Constrained de Bruijn Codes and their Applications

Yeow Meng Chee[*], **Tuvi Etzion**[§], **Han Mao Kiah**[†], **Van Khu Vu**[†], and **Eitan Yaakobi**[§]
[*] Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore
[†] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[§] Department of Computer Science, Technion — Israel Institute of Technology, Haifa, 3200003 Israel
Emails:pvocym@nus.edu.sg, {hmkiah,vankhu001}@ntu.edu.sg, {etzion,yaakobi}@cs.technion.ac.il

***Abstract*—A sequence $s = (s_1, \ldots, s_n)$ is called a (b, h)-constrained de Bruijn sequence if all substrings of length h starting within b consecutive positions are distinct. A set of (b, h)-constrained de Bruijn sequences is called a (b, h)-constrained de Bruijn code. A (b, h)-constrained de Bruijn sequence was constructed and used as a component of a code correcting multiple limited-shift-errors in racetrack memories. In this work, we show that a (b, h)-constrained de Bruijn code can correct deletions and sticky-insertions and also can determine the locations of these errors in an ℓ-symbol read channel. We also show that it is possible to use sequences from a (b, h)-constrained de Bruijn code to construct a code correcting shift-errors in racetrack memories. As a consequence, we improve the rates on previous known codes.**

It is shown in this work that a (b, h)-constrained de Bruijn code is a constrained code avoiding a set of specific patterns. Finally, we present some techniques to compute the maximum asymptotic rate and find some efficient encoding/decoding algorithms for (b, h)-constrained de Bruijn codes.

I. Introduction

A de Bruijn sequence is a cyclic sequence with the property that each possible h-tuple over an alphabet of size q, appears exactly once as a substring of h consecutive elements in the sequence [6]. This makes it possible to determine the exact location of any length h substring in the sequence. These sequences have many applications and are well-studied with many explicit constructions [8], [9], [13]. Recently, a generalization of de Bruijn sequences, called (b, h)-constrained (originally (b, h)-bounded) de Bruijn sequences, were introduced [4]. A (b, h)-*constrained de Bruijn sequence* is a (noncyclic) sequence with the property that all substrings of length h starting in b consecutive positions are distinct. Hence, the exact starting position of any length h substring, can be determined, assuming that this position is in a given segment of length b. In a previous work [4], such a constrained de Bruijn sequence (called a bounded de Bruijn sequence) serves as a marker at specific positions of all codewords and is essential in correcting shift-errors in racetrack memories [4]. Although there is some abuse of terminology, we prefer to continue and use the name of de Bruijn for our sequences and related codes. In the current work, instead of a single sequence, we study the construction of a set of such constrained de Bruijn sequences. As a consequence, we improve the rates of the previous known racetrack codes. Furthermore, this code can correct deletions and sticky-insertions and it can also determine the locations of these errors in the ℓ-symbol read channel.

Symbol-pair read channel and its generalization, ℓ-symbol read channel, have been presented and investigated recently [2], [18]. In this channel, each ℓ bits are read together as an ℓ-tuple. In contrast to the channel where each bit is read individually, this channel has some natural redundancies that help to correct possible errors. In the current literature, only substitution errors have been studied for the ℓ-symbol read channel. In this work, we correct *synchronization errors* in the ℓ-symbol read channel (defined in Section IV). An example of an ℓ-symbol read channel, where synchronization errors can occur, is a racetrack memory with several extra heads. These synchronization errors in the racetrack memory channel are shift errors.

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 [14], [15]. It is composed of *domains*, also known as *cells*, which can be used to store information. The reading mechanism is operated by many *read ports*, called *heads*. All these heads are designed to be at fixed positions. These heads are normally distributed uniformly, but we may add extra $(\ell - 1)$ heads in consecutive positions [3], [4], [19]. Hence, ℓ consecutive bits are read together using these extra heads. These consecutive heads form an ℓ-symbol read channel. To read the information, all domains have to be shifted together. However, this shift operation might not work perfectly and thus some shift-errors (called position errors in [19]) might occur. Two types of shift-errors in racetrack memory are under-shift and over-shift which can be modelled as sticky-insertion and deletion, respectively.

To combat these shift-errors using a single head, Vahid et al. [16] recently studied codes correcting two deletions and insertions. We also can use other classical codes correcting deletions and/or sticky-insertions [1], [7], [11]. But, it is difficult to construct these codes with high rate. However, leveraging the special feature of racetrack memory of having multiple heads, there are a few coding schemes to combat shift-errors [3], [4], [19]. In [4], a construction of asymptotically optimal codes correcting any number of under-shifts (sticky-insertions) and multiple limited-over-shifts (burst of deletions of limited length) using several extra heads has been proposed. In this construction, a (b, h)-constrained de Bruijn sequence is an important component. In the current work, we show how to use a (b, h)-constrained de Bruijn code in order to improve the rate of the previous result from [4].

In view of these applications, in this work we study enumeration and constructions of constrained de Bruijn codes. We demonstrate that these codes are constrained codes and use this property for enumeration techniques and for providing efficient encoding/decoding algorithms of some codes while maximizing the rate. For lack of space, the complete details of some proofs will be presented only in the full version of this paper.

II. Notations and Definitions

Let $\mathbb{F}_q$ denote the q-ary finite field and $[n]$ denote the set $\{1, 2, \ldots, n\}$. For each sequence $\boldsymbol{u} = (u_1, \ldots, u_n) \in \mathbb{F}_q^n$, let $\boldsymbol{u}[i_1, i_2] = (u_{i_1}, u_{i_1+1}, \ldots, u_{i_2}) \in \mathbb{F}_q^{i_2-i_1+1}$, $1 \leqslant i_1 \leqslant i_2 \leqslant n$, denote a substring of $\boldsymbol{u}$ whose length is $i_2 - i_1 + 1$. In case $i_1 = i_2 = i$, we denote the substring $\boldsymbol{u}[i, i]$ by $\boldsymbol{u}[i]$ to specify the i-th symbol u_i of the sequence $\boldsymbol{u}$. A q-ary code $\mathbb{C}$ of length n is a set of q-ary sequences of length n, that is $\mathbb{C} \subseteq \mathbb{F}_q^n$. For each code $\mathbb{C}$ of length n, we define the rate of the code $\mathbb{C}$ to be $R(\mathbb{C}) = \log_q(|\mathbb{C}|)/n$, and the redundancy of the code $\mathbb{C}$ to be $r(\mathbb{C}) = n - \log_q(|\mathbb{C}|)$, where $|\mathbb{C}|$ is the size of the code $\mathbb{C}$.

Let $\mathcal{F}$ be a set of sequences over $\mathbb{F}_q$. A sequence $\boldsymbol{u}$ is said to *avoid* $\mathcal{F}$ or $\mathcal{F}$*-avoiding* if no sequence in $\mathcal{F}$ is a substring of $\boldsymbol{u}$. We denote the set of all q-ary sequences of length n which avoid $\mathcal{F}$ by $\mathcal{A}(n; \mathcal{F})$. A set of $\mathcal{F}$-avoiding sequences of length n is called an $\mathcal{F}$-avoiding code of length n and is denoted by $\mathbb{C}(n; \mathcal{F})$, i.e., $\mathbb{C}(n; \mathcal{F}) \subseteq \mathcal{A}(n; \mathcal{F})$.

Definition 1.

- *A sequence* $\boldsymbol{s} = (s_1, \ldots, s_n)$ *over* $\mathbb{F}_q$ *is called a* (b,h)-***constrained de Bruijn sequence*** *if* $\boldsymbol{s}[i, i+h-1] \neq \boldsymbol{s}[j, j+h-1]$ *for all* $i, j \in [n-h+1]$ *such that* $|i-j| \leqslant b-1$.
- *A set of* (b,h)*-constrained de Bruijn sequences of length* n *is called a* (b,h)***-constrained de Bruijn code****. The set of all* (b,h)*-constrained de Bruijn sequences of length* n *is denoted by* $\mathbb{C}_{DB}(n,b,h)$.

Next, we introduce some constrained codes which are closely related to constrained de Bruijn codes.

Definition 2.

- *A sequence* $\boldsymbol{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{F}_q^n$ *is a* ***period*** p ***sequence*** *if it satisfies* $s_i = s_{i+p}$ *for all* $1 \leqslant i \leqslant n-p$.
- *A sequence* $\boldsymbol{s} \in \mathbb{F}_q^n$ *is called an* m***-limited length, period*** p ***sequence*** *if any period* p *subsequence of* $\boldsymbol{s}$ *has length at most* m.
- *A set of* m*-limited length, period* p *sequences from* $\mathbb{F}_q^n$ *is called an* m***-limited length, period*** p ***code****. The set of all such* m*-limited length, period* p *sequences is denoted by* $\mathbb{C}_{LP}(n,m,p)$.

The first lemma is a straightforward observation.

Lemma 3. *If* $\mathcal{F}$ *is the set of all period* p *sequences of length* $m+1$*, then* $\mathbb{C}_{LP}(n,i,p)$ *is an* $\mathcal{F}$*-avoiding code for each* $1 \leqslant i \leqslant m$.

The code $\mathbb{C}_{LP}(n,m,p)$ was introduced and studied recently [3] and was used in coding for racetrack memories.

III. Constrained de Bruijn Codes

Let $\mathcal{F}_{p,p+h}$ be the set of all period p sequences of length $p+h$ for any given $1 \leqslant p \leqslant b-1$ and let $\mathcal{F} = \cup_{p=1}^{b-1} \mathcal{F}_{p,p+h}$. The following result implies a strong relation between constrained de Bruijn codes, limited length, period codes, and $\mathcal{F}$-avoiding codes.

Theorem 4. *For all given admissible* n, b, h*, and any code* $\mathbb{C} \subset \mathbb{F}_q^n$*, the following three statements are equivalent*

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

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

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

Finally, if $\mathbb{C} \subseteq \bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+h-1, i)$, then by Lemma 3 we have that $\mathbb{C} \subseteq \mathcal{A}(n;\mathcal{F})$. ∎

Corollary 5. *For all given admissible* n, b, h*,*

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

We define the *maximum asymptotic rate* of (b,h)-constrained de Bruijn codes to be $R_{DB}(b,h) = \limsup_{n\to\infty} \frac{\log|\mathbb{C}_{DB}(n,b,h)|}{n}$.[1] We observe that the asymptotic rate $R_{DB}(b,h)$ is 1 only if h tends to infinity. In this case, we are interested in the redundancy of the code.

Theorem 6. *For all* $q,\ n,\ b$*, and* h*,*

$$|\mathbb{C}_{DB}(n,b,h)| \geqslant q^n \left(1 - (b-1)n \cdot \left(\frac{1}{q}\right)^h\right).$$

In particular, for $h \geqslant \lceil \log_q n + \log_q(b-1) \rceil + 1$*, the redundancy of* $\mathbb{C}_{DB}(n,b,h)$ *is at most a single bit.*

Proof: By Corollary 5, $\mathbb{C}_{DB}(n,b,h) = \bigcap_{i=1}^{b-1} \mathbb{C}_{LP}(n, i+h-1, i)$, which implies

$$|\mathbb{C}_{DB}(n,b,h)| = q^n - |\mathbb{C}_{DB}(n,b,h)^c| \geqslant q^n - \sum_{i=1}^{b-1} |\mathbb{C}_{LP}(n, i+h-1, i)^c|, \tag{1}$$

where A^c denotes the complement of the set A. The number of words in $\mathbb{C}_{LP}(n, i+h-1, i)^c$ is upper bounded by

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

Hence,

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

and therefore using (1) we have,

$$|\mathbb{C}_{DB}(n,b,h)| \geqslant q^n \left(1 - (b-1) \cdot n \cdot \left(\frac{1}{q}\right)^h\right).$$

In particular, for $h \geqslant \lceil \log_q n + \log_q(b-1) \rceil + 1$, we obtain

$$|\mathbb{C}_{DB}(n,b,h)| \geqslant (q-1) \cdot q^{n-1}.$$

Thus, the redundancy of $\mathbb{C}_{DB}(n,b,h)$ is at most a single bit. ∎

To encode the (b,h)-constrained de Bruijn code efficiently with only a single bit of redundancy, we may use the *sequence replacement techniques* [17].

For any h, we observe that $b \leqslant 2^h$ and $R_{DB}(b,h) < 1$. To determine exactly the maximum asymptotic rates of (b,h)-constrained de Bruijn codes, we may use the well-known Perron-Frobenius theory [12]. When b and h are given, taking the de Bruijn graph, we can build a finite directed graph with labelled edges such that paths in the graph generate exactly all (b,h)-constrained de Bruijn sequences. An example for the case $(b,h) = (3,3)$ is shown in Fig. 1. Its adjacency matrix is

$$A_G = \begin{pmatrix} 0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&0&0 \\ 1&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&1 \\ 0&0&1&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \end{pmatrix}$$

[1] The lim sup can indeed be replaced by a proper lim [12].

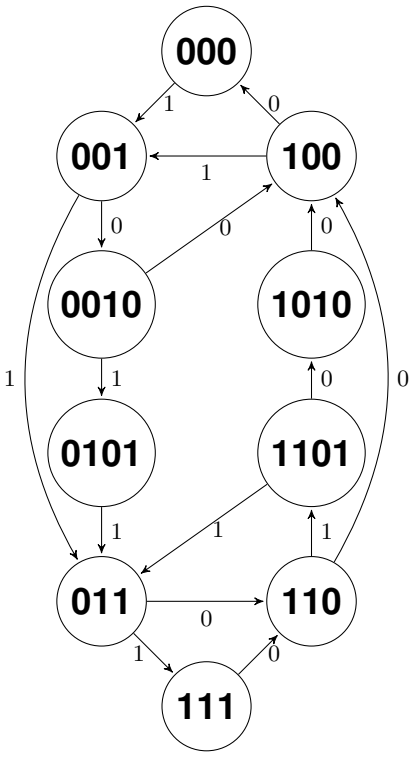

Fig. 1: Labelled graph G of the $(3,3)$-constrained de Bruijn code.

Its largest eigenvalue is $\lambda \approx 1.73459$. So, the capacity of this constrained system, which is the maximum asymptotic rate of $(3,3)$-constrained de Bruijn code, is $\log \lambda = 0.7946$.

Similary, we can compute the maximum asymptotic rates of (b,h)-constrained de Bruijn codes for some values of b,h and tabulate these results in Table I.

TABLE I: The maximum asymptotic rates of (b,h)-constrained de Bruijn codes.

	$h=2$	$h=3$	$h=4$	$h=5$	$h=6$
$b=2$	0.6942	0.8791	0.9468	0.9752	0.9881
$b=3$	0.4056	0.7946	0.9146	0.9614	0.9817
$b=4$	0	0.634	0.860	0.939	0.965

To encode efficiently binary (b,h)-constrained de Bruijn codes which attain the maximum asymptotic rate, we may build *finite state encoders* [12]. The related algorithms and the analysis of their complexity can be found in [12].

Although Perron-Frobenius theory should be sufficient to compute the maximum asymptotic rates of (b,h)-constrained de Bruijn codes for any given pair (b,h), it is more complicated to compute the rates when these values are very large.

Next we present a technique to compute the maximum asymptotic rate of constrained de Bruijn codes explicitly for all h.

Theorem 7. *For any positive integer $h>1$, the maximum asymptotic rate of $(3,h)$-constrained de Bruijn code is $\log_2 \lambda$ where λ is the largest root of the following equation:* $x^{2h-1} = x^{2h-3} + 2x^{2h-4} + \cdots + (h-2)x^h + (h-1)x^{h-1} + (h-1)x^{h-2} + (h-2)x^{h-3} + \cdots + 2x + 1$.

Proof: Recall that $\mathcal{F}_{p,p+h}$ is the set of all period p sequences of length $p+h$. Let $\mathcal{F}_1 = \mathcal{F}_{1,h+1} \cup \mathcal{F}_{2,h+2}$, i.e., $\mathcal{F}_1$ contains the all-zero word of length $h+1$, the all-one word of length $h+1$, and the two words of length $h+2$ which have in each two consecutive positions the patterns 01 or 10. By Corollary 5, $\mathbb{C}_{DB}(n,3,h) = \mathcal{A}(n,\mathcal{F}_1)$, i.e. binary $(3,h)$-constrained de Bruijn code of length n is an $\mathcal{F}_1$-avoiding code of length n.

Let $\mathcal{F}_2$ by the set which contains the all-zero word of length h and the all-one word of length $h+1$. We define the $\mathbf{D}$-morphism defined first in [10], $\mathbf{D} : B^n \mapsto B^{n-1}$, $B = \{0,1\}$, where $\mathbf{D}(\boldsymbol{x}) = \mathbf{D}(x_1, x_2, \ldots, x_n) = \boldsymbol{y} = (y_2, \ldots, y_n)$, with $y_i = x_i + x_{i-1}$, $2 \leqslant i \leqslant n$. It was proved in [10] 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

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

Hence, $\mathcal{A}(n;\mathcal{F}_1)$ and $\mathcal{A}(n;\mathcal{F}_2)$ have the same maximum asymptotic rate.

To continue the proof of the theorem, we now compute the maximum asymptotic rate of $\mathcal{A}(n;\mathcal{F}_2)$. 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*. 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*. It is easy to verify that 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 the same. Let

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

be the mapping such that, for each $\boldsymbol{x} = (0, x_2, \ldots, x_n) \in \mathcal{A}_0(n;\mathcal{F}_2)$ we have that $\Phi_1(\boldsymbol{x}) = (x_{i+1}, \ldots, x_n) \in \mathcal{A}_1(n-i;\mathcal{F}_2)$, 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 h, it follows that the mapping Φ_1 is a well-defined bijection. Therefore,

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

Similarly, we can define the bijection

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

and obtain the equality

$$|\mathcal{A}_1(n;\mathcal{F}_2)| = \sum_{i=1}^{h} |\mathcal{A}_0(n-i;\mathcal{F}_2)|. \tag{3}$$

From (2) and (3), we obtain

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

$$= \sum_{\ell=2}^{h} (\ell-1)|\mathcal{A}_0(n-\ell;\mathcal{F}_2)| + \sum_{\ell=1}^{h-1} \ell|\mathcal{A}_0(n-2h+\ell;\mathcal{F}_2)|$$

$$= |\mathcal{A}_0(n-2;\mathcal{F}_2)| + 2|\mathcal{A}_0(n-3;\mathcal{F}_2)| + \cdots$$

$$+ (h-1)|\mathcal{A}_0(n-h;\mathcal{F}_2)| + (h-1)|\mathcal{A}_0(n-h-1;\mathcal{F}_2)| +$$

$$+ (h-2)|\mathcal{A}_0(n-h-2;\mathcal{F}_2)| + \cdots + |\mathcal{A}_0(n-2h+1;\mathcal{F}_2)|.$$

It is again easy to verify that the maximum asymptotic rates of $\mathcal{A}_0(n-\ell;\mathcal{F}_2)$ for all $2 \leqslant \ell \leqslant 2h-1$ 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 $x^{2h-1} = x^{2h-3} + 2x^{2h-4} + \cdots + (h-2)x^h + (h-1)x^{h-1} + (h-1)x^{h-2} + (h-2)x^{h-3} + \cdots + 2x + 1$. ∎

Using the above recursive formulas in Equations (2) and (3), 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 technique [5].

IV. The ℓ-Symbol Read Channel with Synchronization Errors

In this section, we show that (b,h)-constrained de Bruijn codes can correct synchronization errors such as deletions and sticky-insertions in the ℓ-symbol read channel [2], [18].

Definition 8. *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 the stored information 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 for $i > n$ can be any value.

In this channel, when we receive the ℓ-symbol read sequence $\pi_\ell(\boldsymbol{x})$ correctly, it is easy to get the stored sequence $\boldsymbol{x}$. However, several types of errors, such as substitution errors and synchronization errors, can occur. Here, we focus on the ℓ-symbol read channel with 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. A deletion is the error event when an ℓ-tuple is deleted. A burst of deletions of length at most b is an error when at most b consecutive ℓ-tuples are deleted.

Example 1. Let $\boldsymbol{x} = (0,1,0,0,1,0,0,0)$ be a stored sequence. 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 location $i = 2$ and a burst of deletions of length 2 at locations 6 and 7, we obtain the 2-symbol read sequence $\pi_2(\boldsymbol{y}) = ((0,1),(1,0),(1,0),(0,0),(0,1),(1,0),(0,*))$. □

The goal of this section is to construct codes correcting sticky-insertions and bursts of deletions in the ℓ-symbol read channel. In this channel, it is assumed that the first ℓ-tuple is read correctly[2].

Theorem 9. *Let b and h be two positive integers such that $2^h \geqslant b \geqslant 2$. Let $h + b - 2 = \ell$. The code $\mathbb{C}_{DB}(n, b, h)$ can correct any number of sticky-insertions and any number of bursts of deletions of length at most $b - 2$ in the ℓ-symbol read channel.*

Proof: Let $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ be a stored sequence and $\pi_\ell(\boldsymbol{y}) = ((y_{1,1}, \ldots, y_{1,\ell}), (y_{2,1}, \ldots, y_{2,\ell}), \ldots, (y_{k,1}, \ldots, y_{k,\ell}))$ be a received ℓ-symbol read sequence. Recall that the first ℓ-tuple is always correct, that is $(y_{1,1}, \ldots, y_{1,\ell}) = (c_1, \ldots, c_\ell)$. We now consider the second ℓ-tuple $(y_{2,1}, \ldots, y_{2,\ell})$. If the second ℓ-tuple is not correct, the error can be only a sticky insertion or a burst of deletions of length at most $b - 2$. Hence, there are b possibilities for the second ℓ-tuple. That is, there exists $1 \leqslant i \leqslant b$ such that $(y_{2,1}, \ldots, y_{2,\ell}) = \boldsymbol{c}[i, i + \ell - 1]$. Hence, there exists $1 \leqslant i \leqslant b$ such that $(y_{2,1}, \ldots, y_{2,h}) = \boldsymbol{c}[i, i + h - 1]$ since $h \leqslant \ell$. Furthermore, $\boldsymbol{c}[i, i + h - 1] \neq \boldsymbol{c}[j, j + h - 1]$ for all $1 \leqslant i, j \leqslant b$ such that $i \neq j$ since $\boldsymbol{c} \in \mathbb{C}_{DB}(q, b, h)$. Hence, there is exactly one index $i_0 \in [b]$ for which $(y_{2,1}, \ldots, y_{2,h}) = \boldsymbol{c}[i_0, i_0 + h - 1]$ and thus $(y_{2,1}, \ldots, y_{2,\ell}) = \boldsymbol{c}[i_0, i_0 + \ell - 1]$.

The index i_0 is determined as follows. We note that $\boldsymbol{c}[i, i + h - 1] = (y_{1,i}, \ldots, y_{1,i+h-1})$ for all $1 \leqslant i \leqslant b - 1$ since $(c_1, \ldots, c_\ell) = (y_{1,1}, \ldots, y_{1,\ell})$. If $\boldsymbol{c}[i, i+h-1] \neq (y_{2,1}, \ldots, y_{2,h})$ for all $1 \leqslant i \leqslant b - 1$ then $i_0 = b$. Otherwise, i_0 is the unique index in $[b - 1]$ such that $\boldsymbol{c}[i_0, i_0 + h - 1] = (y_{2,1}, \ldots, y_{2,h})$.

If $i_0 = 1$, then the error is a sticky-insertion. To correct this error, it is only required to remove the repeated ℓ-tuple. If $i_0 = 2$, then there is no error and the second tuple is correct. If $3 \leqslant i_0 \leqslant b$, then there is a burst of deletions of length $i_0 - 2$. To correct this error, we add $(i_0 - 2)$ ℓ-tuples $\boldsymbol{c}[j, j + \ell - 1]$ for $j = 2, \ldots, i_0 - 1$. So, we not only determine the location of synchronization error but also correct this error to get the first i_0 ℓ-tuples as well as the first $i_0 + \ell - 1$ symbols exactly. We now consider the $(i_0 + 1)$-st tuple which is $(y_{3,1}, \ldots, y_{3,\ell})$ and repeat the same procedure to determine the error locations and types and correct them. The process will end when $i_0 \geqslant n - \ell + 1$, that is, when all n symbols are obtained correctly. ■

From the above proof, we can derive an efficient decoding algorithm to recover the stored sequence $\boldsymbol{x}$ in $\Theta(n)$ time, for all fixed b and h.

[2]This requirement is needed since the first bit is read only once. In case the ℓ-symbol read sequence is defined cyclically this assumption is not necessary.

V. Racetrack Memories with Extra Heads

In this section, we will show the application of (b, h)-constrained de Bruijn codes in the construction of codes correcting shift-errors in racetrack memory. Let us review the model of racetrack memories with extra heads and previous results [4].

Let N, n, m be three positive integers such that $N = n \cdot m$. A racetrack memory comprises of N cells and m heads which are distributed uniformly. Each head will read a segment of n cells. For example, in Fig. 2, a 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 $\boldsymbol{c}^i = (c_{i,1}, \ldots, c_{i,n})$ where $c_{i,j} = c_{(i-1)\cdot n+j}$ for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$. Hence, the output matrix from all m heads (without error) is:

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

It is shown that when an under-shift occurs, one column in the above matrix is repeated. When an over-shift occurs, one or a few consecutive columns in the matrix are deleted. Our goal is to combat these shift-errors in racetrack memories. Several constructions of codes correcting multiple bursts of deletions of length at most b and multiple sticky-insertions have been proposed in [4] to correct these shift-errors. Another approach which has been proposed in [4] is to add some extra heads such that it is possible to construct a code with better rate.

We now focus on the model with some consecutive extra heads next to the first head. For example, in Fig. 2, 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 heads are the *primary heads*. Hence, there are ℓ heads which read the first data segment together, the first primary head and all $\ell - 1$ secondary heads. For $2 \leqslant i \leqslant m$, each other primary head will read one data segment individually. So, 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 the map $\Phi_{m-1} : \{0,1\}^{m-1} \mapsto \mathbb{F}_{q_2}$ be any bijection. For each column $\widehat{\boldsymbol{c}}_j = (c_{2,j}, \ldots, c_{m,j})$, $\Phi_{m-1}(\widehat{\boldsymbol{c}}_j) = v_j \in \mathbb{F}_{q_2}$, and together $\Phi(\boldsymbol{c}[n + 1, N]) = \boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{F}_{q_2}^n$.

The output from all ℓ heads in the first segment is

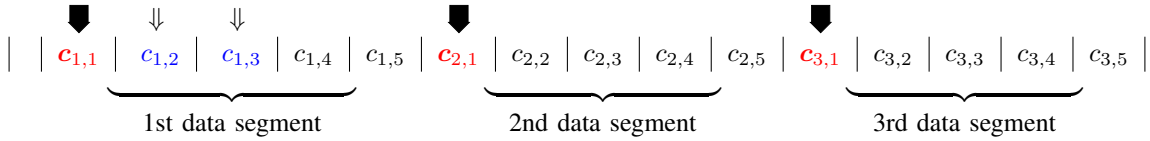

Fig. 2: Racetrack memory with two extra heads

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

In fact, this is the ℓ-symbol read sequence $\pi_\ell(\boldsymbol{c}[1,n])$ of the first data segment $\boldsymbol{c}[1,n] = (c_{1,1}, \dots, c_{1,n})$.

Under this setup, there exists a construction of codes correcting any number of sticky-insertions and multiple bursts of deletions where the first data segment can determine the locations of the synchronization errors and the last $(m-1)$ data segments can correct multiple erasures [4, Construction 12]. Therefore, the first data segment is chosen as a fixed constrained de Bruijn sequence and the last $(m-1)$ data segments are chosen from all words in some erasure-correcting-code. In that construction, the first data segment is used only to determine the locations of the synchronization errors and does not store any information. This construction is improved by showing that it is possible to choose the first data segment from all sequences from a constrained de Bruijn code. When there are synchronization errors, we can recover the sequence and also determine the locations of the synchronization errors. Thus, it is possible to store information in the first segment and thereby improve the rate of the codes from [4, Construction 12].

Construction 10. *Let m, n, b, h, ℓ be positive integers such that $h + b - 2 = \ell$, $\mathbb{C}_{DB}(n, b, h)$ be a binary (b, h)-constrained de Bruijn code, and $\boldsymbol{c}^1 = \boldsymbol{c}[1,n] = (c_{1,1}, \dots, c_{1,n}) \in \mathbb{C}_{DB}(n, b, h)$. Let $\mathbb{C}_{q_2}(n, t)$ be a q_2-ary t-erasure-correcting code of length n, where $t = (b-2) \cdot t_1$ and $q_2 = 2^{m-1}$, and let $\boldsymbol{c}[n+1, N] = (\boldsymbol{c}^2, \dots, \boldsymbol{c}^m)$; where $\boldsymbol{v} = (v_1, \dots, v_n) \in \mathbb{C}_{q_2}(n, t)$ (see the related definition of $\boldsymbol{v}$). Define*

$$\mathbb{C}_{RM}(N, t_1, b-2) \stackrel{\text{def}}{=} \{(\alpha_1, \alpha_2) \,:\, \alpha_1 \in \boldsymbol{c}[1,n],\ \alpha_2 \in \boldsymbol{c}[n+1, N]\}.$$

The code $\mathbb{C}_{RM}(N, t_1, b-2)$ can correct any number of sticky-insertions and t_1 bursts of deletions of length at most $b-2$ using $\ell - 1$ extra heads.

Construction 10 improves upon Construction 12 in [4]. In the first data segment, any word from the constrained de Bruijn code can be stored instead of choosing the fixed constrained de Bruijn sequence. From Theorem 9, we can recover the stored sequence in the first data segment when there are sticky-insertions and bursts of deletions of length at most $b-2$ and 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. The code $\mathbb{C}_{q_2}(n, t)$ can correct these errors and this way recover the original sequence $\boldsymbol{v}$. So, the code $\mathbb{C}_{RM}(N, t_1, b-2)$ can correct all sticky-insertions and at most t_1 bursts of deletions of length at most $b-2$ to recover the stored sequence $\boldsymbol{c}$.

The size of this code is

$$|\mathbb{C}_{RM}(N, t_1, b-2)| = |\mathbb{C}_{DB}(n, b, h)| \cdot |\mathbb{C}_{q_2}(n, t)|.$$

Theorem 11. *Consider a racetrack memory comprising of $N = m \cdot n$ cells and m primary heads which are distributed uniformly. Using $\ell - 1$ extra secondary heads, it is possible to construct a code $\mathbb{C}_{RM}(N, t_1, b-2)$ correcting a combination of any number of sticky-insertions and t_1 bursts of deletions of length at most $b-2$ such that its asymptotic rate satisfies*

$$\lim_{N\to\infty} \frac{\log |\mathbb{C}_{RM}(N, t_1, b-2)|}{N} \geqslant \frac{m-1}{m} \cdot (1 - \delta - \epsilon) + \frac{R_{DB}(b,h)}{m}$$

where $\ell - b + 2 = h$, $t_1 \cdot (b-2) = \delta \cdot n$ and $1 > \delta, \epsilon > 0$.

Compared to the previous result in [4, Theorem 3], we can construct a code with higher rate using a few more extra heads. For example, in the case $b = h = 3$, according to the results from Section III we get that $R_{DB}(3,3) \approx 0.7946$. So, using two more extra heads, the asymptotic rate of these codes is $0.7946/m$ higher than the asymptotic rate of the codes from [4].

References

[1] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, pp. 3403–3410, 2018.

[2] Y. Cassuto and M. Blaum, "Codes for symbol-pair read channels," *IEEE Trans. Inform. Theory*, vol. 57, no. 12, pp. 8011–8020, 2011.

[3] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Coding for racetrack memories," *IEEE Trans. Inform. Theory,* vol. 64, no. 11, pp. 7094–7112, 2018.

[4] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Codes correcting limited-shift errors in racetrack memories," *Proc. IEEE Int. Symp. Inform. Theory,* pp. 96–100, USA, Jun. 2018.

[5] T. Cover, "Enumerative source coding," *IEEE Trans. Inform. Theory,* vol. 19, no. 1, pp. 73–77, 1973.

[6] N. G. De Bruijn, "A combinatorial problem," *Koninklijke Nederlandse Akademie v. Wetenschappen* vol. 49, pp. 758–764, 1946.

[7] 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.

[8] T. Etzion and A. Lempel, "Algorithms for the generation of full-length shift-register sequences," *IEEE Trans. on Information Theory,* vol. 30, pp. 480–484, 1984.

[9] H. Fredricksen, "A survey of full length nonlinear shift register cycle algorithms," *SIAM Review*, vol. 24, pp. 195–221, 1982.

[10] A. Lempel, "On a homomorphism of de Bruijn graph and its applications to the design of feedback shift registers", *IEEE Trans. Computers*, vol. 19, pp. 1204–1209, 1970.

[11] V. I. Levenshtein, "Binary codes capable of correcting insertions, deletions and reversals," *Dokl. Akad. Nauk SSSR*, vol. 163, no. 4, pp. 845–848, 1965. Translation: *Sov. Phys. Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[12] B.H. Marcus, R.M. Roth, and P.H. Siegel, *An introduction to coding for constrained system*, 5th edition, Oct. 2001.

[13] C. J. Mitchell, T. Etzion, and K. G. Paterson, "A method for constructing decodable de Bruijn sequences," *IEEE Trans. Inform. Theory*, vol. 42, no. 5, pp. 1472–1478, 1996.

[14] S. S. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, 2008.

[15] Z. Sun, W. Wu, and H. Li, "Cross-layer racetrack memory design for ultra high density and low power consumption," *Design Automation Conference (DAC)*, pp.1–6, 2013.

[16] A. Vahid, G. Mappouras, D. J. Sorin, and R. Calderbank, "Correcting two deletions and insertions in racetrack memory," 2017, arXiv:1701.06478.

[17] A.J.V. Wijngaarden and K.A.S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE J. Select Areas Comm.,* vol. 28, no. 2, pp. 200–207, Feb. 2010.

[18] E. Yaakobi, J. Bruck, and P. H. Siegel, "Constructions and decoding of cyclic codes over b-symbol read channels," *IEEE Trans. Inform. Theory*, vol. 62, no. 4, pp. 1541–1551, 2016.

[19] C. Zhang, G. Sun, X. Zhang, W. Zhang, W. Zhao, T. Wang, Y. Liang, Y. Liu, Y. Wang, and J. Shu, "Hi-fi playback: Tolerating position errors in shift operations of racetrack memory," *2015 ACM/IEEE 42nd Annual Inter. Symp. on Computer Architecture (ISCA)*, pp. 694–706, Jul. 2015.