatrix} \boldsymbol{c}^{1,1} \\ \boldsymbol{c}^{1,2} \\ \vdots \\ \boldsymbol{c}^{1,\ell} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n-1} & c_{1,n} \\ c_{1,2} & c_{1,3} & \cdots & c_{1,n} & * \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{1,\ell} & c_{1,\ell+1} & \cdots & * & * \end{pmatrix}.$$

It is readily verified that this is the acyclic ℓ-symbol read sequence $\pi_\ell(\boldsymbol{c}[1,n])$ for the first data segment $\boldsymbol{c}[1,n] = (c_{1,1},\ldots,c_{1,n})$. If a single over-shift error has occurred, then a single column or an ℓ-tuple $(c_{1,i},\ldots,c_{1,\ell+i-1})$ was deleted. Similarly, if a single under-shift error has occurred, then a single column or an ℓ-tuple is repeated. Hence, using the results in Section VI, if $\boldsymbol{c}^1 = \boldsymbol{c}[1,n] = (c_1,\ldots,c_n) \in \mathbb{C}_{DB}(n,b,k)$ is a locally-constrained de Bruijn sequence, it is possible to locate and correct all sticky-insertions and multiple bursts of deletions of a limited length. Once we have located all errors, if $\boldsymbol{v}$, defined above, is a codeword that can correct multiple erasures, then we can correct multiple deletions and sticky-insertions. Following the idea in Construction 3, we can construct a code whose codewords have two components: a sequence to locate all errors and a sequence to correct multiple erasures. We are now ready to present such a construction as follows.

Construction 4: Let m,n,b,k,ℓ be positive integers such that $\ell = k+b-2$. Let $\mathbb{C}_{DB}(n,b,k)$ be a (n,b,k)-locally-constrained de Bruijn code. Let $\mathbb{C}_{q_2}(n,t)$ be a q_2-ary t-erasure-correcting code of length n, where $t=(b-2)\cdot t_1$, for some integer t_1 and $q_2 = 2^{m-1}$. Define,

$$\mathbb{C}_3(N,t_1,b-2) \triangleq \{(c_1,\ldots,c_N) \ : \ \boldsymbol{c}[1,n] \in \mathbb{C}_{DB}(n,b,k); \\ \Phi(\boldsymbol{c}[n+1,N]) \in \mathbb{C}_{q_2}(n,t)\}.$$

Theorem 17: The code $\mathbb{C}_3(N,t_1,b-2)$ has size $|\mathbb{C}_{DB}(n,b,k)|\cdot|\mathbb{C}_{q_2}(n,t)|$ and using $\ell-1=k+b-3$ extra heads, it can correct any number of sticky-insertions and t_1 bursts of deletions whose length is at most $b-2$.

Proof: The size of the code is an immediate observation from the definition of the code $\mathbb{C}_3(N,t_1,b-2)$. The first data segment of length n consists of any sequence $\boldsymbol{s}$ from $\mathbb{C}_{DB}(n,b,k)$. By Theorem 16, we can recover the sequence in the first data segment when there are any number of sticky-insertions and bursts of deletions of length at most $b-2$. Moreover, we can also determine the locations of these errors. In the output from the last $m-1$ heads, all sticky-insertions can be corrected easily and all deletions become erasures since we know the locations of these errors. There are at most $t=t_1\cdot(b-2)$ erasures and the decoding procedure of the code $\mathbb{C}_{q_2}(n,t)$ can be applied to correct these erasures. Thus, the sequence $\boldsymbol{s}$ can be recovered. Hence, the code $\mathbb{C}_3(N,t_1,b-2)$ can correct all sticky-insertions and at most t_1 bursts of deletions whose length is at most $b-2$. □

Corollary 10: Consider a racetrack memory comprising $N=m\cdot n$ cells and m primary heads which are equally spaced. Using $\ell-1=k+b-3$ extra secondary heads, it is possible to construct a code correcting a combination of any number of sticky-insertions and t_1 bursts of deletions whose length is at most $b-2$ such that its asymptotic rate satisfies

$$\lim_{N\to\infty}\frac{\log_2|\mathbb{C}_3(N,t_1,b-2)|}{N} \geq \frac{m-1}{m}\cdot(1-\delta-\epsilon)+\frac{R_{DB}(b,k)}{m}$$

where $\ell-b+2=k$, $t_1\cdot(b-2)=\delta\cdot n$, $0<\delta,\epsilon<1$, and $R_{DB}(b,k)$ is the maximal rate of (b,k)-locally-constrained de Bruijn code.

Proof: We note that there exists a code $\mathbb{C}_{q_2}(n,t)$ of length n correcting $t=t_1\cdot(b-2)$ erasures with the asymptotic rate at least

$$\lim_{n\to\infty}\frac{\log_{q_2}|\mathbb{C}_{q_2}(n,t)|}{n} \geq 1-\delta-\epsilon.$$

Since $q_2=2^{m-1}$ and $N=mn$, we have

$$\lim_{n\to\infty}\frac{\log_2|\mathbb{C}_{q_2}(n,t)|}{N} \geq \frac{m-1}{m}(1-\delta-\epsilon). \tag{3}$$

Moreover, the asymptotic rate of the locally-constrained de Bruijn code is

$$R_{DB}(b,k)=\lim_{n\to\infty}\frac{\log_2|\mathbb{C}_{DB}(n,b,k)|}{n}. \tag{4}$$

Therefore, by (3), (4), and Theorem 17, the rate of the code $\mathbb{C}_3(N,t_1,b-2)$ in Construction 4 can be computed as follows:

$$\begin{aligned}\lim_{N\to\infty}&\frac{\log_2|\mathbb{C}_3(N,t_1,b-2)|}{N} \\ &= \lim_{N\to\infty}\frac{\log_2(|\mathbb{C}_{q_2}(n,t)||\mathbb{C}_{DB}(n,b,k)|)}{N} \\ &\geq \frac{m-1}{m}(1-\delta-\epsilon)+\frac{R_{DB}(b,k)}{m}.\end{aligned}$$

□

The results of Corollary 10 can be compared to the results of Corollary 9. When $b=k=3$, by Table I, we have that $R_{DB}(3,3)\approx 0.7946$. Hence, using two more extra secondary heads, the asymptotic rate of the above code is $\frac{m-1}{m}(1-\delta-\epsilon)+\frac{0.7946}{m}=1-\delta-\epsilon+\frac{0.7946-1+\delta+\epsilon}{m}$. We note that, by Corollary 9, without using extra heads, the asymptotic rate of the codes constructed in Construction 3, $\frac{m-\log_2 b}{m}(1-\delta-\epsilon)$, is smaller than the asymptotic rate of the above codes. Hence, using some extra heads, we significantly improve the asymptotic rate of these codes. Furthermore, the theoretical maximal asymptotic rate is $1-\delta$. When $1-\delta<0.7946$, using two extra heads, the asymptotic rate of our constructed code is higher than the

theoretical maximal asymptotic rate without extra heads. Since $R_{DB}(n,k)$ tends to 1 when k is close to $\log n$, for any δ, using multiple extra heads, it is possible to construct codes whose asymptotic rate is higher than the theoretical maximal asymptotic rate without extra heads.

VIII. Conclusion and Open Problems

We have defined a new family of sequences and codes named locally-constrained de Bruijn sequences and locally-constrained de Bruijn codes. This family of sequences generalizes the family of de Bruijn sequences. These newly defined sequences have some constraints on the possible appearances of the same k-tuples in substrings of a bounded length. As such these sequences can be viewed also as constrained sequences and the related codes as constrained codes. Properties, constructions, and enumeration of the sequences were discussed, along with encoding and decoding algorithms for the related codes. We have demonstrated the use of locally-constrained de Bruijn sequences in combating synchronization errors in storage-related applications such as the ℓ-symbol read channel and racetrack memories.

The newly defined sequences raise many questions and directions for future research some of which are outlined.

1) Find more constructions for locally-constrained de Bruijn codes with new parameters and with larger rates. Especially, we are interested in more (b,k)-constrained de Bruijn codes for which b is about q^t and $k = c \cdot t$, where c is a small constant, and the rate of the code tends to 1 when k go to infinity.
2) Find better bounds (lower and upper) on the rates of locally-constrained de Bruijn codes with various parameters. Especially we want to find the exact rates for infinite families of parameters, where each family itself has an infinite set of parameters.
3) Find more applications for locally-constrained de Bruijn sequences and locally-constrained de Bruijn codes.

Acknowledgment

The authors would like to thank the associate editor and three anonymous reviewers which have done an incredible work in their careful review and insightful comments. These important comments and suggestions have helped us to improve the paper considerably.

References

[1] T. van Aardenne-Ehrenfest and N. G. de Bruijn, "Circuits and trees in oriented linear graphs," *Simon Stevin*, vol. 28, pp. 203–217, Jan. 1951.
[2] N. Alon, J. Edmonds, and M. Luby, "Linear time erasure codes with nearly optimal recovery," in *Proc. IEEE 36th Annu. Found. Comput. Sci.*, Milwaukee, WI, USA, Oct. 1995, pp. 512–519.
[3] Z. Barzilai, D. Coppersmith, and A. L. Rosenberg, "Exhaustive generation of bit patterns with applications to VLSI self-testing," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 190–194, Feb. 1983.
[4] V. E. Beneš, "Optimal rearrangeable multistage connecting networks," *Bell System Tech. J.*, vol. 43, no. 4, pp. 1641–1656, Jul. 1964.
[5] J. Brakensiek, V. Guruswami, and A. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, pp. 3403–3410, May 2018.
[6] A. M. Bruckstein, T. Etzion, R. Giryes, N. Gordon, R. J. Holt, and D. Shuldiner, "Simple and robust binary self-location patterns," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4884–4889, Jul. 2012.
[7] N. G. de Bruijn, "A combinatorial problem," in *Proc. Nederl. Akad. Wetensch.*, vol. 49, 1946, pp. 158–164.
[8] Y. Cassuto and M. Blaum, "Codes for symbol-pair read channels," *IEEE Trans. Inf. Theory*, vol. 57, no. 12, pp. 8011–8020, Dec. 2011.
[9] M. J. Chaisson, D. Brinza, and P. A. Pevzner, "De novo fragment assembly with short mate-paired reads: Does the read length matter?" *Genome Res.*, vol. 19, no. 2, pp. 336–346, Dec. 2008.
[10] A. H. Chan, R. A. Games, and E. L. Key, "On the complexities of de Bruijn sequences," *J. Combinat. Theory, A*, vol. 33, no. 3, pp. 233–246, Nov. 1982.
[11] Z. Chang, J. Chrisnata, M. F. Ezerman, and H. M. Kiah, "Rates of DNA sequence profiles for practical values of read lengths," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7166–7177, Nov. 2017.
[12] Y. M. Chee, T. Etzion, H. M. Kiah, V. Khu Vu, and E. Yaakobi, "Constrained de Bruijn codes and their applications," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 2369–2373.
[13] Y. M. Chee, H. M. Kiah, A. Vardy, E. Yaakobi, and V. K. Vu, "Codes correcting position errors in racetrack memories," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2017, pp. 161–165.
[14] Y. M. Chee, H. M. Kiah, A. Vardy, K. Van Vu, and E. Yaakobi, "Codes correcting limited-shift errors in racetrack memories," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 96–100.
[15] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Coding for racetrack memories," *IEEE Trans. Inf. Theory*, vol. 64, no. 11, pp. 7094–7112, Nov. 2018.
[16] Y. M. Chee, L. Ji, H. M. Kiah, C. Wang, and J. Yin, "Maximum distance separable codes for symbol-pair read channels," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7259–7267, Nov. 2013.
[17] L. Cheng, T. G. Swart, H. C. Ferreira, and K. A. S. Abdel-Ghaffar, "Codes for correcting three or more adjacent deletions or insertions," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 1246–1250.
[18] P. E. C. Compeau, P. A. Pevzner, and G. Tesler, "How to apply de Bruijn graphs to genome assembly," *Nature Biotechnol.*, vol. 29, no. 11, pp. 987–991, Nov. 2011.
[19] T. Cover, "Enumerative source coding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, Jan. 1973.
[20] L. Dolecek and V. Anantharam, "Repetition error correcting sets: Explicit constructions and prefixing methods," *SIAM J. Discrete Math.*, vol. 23, no. 4, pp. 2120–2146, 2010.
[21] T. Etzion, "Constructions for perfect maps and pseudorandom arrays," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1308–1316, Sep. 1988.
[22] T. Etzion, N. Kalouptsidis, N. Kolokotronis, K. Limniotis, and K. G. Paterson, "Properties of the error linear complexity spectrum," *IEEE Trans. Inf. Theory*, vol. 55, no. 10, pp. 4681–4686, Oct. 2009.
[23] T. Etzion and A. Lempel, "Algorithms for the generation of full-length shift-register sequences," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 3, pp. 480–484, May 1984.
[24] T. Etzion and A. Lempel, "Construction of de Bruijn sequences of minimal complexity," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 5, pp. 705–709, Sep. 1984.
[25] T. Etzion and A. Lempel, "An efficient algorithm for generating linear transformations in a shuffle-exchange network," *SIAM J. Comput.*, vol. 15, no. 1, pp. 216–221, Feb. 1986.
[26] C. Flye-Sainte Marie, "Solution to problem number 58," *I'Intermediare des Mathematiciens*, vol. 1, pp. 107–110, Jan. 1894.
[27] H. Fredricksen, "A survey of full length nonlinear shift register cycle algorithms," *SIAM Rev.*, vol. 24, no. 2, pp. 195–221, Apr. 1982.
[28] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded strings from multiset substring spectra," *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 2540–2544.
[29] R. Games and A. Chan, "A fast algorithm for determining the complexity of a binary sequence with period 2^n (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 1, pp. 144–146, Jan. 1983.
[30] S. W. Golomb, *Shift Register Sequences*. San Francisco, CA, USA: Holden Day, 1967.
[31] S. W. Golomb, *Digital Communication With Space Application*. Los Altos, CA, USA: Penisula, 1982.
[32] S. W. Golomb and G. Gong, *Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar*. Cambridge, U.K.: Cambridge Univ. Press, 2005.
[33] I. J. Good, "Normal recurring decimals," *J. London Math. Soc.*, vols. 21, no. 3, pp. 167–169, Jul. 1946.
[34] V. Guruswami and P. Indyk, "Linear-time encodable/decodable codes with near-optimal rate," *IEEE Trans. Inf. Theory*, vol. 51, no. 10, pp. 3393–3400, Oct. 2005.

[35] S. K. Hanna and S. E. Rouayheb, "Correcting bursty and localized deletions using guess & check codes," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2017, pp. 9–16.
[36] A. S. J. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Trans. Inf. Theory*, vol. 48, no. 1, pp. 305–308, Jan. 2002.
[37] Y. Hong, "Bounds of eigenvalues of graphs," *Discrete Math.*, vol. 123, nos. 1–3, pp. 65–74, Dec. 1993.
[38] Y.-C. Hsieh, "Decoding structured light patterns for three-dimensional imaging systems," *Pattern Recognit.*, vol. 34, no. 2, pp. 343–349, Feb. 2001.
[39] R. M. Idury and M. S. Waterman, "A new algorithm for DNA sequence assembly," *J. Comput. Biol.*, vol. 2, no. 2, pp. 291–306, Jan. 1995.
[40] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Duplication-correcting codes for data storage in the DNA of living organisms," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1028–1032.
[41] E. Key, "An analysis of the structure and complexity of nonlinear binary sequence generators," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 732–736, Nov. 1976.
[42] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3125–3146, Jun. 2016.
[43] A. Lempel, "On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers," *IEEE Trans. Comput.*, vol. C-19, no. 12, pp. 1204–1209, Dec. 1970.
[44] A. Lempel, "Cryptology in transition," *ACM Comput. Surveys*, vol. 11, no. 4, pp. 285–303, Dec. 1979.
[45] A. Lempel and M. Cohn, "Design of universal test sequences for VLSI," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 1, pp. 10–17, Jan. 1985.
[46] S. Li and Y. Tian, "Some bounds on the largest eigenvalues of graphs," *Appl. Math. Lett.*, vol. 25, no. 3, pp. 326–332, Mar. 2012.
[47] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Dokl. Akad. Nauk SSSR*, vol. 163, no. 8, pp. 845–848, 1965.
[48] X. Li and M. S. Waterman, "Estimating the repeat structure and length of DNA sequences using ℓ-tuples," *Genome Res.*, vol. 13, pp. 1916–1922, Aug. 2003.
[49] F. J. MacWilliams and N. J. A. Sloane, "Pseudo-random sequences and arrays," *Proc. IEEE*, vol. 64, no. 12, pp. 1715–1729, Dec. 1976.
[50] H. Mahdavifar and A. Vardy, "Asymptotically optimal sticky-insertion-correcting codes with efficient encoding and decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2683–2687.
[51] G. Mappouras, A. Vahid, R. Calderbank, and D. J. Sorin, "GreenFlag: Protecting 3D-racetrack memory from shift errors," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2019, pp. 1–12.
[52] B. H. Marcus, R. M. Roth, and P. H. Siegel, "Constrained systems and coding for recording channels," in *Handbook of Coding Theory*, V. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, 1998, ch. 20, pp. 1635–1764.
[53] U. M. Maurer, "Asymptotically-tight bounds on the number of cycles in generalized de Bruijn-Good graphs," *Discrete Appl. Math.*, vols. 37–38, pp. 421–436, Jul. 1992.
[54] R. A. Morano, C. Ozturk, R. Conn, S. Dubin, S. Zietz, and J. Nissano, "Structured light using pseudorandom codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 322–327, Mar. 1998.
[55] J. Pagès, J. Salvi, C. Collewet, and J. Forest, "Optimised de Bruijn patterns for one-shot shape acquisition," *Image Vis. Comput.*, vol. 23, no. 8, pp. 707–720, Aug. 2005.
[56] S. S. P. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, 2008.
[57] P. A. Pevzner, H. Tang, and M. S. Waterman, "A new approach to fragment assembly in DNA sequencing," in *Proc. PNAS*, vol. 98, 2001, pp. 9748–9753.
[58] F. Sala, R. Gabrys, C. Schoeny, and L. Dolecek, "Exact reconstruction from insertions in synchronization codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2428–2445, Apr. 2017.
[59] J. Salvi, S. Fernandez, T. Pribanic, and X. Llado, "A state of the art in structured light patterns for surface profilometry," *Pattern Recognit.*, vol. 43, no. 8, pp. 2666–2680, Aug. 2010.
[60] M. R. Samatham and D. K. Pradhan, "The de Bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI," *IEEE Trans. Comput.*, vol. 38, no. 4, pp. 567–581, Apr. 1989.
[61] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes for correcting a burst of deletions or insertions," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 1971–1985, Apr. 2016.
[62] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne, "A digital watermark," *Image Process. Proc.*, vol. 2, pp. 86–90, Nov. 1994.
[63] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, no. 2, pp. 153–161, Feb. 1971.
[64] Z. Sun, W. Wu, and H. Li, "Cross-layer racetrack memory design for ultra high density and low power consumption," in *Proc. 50th Annu. Design Autom. Conf. (DAC)*, 2013, pp. 1–6.
[65] A. Varma and C. S. Raghavendra, "Rearrangeability of multistage shuffle/exchange networks," *IEEE Trans. Commun.*, vol. 36, no. 10, pp. 1138–1147, Oct. 1988.
[66] A. J. V. Wijngaarden and K. A. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 200–207, Feb. 2010.
[67] E. Yaakobi, J. Bruck, and P. H. Siegel, "Constructions and decoding of cyclic codes over b-symbol read channels," *IEEE Trans. Inf. Theory*, vol. 62, pp. 1541–1551, 2016.
[68] C. Zhang *et al.*, "Hi-fi playback: Tolerating position errors in shift operations of racetrack memory," in *Proc. ACM/IEEE 42nd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jul. 2015, pp. 694–706.
[69] Y. Zhang and M. S. Waterman, "An Eulerian path approach to global multiple alignment for DNA sequences," *J. Comput. Biol.*, vol. 10, no. 6, pp. 803–819, Dec. 2003.

Yeow Meng Chee (Senior Member, IEEE) received the B.Math. degree in computer science and combinatorics and optimization and the M.Math. and Ph.D. degrees in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1988, 1989, and 1996, respectively. He was a Professor with the School of Physical and Mathematical Sciences and the Interim Dean of science with Nanyang Technological University. He is currently a Professor with the Department of Industrial Systems Engineering and Management and the Associate Vice President of innovation and enterprise with the National University of Singapore. His research interests include the interplay between combinatorics and computer science/engineering, particularly combinatorial design theory, coding theory, extremal set systems, and electronic design automation.

Tuvi Etzion (Fellow, IEEE) was born in Tel Aviv, Israel, in 1956. He received the B.A., M.Sc., and D.Sc. degrees from the Technion—Israel Institute of Technology, Haifa, Israel, in 1980, 1982, and 1984, respectively.

In 1984, he held a position with the Department of Computer Science, Technion, where he is currently Bernard Elkin Chair in computer science. From 1985 to 1987, he was a Visiting Research Professor with the Department of Electrical Engineering and Systems, University of Southern California, Los Angeles. During the summers of 1990 and 1991, he was a Visiting Bellcore, Morristown, NJ, USA. From 1994 to 1996, he was a Visiting Research Fellow with the Computer Science Department, Royal Holloway University of London, Egham, England. He also had several visits to the Coordinated Science Laboratory, University of Illinois in Urbana-Champaign, from 1995 to 1998, two visits to HP Bristol during the summers of 1996 and 2000, a few visits to the Department of Electrical Engineering, University of California at San Diego, from 2000 to 2017, several visits to the Mathematics Department, Royal Holloway University of London, from 2007 to 2017, a few visits to the School of Physical and Mathematical Science (SPMS), Nanyang Technological University, and to the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore, from 2016 to 2019, and a few visits to Jiaotong University, Beijing, from 2017 to 2019. His research interests include applications of discrete mathematics to problems in computer science and information theory, coding theory, coding for storage, and combinatorial designs.

Dr. Etzion was an Associate Editor for Coding Theory of IEEE TRANSACTIONS ON INFORMATION THEORY from 2006 to 2009. From 2004 to 2009, he was an Editor of the *Journal of Combinatorial Designs*. Since 2011, he has been an Editor of *Designs, Codes, and Cryptography*, and *Advances in Mathematics of Communications* since 2013. He has been the Editor-in-Chief of *Journal of Combinatorial Theory, Series A*, since 2021.

Han Mao Kiah (Member, IEEE) 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 at 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.

Sagi Marcovich (Member, IEEE) received the B.Sc. degree in software engineering from the Technion—Israel Institute of Technology, Haifa, Israel, in 2016, where he is currently pursuing the Ph.D. degree with the Computer Science Department. His research interests include algorithms, information theory, and coding theory with applications to DNA-based storage.

Alexander Vardy (Fellow, IEEE) was born in Moscow, Russia, in 1963. He received the B.Sc. degree *(summa cum laude)* from the Technion—Israel Institute of Technology, Israel, in 1985, and the Ph.D. degree from Tel-Aviv University, Israel, in 1991.

From 1985 to 1990, he was with Israeli Air Force, where he worked on electronic counter measures systems and algorithms. From 1992 to 1993, he was a Visiting Scientist with IBM Almaden Research Center, San Jose, CA, USA. From 1993 to 1998, he was with the University of Illinois at Urbana-Champaign, first as an Assistant Professor then as an Associate Professor. Since 1998, he has been with the University of California at San Diego (UCSD), where he is currently Jack Keil Wolf Chair Professor with the Department of Electrical and Computer Engineering and the Department of Computer Science. While on sabbatical from UCSD, he has held long-term visiting appointments with CNRS, France; EPFL, Switzerland; Technion—Israel Institute of Technology; and Nanyang Technological University, Singapore. His research interests include error-correcting codes, algebraic and iterative decoding algorithms, lattices and sphere packings, coding for storage systems, cryptography, computational complexity theory, and fun math problems.

Dr. Vardy has been a member of the Board of Governors of the IEEE Information Theory Society from 1998 to 2006 and from 2011 to 2017. In 1996, he was appointed as a fellow with the Center for Advanced Study, University of Illinois, and received the Xerox Award for Faculty Research. In 1996, he became a fellow of David and Lucile Packard Foundation. He received an IBM Invention Achievement Award in 1993 and NSF Research Initiation and CAREER Awards in 1994 and 1995. He received the IEEE Information Theory Society Paper Award (jointly with Ralf Koetter) in 2004. In 2005, he received the Fulbright Senior Scholar Fellowship and the Best Paper Award at the IEEE Symposium on Foundations of Computer Science (FOCS). In 2017, his work on polar codes was recognized by the IEEE Communications and Information Theory Societies Joint Paper Award. From 1995 to 1998, he was an Associate Editor of *Coding Theory*, From 1998 to 2001, he was the Editor-in-Chief of IEEE TRANSACTIONS ON INFORMATION THEORY.

Van Khu Vu (Member, IEEE) received the B.Sc. degree in mathematics from Vietnam National University (VNU), Hanoi, Vietnam, in 2010, and the Ph.D. degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2018. From 2010 to 2012, he was a Lecturer at VNU College of Sciences, Hanoi. From 2018 to 2019, he was a Research Fellow at the School of Physical and Mathematical Sciences, NTU. He is currently a Research Fellow at the Department of Industrial Systems Engineering and Management, National University of Singapore. His primary research interests include algorithms, combinatorics, and coding theory.

Eitan Yaakobi (Senior Member, IEEE) received the B.A. degree in computer science and mathematics and the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California at San Diego, in 2011. From 2011 to 2013, he was a Post-Doctoral Researcher with the Department of Electrical Engineering, California Institute of Technology, and the Center for Memory and Recording Research, University of California at San Diego. He is currently an Associate Professor at the Computer Science Department, Technion—Israel Institute of Technology. His research interests include information and coding theory with applications to non-volatile memories, associative memories, DNA storage, data storage, and retrieval.