Balanced de Bruijn Sequences

Sagi Marcovich
Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa 3200003, Israel
Email: sagimar@cs.technion.ac.il

Tuvi Etzion
Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa 3200003, Israel
Email: etzion@cs.technion.ac.il

Eitan Yaakobi
Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa 3200003, Israel
Email: yaakobi@cs.technion.ac.il

***Abstract*—The de Bruijn graph and its sequences have found many diverse applications in information theory as well as other areas of computer science and engineering such as interconnection networks, VLSI decomposition, and most recently in DNA storage. Binary balanced sequences have also been a subject to a large research during the last forty years with various applications and a lot of interest in information theory. There have been some works on classification of balanced sequences mainly based on their spectral-null order. This work generalizes the concept of de Bruijn sequences, based on the de Bruijn graph of order ℓ, where each edge is multiplied to a fixed number of multiple edges. This implies that in the sequences derived from the generalized graph each ℓ-tuple has the same multiplicity. Using this generalization we form an interesting hierarchy between balanced sequences. Furthermore, another hierarchy is given by the derivatives of balanced sequences. Enumeration for each such family of sequences and efficient encoding and decoding algorithms are also provided.**

I. Introduction

The binary de Bruijn graph of order ℓ, G_ℓ, was introduced in 1946 by de Bruijn [4]. His target in introducing this graph was to find a recursive method to enumerate the number of cyclic binary sequences of length 2^ℓ such that each ℓ-tuple appears as a cyclic window exactly once in each sequence. These sequences were later called *de Bruijn sequences*. It should be mentioned that in parallel also Good [14] defined the same graph. These results were later generalized in [34] for any alphabet of finite size m.

The vertices of G_ℓ are the binary words of length $\ell-1$, and the edges correspond to the words of length ℓ. There is an edge $u \to v$ if v can be obtained from u by shifting one entry left and appending a symbol. Eulerian cycles in de Bruijn graphs, i.e. cycles that visit all the edges of G_ℓ exactly once, are called *de Bruijn cycles*. Alternatively, these can be viewed as Hamiltonian cycles in $G_{\ell+1}$ as well, i.e., cycles that visit each one of the vertices of $G_{\ell+1}$ exactly once. It was proven in [4] that the number of de Bruijn cycles in G_ℓ is $2^{2^{\ell-1}-\ell}$. Since each de Bruijn cycle induces 2^ℓ distinct acyclic de Bruijn sequences, the number of such acyclic de Bruijn sequences is $2^{2^{\ell-1}}$ and their asymptotic rate is $1/2$. One of the first applications of the de Bruijn graph was in the introduction of shift-register sequences and linear feedback shift registers [13]. Throughout the years, an extensive amount of works have studied the de Bruijn graph and its sequences. Several of those include [5], [6], [8], [10], [11], [15], [19], [20], [21], [25], [30]. Most recently, DNA storage has brought fresh interest to this family of sequences, see [3], [17], [29].

A de Bruijn sequence of length 2^ℓ can be viewed as a sequence with balanced occurrences of ℓ-tuples. This immediately implies a connection to another known family of sequences, which is the set of *balanced sequences*. A sequence is said to be balanced if it contains an identical number of *zeros* and *ones*. In [18], Knuth proved that the redundancy of this family of sequences is $0.5\log n + \mathcal{O}(1)$ bits and designed an efficient algorithm with linear time complexity which encodes an arbitrary binary sequence to a balanced one using $\log n + \Theta(\log\log n)$ redundancy bits. In the following years, various generalizations of the family of balanced sequences and Knuth's algorithm were studied; see e.g. [1], [2], [16], [26], [27], [28], [31], [32], [33], [35].

This paper studies a novel family of codes that generalizes both de Bruijn sequences and balanced sequences. For integers ℓ, μ, we say that an acyclic sequence is an (ℓ,μ)-*balanced de Bruijn* sequence, or (ℓ,μ)-*BdB* sequence in short, if it contains each binary ℓ-tuple as a window exactly μ times. Note that when $\mu = 1$, the set of (ℓ,μ)-BdB sequences, denoted by $\mathcal{B}(\ell,\mu)$, is the set of acyclic de Bruijn sequences, and when $\ell = 1$ it is the set of balanced sequences.

The first part of this paper studies the problem for the value $\ell = 2$ and presents an encoding algorithm for the set $\mathcal{B}(2, n/4)$ that is based on Knuth's algorithm and uses $2\log n + \Theta(\log\log n)$ redundancy bits. Next, the main result of this paper, an exact enumeration of $\mathcal{B}(\ell,\mu)$ for all admissible values of ℓ, μ, is presented. This enumeration uses a construction which is based on a variation of the de Bruijn graph, where each edge is multiplied μ times. The result of this enumeration is

$$|\mathcal{B}(\ell,\mu)| = 2^{2^{\ell-1}} \binom{2\mu-1}{\mu-1}^{2^{\ell-1}}.$$

Specifically, we prove that when $\ell = c$ for some constant c, the redundancy of $\mathcal{B}(\ell,\mu)$ equals $2^{c-2}\log n + \mathcal{O}(1)$, and when μ is a constant, the asymptotic rate of this set is

$$\mathbb{R}(\mu) = \limsup_{\ell\to\infty} \frac{\log|\mathcal{B}(\ell,\mu)|}{\mu 2^\ell} = \frac{1+\log\binom{2\mu-1}{\mu-1}}{2\mu}.$$

Furthermore, we present a generic encoding scheme for (ℓ,μ)-BdB sequences with $\ell = \mathcal{O}(\log\log n)$ that uses $2^{\ell-1}(\log\mu+1) + \ell + \Theta(\log\log\mu)$ redundancy bits.

The paper ends with another generalization of the family of balanced sequences. For a sequence $\boldsymbol{s} = (s_0, s_1, \ldots, s_{n-1}) \in \mathbb{F}_2^n$, let $\boldsymbol{s}' = (s_0+s_1, s_1+s_2 \ldots, s_{n-1}+s_0) \in \mathbb{F}_2^n$ denote its *derivative sequence*. These sequences and their applications were studied in several works; see e.g. [5], [9], [12], [20], [24]. A sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is called a *k-order balanced sequence* if the sequences $\boldsymbol{s}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k-1)}$ are all balanced, where $\boldsymbol{s}^{(i)}$ is the derivative of $\boldsymbol{s}^{(i-1)}$ and $\boldsymbol{s}^{(0)} = \boldsymbol{s}$. We study the cardinality of this novel family of sequences for several fixed values of k, and present an efficient encoder that is based on Knuth's algorithm using $k\log n + \Theta(\log\log n)$ redundancy bits.

The rest of this paper is organized as follows. In Section II, we formally define the families of sequences studied in this paper, and review several previous results. In Section III, we study the family of (ℓ,μ)-BdB sequences, for various values of μ and ℓ, and in Section IV the family of k-order balanced derivatives is studied. Due to the lack of space, some of the proofs in this paper are omitted.

978-1-5386-8209-8/21/$31.00 ©2021 IEEE

II. Definitions and Preliminaries

For two integers $i, k \in \mathbb{N}$ such that $i \leqslant k$ we denote by $[i, k]$ the set $\{i, \ldots, k\}$ and use $[k]$ as a shorthand for $[0, k-1]$. Let $n \in \mathbb{N}$ be an integer and let $\boldsymbol{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{F}_2^n$ denote a sequence. For two positive integers $i, k \leqslant n$, let $\boldsymbol{s}_{i,k}$ denote the *cyclic window* $(s_i, \ldots, s_{i+k-1})$, when the indices are taken modulo n. Additionally, let $\mathrm{Pref}_k(\boldsymbol{s}) = \boldsymbol{s}_{0,k}$, $\mathrm{Suff}_k(\boldsymbol{s}) = \boldsymbol{s}_{n-k,k}$ denote the *k-prefix*, *k-suffix* of $\boldsymbol{s}$, respectively. Let $w_H(\boldsymbol{s})$ denote the *Hamming weight* of $\boldsymbol{s}$. The notation $\boldsymbol{s}^i$ denotes the concatenation of $\boldsymbol{s}$ i times and $\boldsymbol{s} \circ \boldsymbol{v}$ is the concatenation of $\boldsymbol{s}$ and another sequence $\boldsymbol{v}$. For an index $i \in [n]$, $b(i)$ denotes the binary representation of i using $\lceil \log_2(n) \rceil$ bits. Finally, the redundancy of a set $A \subseteq \mathbb{F}_2^n$ is defined as $\mathrm{red}(A) = n - \log_2 |A|$. For the rest of this paper, we assume all logarithms are taken to base 2.

Definition 1. *The ℓ-th order binary* ***de Bruijn graph*** *G_ℓ is the digraph (V, E), where $V = \mathbb{F}_2^{\ell-1}$ and*

$$E = \{((s_0, s_1, \ldots, s_{\ell-2}), (s_1, s_2, \ldots, s_{\ell-1})) \mid s_i \in \mathbb{F}_2\}.$$

Note that the edges of G_ℓ correspond to the set of binary ℓ-tuples, $\mathbb{F}_2^\ell$.

Definition 2. *Let ℓ be an integer and $n = 2^\ell$. A sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is called an* ***ℓ-de Bruijn sequence*** *if $\boldsymbol{s}$ contains each binary ℓ-tuple as a cyclic window exactly once.*

The connection between Eulerian cycles in G_ℓ to ℓ-de Bruijn sequences is as follows. In order to generate a sequence from a cycle, we pick any edge in the cycle to start from and consider the first entry of each consecutive edge in the cycle.

Example 1. Let $\ell = 3, n = 8$. Then,

$$\boldsymbol{s} = 00010111$$

is a 3-de Bruijn sequence. $\boldsymbol{s}$ can be generated from G_3 using the Eulerian cycle

$$00 \overset{000}{\to} 00 \overset{001}{\to} 01 \overset{010}{\to} 10 \overset{101}{\to} 01 \overset{011}{\to} 11 \overset{111}{\to} 11 \overset{110}{\to} 10 \overset{100}{\to} 00.$$

Definition 3. *Let $n \in \mathbb{N}$ be an integer and let $\boldsymbol{s} \in \mathbb{F}_2^n$ be a sequence. We say that $\boldsymbol{s}$ is* ***balanced*** *if $w_H(\boldsymbol{s}) = \lceil n/2 \rceil$.*

Note that Definition 3 is applicable for both even and odd sequence lengths. When n is even, it is required that the numbers of *zeros* and *ones* are identical. When n is odd, the bit *one* appears one more time than the bit *zero*.

We define next a family of sequences which constitute a generalization of de-Bruijn sequences, as well as a generalization of the family of balanced sequences.

Definition 4. *Let ℓ, μ be integers, and $n = \mu 2^\ell$. A sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is called an (ℓ, μ)-* ***balanced de Bruijn*** *sequence, or an (ℓ, μ)-* ***BdB*** *sequence in short, if $\boldsymbol{s}$ contains each binary ℓ-tuple as a cyclic window exactly μ times, i.e., for every $\boldsymbol{v} \in \mathbb{F}_2^\ell$, there exist μ different indices $i_1, \ldots, i_\mu \in [n]$ such that $\boldsymbol{s}_{i_j,\ell} = \boldsymbol{v}$ for every $j \in [\mu]$.*

Example 2. Let $\ell = 3, \mu = 2, n = 16$. Then, the sequence

$$\boldsymbol{s} = 0000100101101111$$

is a $(3, 2)$-BdB sequence, since it contains each binary 3-tuple as a window exactly twice.

We denote the set of all (ℓ, μ)-BdB sequences over $\mathbb{F}_2$ by $\mathcal{B}(\ell, \mu)$. We also define the asymptotic rate of $\mathcal{B}(\ell, \mu)$ as a function of the multiplicity μ by

$$\mathbb{R}(\mu) = \limsup_{\ell \to \infty} \frac{\log |\mathcal{B}(\ell, \mu)|}{\mu 2^\ell}.$$

Observe that when $\mu = 1$ the set $\mathcal{B}(\ell, 1)$ is exactly the set of de Bruijn sequences, and hence $\mathbb{R}(1) = 1/2$. On the other hand, for $\ell = 1$, we receive the set $\mathcal{B}(1, n/2)$ for $n = 2\mu$, i.e., the set of length-n balanced sequences, and thus

$$\mathrm{red}(\mathcal{B}(1, n/2)) = 0.5 \log n + \mathcal{O}(1).$$

Lastly, we present the useful definition of a derivative sequence.

Definition 5. *Let $\boldsymbol{s} \in \mathbb{F}_2^n$ be a sequence. Its* ***derivative sequence****, denoted by $\boldsymbol{s}'$, is the sequence*

$$\boldsymbol{s}' = (s_0 + s_1, s_1 + s_2, \ldots, s_{n-2} + s_{n-1}, s_{n-1} + s_0) \in \mathbb{F}_2^n.$$

It follows immediately from Definition 5 that $\boldsymbol{s}$ can be uniquely constructed from $\boldsymbol{s}'$ and s_0 by repeatedly invoking $s_i = s'_{i-1} + s_{i-1}$ for $i = 1, \ldots, n-1$.

III. Balanced de Bruijn Sequences

In this section, we present enumeration methods and encoding schemes of (ℓ, μ)-BdB sequences for various values of ℓ, μ.

A. Encoding Algorithm for $\ell = 2$

First, we present an encoding algorithm for $\mathcal{B}(2, \mu)$, which is based on the connection of this set to the family of balanced sequences. Let $n = 4\mu$ denote the length of the sequence.

We seek to extend Knuth's algorithm [18], which encodes an arbitrary binary sequence to a balanced one with $\log n + \Theta(\log \log n)$ redundancy bits, for the case where each 2-tuple appears exactly $n/4$ times. We derive the following property.

Lemma 6. *Let $\boldsymbol{s} \in \mathbb{F}_2^n$ be a sequence. Then, $\boldsymbol{s}$ is a $(2, n/4)$-BdB sequence if and only if $\boldsymbol{s}$ is balanced and $\boldsymbol{s}'$ is balanced.*

Remember that Knuth's algorithm [18] balances a sequence $\boldsymbol{x} \in \mathbb{F}_2^n$ by picking a *balancing index*, i, and flipping $\mathrm{Pref}_i(\boldsymbol{x})$ such that $\boldsymbol{x}$ is balanced. Then, the index i is encoded as a balanced sequence using $\log n + \Theta(\log \log n)$ bits. Let $\mathcal{E}_K$ denote this encoder. Additionally, for a sequence $\boldsymbol{x} \in \mathbb{F}_2^n$ let $\delta(\boldsymbol{x})$ denote the *imbalance* of $\boldsymbol{x}$, i.e., $\delta(\boldsymbol{x}) = \lceil n/2 \rceil - w_H(\boldsymbol{x})$. Notice that by flipping any prefix of $\boldsymbol{x}$, the derivative $\boldsymbol{x}'$ is changed by at most two bits, and thus the imbalance of $\boldsymbol{x}'$ is changed by at most 2.

Algorithm 1 receives $\boldsymbol{s}$, an input sequence over $\mathbb{F}_2^{n-\rho}$ where $\rho = 2 \log n + \Theta(\log \log n)$, and outputs a sequence in $\mathcal{B}(2, n/4)$. The algorithm first uses Knuth's algorithm to balance $\boldsymbol{s}'$ and $\boldsymbol{s}$. The balancing indices are concatenated to construct the indices sequence $\boldsymbol{v}$ of length[1] $2 \log n$. Then, the process is repeated to create $\boldsymbol{u}$, the indices sequence of $\boldsymbol{v}$, of length $2 \log(2 \log n)$. Since $\boldsymbol{u}$ has length of $\Theta(\log \log n)$, it can be easily transformed to a balanced sequence with a balanced derivative using a fixed encoding map. Later, $\boldsymbol{s}$ and the balanced indices sequences are appended to construct $\boldsymbol{x}$, a balanced sequence with a nearly balanced derivative. Finally, $\boldsymbol{x}$ and its derivative are balanced manually by appending sequences of constant lengths m_0 and m_1 respectively. The

[1] For simplicity, in this section we drop some of the ceiling notations.

generated result is a sequence that belongs to $\mathcal{B}(2, n/4)$ according to Lemma 6.

Throughout Algorithm 1, for simplicity, we sometimes operate on a derivative of a sequence rather than on the actual sequence $\boldsymbol{x}$. This implicitly means that we invoke a derivative to obtain $\boldsymbol{x}'$, then operate on $\boldsymbol{x}'$ to receive $\widehat{\boldsymbol{x}}'$, and the result is the sequence $\widehat{\boldsymbol{x}}$ that is reconstructed from $\widehat{\boldsymbol{x}}'$ and x_0. Lastly, denote the words $\boldsymbol{w}_1 = 0001, \boldsymbol{w}_0 = 0011, \boldsymbol{w}_{-1} = 0111$ and notice that $\delta(\boldsymbol{w}_1) = 1, \delta(\boldsymbol{w}_0) = 0, \delta(\boldsymbol{w}_{-1}) = -1$.

Algorithm 1 Encoding $(2, n/4)$-BdB sequences

Input: A sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$
Output: A sequence $\boldsymbol{x} \in \mathcal{B}(2, n/4)$

1: Use $\mathcal{E}_K$ to find i_1, the balancing index of $\boldsymbol{s}'$. Flip the i_1-prefix of $\boldsymbol{s}'$ which transforms $\boldsymbol{s}$ to $\boldsymbol{s}^*$.
2: Use $\mathcal{E}_K$ to find i_0, the balancing index of $\boldsymbol{s}^*$, and flip its i_0-prefix to receive $\widehat{\boldsymbol{s}}$.
3: Construct the indices sequence $\boldsymbol{v} = b(i_0) \circ b(i_1)$ of length $2\log n$.
4: Repeat Steps 1-3 with $\boldsymbol{v}$ as an input to receive $\widehat{\boldsymbol{v}}$ and the indices sequence $\boldsymbol{u}$ of length $2\log(2\log n)$.
5: Use an encoding map to encode $\boldsymbol{u}$ to $\widehat{\boldsymbol{u}}$, a balanced sequence with a balanced derivative of length $\Theta(\log\log n)$.
6: Let $\boldsymbol{x} = 0 \circ \widehat{\boldsymbol{s}} \circ \widehat{\boldsymbol{v}} \circ \widehat{\boldsymbol{u}}$.
7: Denote $g = \delta(\boldsymbol{x}')$. Append 1^{2g} to the end of $\boldsymbol{x}'$ if $g \geqslant 0$, and $0^{2|g|}$ otherwise, then append $(01)^{\frac{m_1}{2}-2|g|}$ to receive $\widehat{\boldsymbol{x}}$.
8: Denote $h = \delta(\widehat{\boldsymbol{x}})$. Append $\boldsymbol{w}_1^h$ to the end of $\hat{\boldsymbol{x}}$ if $h \geqslant 0$, and $\boldsymbol{w}_{-1}^{|h|}$ otherwise, then append $\boldsymbol{w}_0^{\frac{m_0}{4}-4|h|}$.
9: Return $\widehat{\boldsymbol{x}}$.

It is presented later in Corollary 15 that

$$\mathrm{red}(\mathcal{B}(2, n/4)) = 2\log n + \mathcal{O}(1).$$

Hence, the size of the encoding map used at Step 5 is $2\log\log n + \Theta(\log\log\log n)$. Due to its small size, this encoding map can be generated efficiently and does not affect the complexity of the algorithm.

Note that the lengths m_1, m_0 are a function of the imbalance of $\boldsymbol{x}'$ and $\widehat{\boldsymbol{x}}$ respectively, which can be upper bounded and set in advance independently of the input sequence.

Theorem 7. *Algorithm 1 returns a sequence $\boldsymbol{x} \in \mathcal{B}(2, n/4)$ that can be uniquely decoded to its input sequence $\boldsymbol{s}$. The algorithm uses $\rho = 2\log n + \Theta(\log\log n)$ redundancy bits and its time complexity is $\mathcal{O}(n)$.*

B. Enumeration Using the de Bruijn Graph

Next, we use a generalization of the de Bruijn graph, G_ℓ, which was presented in Definition 1, in order to enumerate and encode (ℓ, μ)-BdB sequences.

Let μ, ℓ be positive integers and denote $n = \mu 2^\ell$. We construct a digraph $G_{\ell,\mu} = (V, E)$ which is based on the de Bruijn graph G_ℓ. Similarly to the vertices of G_ℓ, the set V contains $2^{\ell-1}$ vertices represented by the binary $(\ell-1)$-tuples. The set E has $\mu 2^\ell$ edges represented by the binary ℓ-tuples. The edges are the same as in G_ℓ, but between any two vertices which have an edge in G_ℓ, E has μ parallel edges.

Let $\boldsymbol{r} = 0^{\ell-1}$ denote the all-zero vertex of the graph $G_{\ell,\mu}$ for the rest of this paper. A *reverse spanning tree* T of $G_{\ell,\mu}$ is a graph whose underline graph is a tree rooted at $\boldsymbol{r}$, it contains all the vertices of $G_{\ell,\mu}$, and there is a unique directed path from each vertex v of T to $\boldsymbol{r}$. Note that the set of reverse spanning trees is isomorphic to the ordinary spanning trees of $G_{\ell,\mu}$, where their edges are reversed. It is known from [22], [23] that the number of reverse spanning trees of G_ℓ is $2^{2^{\ell-1}-\ell}$, and thus it is also the number of reverse spanning trees of $G_{\ell,\mu}$.

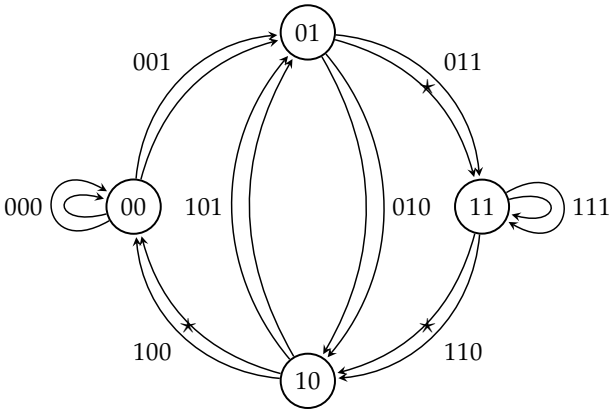
Algorithm 2 is applied on $G_{\ell,\mu}$ and a reverse spanning tree T of $G_{\ell,\mu}$. It is a nondeterministic algorithm that generates numerous different paths that visit each one of the edges of $G_{\ell,\mu}$ exactly once. First, all the edges are set to be *unmarked* at the beginning of the algorithm. When a vertex v of $G_{\ell,\mu}$ is visited, the edge on which v is left changes its status from unmarked to *marked*. In order to guarantee the uniqueness of the path, the edges of T are notated as *starred*, and the algorithm leaves a vertex on a starred edge only if it is the last unmarked edge left. Finally, for each generated path, the algorithm derives its associated BdB sequence.

Algorithm 2 Generating sequence in $\mathcal{B}(\ell, \mu)$ from T

Input: A reverse spanning tree T of $G_{\ell,\mu}$
Output: A sequence $\boldsymbol{s} \in \mathcal{B}(\ell, \mu)$

1: Place a star on all the edges of T. All the edges of $G_{\ell,\mu}$ are set to be unmarked.
2: Set the current vertex $v := \boldsymbol{r}$.
3: **while** the starred edge outgoing of v is unmarked **do**
4: **if** v has an unstarred unmarked outgoing edge **then**
5: Pick at random an unstarred edge $e = v \to u$.
6: Leave v on the edge e to its endpoint u. Set $v := u$ and the edge e as marked.
7: **else**
8: Leave v on the starred edge, to its endpoint u. Set $v := u$ and the edge $v \to u$ as marked.
9: **end if**
10: **end while**
11: Construct the output sequence $\boldsymbol{s}$ by considering the first bit of each edge of the path.

Example 3. Let $\ell = 3, \mu = 2$. The graph $G_{\ell,\mu}$ is



where the starred edges correspond to the tree

$$T = 01 \overset{011}{\to} 11 \overset{110}{\to} 10 \overset{100}{\to} 00.$$

One of the paths generated by Algorithm 2 when invoked with T is

$$00 \overset{000}{\to} 00 \overset{000}{\to} 00 \overset{001}{\to} 01 \overset{010}{\to} 10 \overset{100}{\to} 00 \overset{001}{\to} 01 \overset{010}{\to} 10 \overset{101}{\to} 01$$
$$\overset{011}{\to} 11 \overset{110}{\to} 10 \overset{101}{\to} 01 \overset{011^\star}{\to} 11 \overset{111}{\to} 11 \overset{111}{\to} 11 \overset{110^\star}{\to} 10 \overset{100^\star}{\to} 00,$$

which corresponds to the sequence $\boldsymbol{s} = 0000100101101111$.

The next few lemmas and corollaries describe the connection between paths generated by Algorithm 2 and sequences of $\mathcal{B}(\ell,\mu)$. Lemma 8 and Lemma 9 are proved similarly as related results in [22], [23].

Lemma 8. *The path generated by Algorithm 2 ends at the root $\boldsymbol{r}$.*

Lemma 9. *When Algorithm 2 terminates all the edges of the graph were traversed, i.e., all the edges of $G_{\ell,\mu}$ are marked.*

It follows from Lemma 9 that each reverse spanning tree T corresponds to some sequences of length $\mu 2^\ell$ in which each ℓ-tuple appears exactly μ times, i.e., an (ℓ,μ)-BdB sequence. Since the chosen root is $\boldsymbol{r}=0^{\ell-1}$, it follows that every such a sequence starts with $\ell-1$ consecutive *zeros*.

Lemma 10. *Each acyclic sequence of $\mathcal{B}(\ell,\mu)$ which starts with $\ell-1$ consecutive* zeros *can be obtained from exactly one reverse spanning tree T.*

Proof: Given a vertex $v\neq\boldsymbol{r}$, its last appearance on the sequence $\boldsymbol{s}$, as an $(\ell-1)$-tuple, is related to an edge $v\to u$. This edge is the unique starred edge going out of v in $G_{\ell,\mu}$, by the definition of the algorithm. Thus, the starred edges and the reverse spanning tree T are uniquely defined by the sequence $\boldsymbol{s}$. ∎

Corollary 11. *Each acyclic sequence of $\mathcal{B}(\ell,\mu)$ which starts with $\ell-1$ consecutive* zeros *is constructed exactly once by Algorithm 2.*

Lemma 12. *From each reverse spanning tree T, there are $2\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$ distinct acyclic sequences which are constructed by Algorithm 2.*

Combining Corollary 11, Lemma 12, and the fact that the number of reversed spanning trees rooted at $\boldsymbol{r}$ in $G_{\ell,\mu}$ is $2^{2^{\ell-1}-\ell}$, the following result is immediately inferred.

Corollary 13. *The total number of distinct acyclic sequences of $\mathcal{B}(\ell,\mu)$ formed by Algorithm 2 is $2^{2^{\ell-1}-\ell+1}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$.*

From Corollary 11, the enumerated value of Corollary 13 accounts only for sequences of $\mathcal{B}(\ell,\mu)$ that start with $\ell-1$ *zeros*. The next theorem proves that multiplying the value of Corollary 13 by $|V|=2^{\ell-1}$ gives the exact size of $\mathcal{B}(\ell,\mu)$.

Theorem 14. *For every integers ℓ,μ, the size of $\mathcal{B}(\ell,\mu)$ is*

$$|\mathcal{B}(\ell,\mu)|=2^{2^{\ell-1}}\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$$

Proof: Let $\boldsymbol{x}\in\mathcal{B}(\ell,\mu)$ be a sequence that is generated by Algorithm 2 and starts with $0^{\ell-1}$. Let $p(\boldsymbol{x})$ denote the *period* of $\boldsymbol{x}$, that is, the smallest positive integer that satisfies $x_i=x_{i+p(\boldsymbol{x})}$ for every $i\in[n-p(\boldsymbol{x})+1]$. We distinguish between two cases based on the period $p(\boldsymbol{x})$.

First assume that $\boldsymbol{x}$ is aperiodic. We denote for every $v\in\mathbb{F}_2^{\ell-1}$ the sequences $\boldsymbol{x}_{v,1},\ldots,\boldsymbol{x}_{v,2\mu}$, which are the cyclic shifts of $\boldsymbol{x}$ such that for every $i\in[1,2\mu]$, $\boldsymbol{x}_{v,i}$ starts with the i-th occurrence of v in $\boldsymbol{x}$. Since $\boldsymbol{x}$ is aperiodic, it follows that all these sequences are distinct. We define for the root $\boldsymbol{r}$ and for every $i\in[1,2\mu]$ the set $B(\boldsymbol{x}_{\boldsymbol{r},i})=\{\boldsymbol{x}_{v,i}\mid v\in\mathbb{F}_2^{\ell-1}\}$. Note that since from Corollary 11 all the sequences $\boldsymbol{x}_{\boldsymbol{r},1},\ldots,\boldsymbol{x}_{\boldsymbol{r},2\mu}$ are generated by Algorithm 2, we pick one of them w.l.o.g to denote the baseline sequence $\boldsymbol{x}$. It follows immidiately from the definition that for every $\boldsymbol{x}\neq\boldsymbol{s}$ that are generated by the algorithm, $B(\boldsymbol{x})\cap B(\boldsymbol{s})=\varnothing$ and hence applying $B(\boldsymbol{x})$ on each aperiodic output of the algorithm generates $2^{\ell-1}$ distinct sequences in $\mathcal{B}(\ell,\mu)$.

On the other hand, if $\boldsymbol{x}$ is periodic, let $\boldsymbol{y}=\mathrm{Pref}_{p(\boldsymbol{x})}(\boldsymbol{x})$. Clearly, $\boldsymbol{y}\in\mathcal{B}(\ell,\mu')$ where $\mu'=\mu\frac{|\boldsymbol{y}|}{|\boldsymbol{x}|}$. This time, only the sequences $\boldsymbol{x}_{\boldsymbol{r},1},\ldots,\boldsymbol{x}_{\boldsymbol{r},2\mu'}$ are distinct and are generated by the algorithm once. We define for each $\boldsymbol{x}_{\boldsymbol{r},i}$ the set $B(\boldsymbol{x}_{\boldsymbol{r},i})=\{(\boldsymbol{y}_{\boldsymbol{r},i})^{\frac{\mu}{\mu'}}\mid v\in\mathbb{F}_2^{\ell-1}\}$. From similar arguments as the aperiodic case, this construction yields $2^{\ell-1}$ distinct sequences in $\mathcal{B}(\ell,\mu)$ for each periodic output of Algorithm 2. ∎

Next, we use the result of Theorem 14 in order to derive the redundancy and the asymptotic rate of $\mathcal{B}(\ell,\mu)$ for different values of ℓ,μ.

Corollary 15. *The redundancy of $\mathcal{B}(\ell,\mu)$ satisfies*

$$red(\mathcal{B}(\ell,\mu))=2^{\ell-1}\left(0.5\log\mu+0.5\log\pi-\log(1-\frac{1}{c\mu})\right),$$

where $8\leqslant c\leqslant 9$. In particular, for a constant $\ell\in\mathbb{N}$ and $n=\mu 2^\ell$ for some integer μ, the redundancy of $\mathcal{B}(\ell,n/2^\ell)$ is

$$red(\mathcal{B}(\ell,n/2^\ell))=2^{\ell-2}\log n+\mathcal{O}(1).$$

Another interesting case is when μ is a constant. Hence, we have the following corollary.

Corollary 16. *Let $\mu\in\mathbb{N}$ be a constant integer. The asymptotic rate of $\mathcal{B}(\ell,\mu)$ is*

$$\mathbb{R}(\mu)=\frac{1+\log\binom{2\mu-1}{\mu-1}}{2\mu}.$$

Table I summarizes the asymptotic rate results of $\mathcal{B}(\ell,\mu)$ for some chosen values of μ, compared with the known rates of de Bruijn sequences.

TABLE I
RATES OF EXACT COVERING WITH MULTIPLICITY

Multiplicity	Rate
$\mu=1$ (de Bruijn)	0.5
$\mu=2$	0.646
$\mu=3$	0.720
$\mu=4$	0.766
$\mu=8$	0.853
$\mu=16$	0.911

Remark 1. Note that it is possible to alter the definition of (ℓ,μ)-BdB sequences such that the sequence length is $n=\mu 2^\ell+\ell-1$ and only acyclic windows are considered. However, since the in-degree and the out-degree of every vertex in $G_{\ell,\mu}$ are identical, each such a sequence induces a cycle in the graph and therefore its $(\ell-1)$-suffix equals its $(\ell-1)$-prefix. Hence, the two representations are isomorphic.

Next, we present an encoding algorithm that utilizes Algorithm 2 in order to generate (ℓ,μ)-BdB sequences of length $n=\mu 2^\ell$, for $\ell=\mathcal{O}(\log\log n)$. From Lemma 12, given a reverse spanning tree T of G_ℓ, Algorithm 2 forms $2\binom{2\mu-1}{\mu-1}^{2^{\ell-1}}$ distinct sequences of $\mathcal{B}(\ell,\mu)$. This value corresponds to $\binom{2\mu}{\mu}$ possible orders to travel the outgoing edges of the root $\boldsymbol{r}$, and $\binom{2\mu-1}{\mu-1}$ possible orders to travel the outgoing edges of each $v\neq\boldsymbol{r}$. This immediately yields an efficient enumerative encoding algorithm [7] for these sequences for a given reverse

spanning tree T. The series of choices to travel the outgoing edges of $\boldsymbol{r}$ can be represented by a balanced word $\boldsymbol{z}$ of length 2μ. Assume that the outgoing edges of $\boldsymbol{r}$ are e_0, e_1, then at the i-th visit of $\boldsymbol{r}$, the algorithm picks to travel e_0 if $z_i = 0$, and e_1 otherwise. Similarly, the series of choices to travel the outgoing edges of any $v \neq \boldsymbol{r}$, with outgoing edges e_0 (for edges parallel to the starred edge) and e_1, can be represented by a word $\boldsymbol{z}$ of length $2\mu - 1$ and weight μ. In this case, the starred edge is traveled at the last visit of v, and otherwise the chosen edge is picked depending if $z_i = 0$ for e_0 or $z_i = 1$ for e_1. Each of these balanced words can be encoded using Knuth's algorithm [18] with $\log \mu + \Theta(\log\log \mu)$ redundancy bits.

Note that this algorithm needs to generate reverse spanning trees of G_ℓ. Unfortunately, there is no enumerative encoding algorithm for these spanning trees for large ℓ. Therefore, in order to have linear time complexity, the algorithm is applicable only for small values of ℓ that satisfy $\ell = \mathcal{O}(\log\log n)$. Then, the redundancy of the algorithm is

$$2^{\ell-1}(\log \mu + 1) + \ell + \Theta(\log\log \mu).$$

IV. Balanced Derivatives

In this section, we study the family of sequences with balanced derivatives, defined next. These sequences are highly related to balanced de Bruijn sequences as proven later in Lemma 18.

Definition 17. *Let k be an integer. A sequence $\boldsymbol{s} \in \mathbb{F}_2^n$ is called **k-order balanced** if $\boldsymbol{s}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k-1)}$ are all balanced, where $\boldsymbol{s}^{(i)}$ is the derivative of $\boldsymbol{s}^{(i-1)}$ for $1 \leqslant i \leqslant k-1$ and $\boldsymbol{s}^{(0)} = \boldsymbol{s}$.*

Let $\mathcal{D}(n,k)$ denote the family of k-order balanced sequences over $\mathbb{F}_2^n$. The following lemma presents a connection between k-order balanced sequences and (ℓ,μ)-BdB sequences.

Lemma 18. *For all admissible integers ℓ, μ, every (ℓ,μ)-BdB sequence is also ℓ-order balanced.*

Lemma 18 gives a lower bound on the cardinality of $\mathcal{D}(n,k)$, using the results presented in Section III-B. From Lemma 6, in the case of $k = 2$ the opposite direction of Lemma 18 holds as well and thus the bound is tight.

Corollary 19. *The redundancy of $\mathcal{D}(n,2)$ satisfies*

$$red(\mathcal{D}(n,2)) = \log n + \mathcal{O}(1).$$

However, for $k > 2$ the opposite direction of Lemma 18 no longer holds and a tighter bound remains an open question. We present next a tight bound for the case of $k = 3$ that is obtained using exact enumeration of 3-order balanced sequences.

Theorem 20. *The redundancy of $\mathcal{D}(n,3)$ satisfies*

$$red(\mathcal{D}(n,3)) = 1.5\log n + \mathcal{O}(1).$$

Next, we have Theorem 21 which gives an upper bound on the redundancy of $\mathcal{D}(n,k)$ for any fixed $k \in \mathbb{N}$. This bound significantly improves the upper bound of $2^{k-2}\log n + \mathcal{O}(1)$ that follows from Lemma 18 and Corollary 15.

Theorem 21. *Let $k \in \mathbb{N}$ be a constant integer. The redundancy of $\mathcal{D}(n,k)$ satisfies*

$$red(\mathcal{D}(n,k)) \leqslant k\log n + \Theta(\log\log n).$$

The proof of Theorem 21 is given by an explicit encoding algorithm, presented next. This algorithm generalizes Algorithm 1 and uses Knuth's encoder, $\mathcal{E}_K$, in order to efficiently encode k-order balanced sequences for any fixed balanced degree k while its redundancy is

$$\rho = k\log n + \Theta(\log\log n).$$

Algorithm 3 receives an input sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$, and uses $\mathcal{E}_K$ to iteratively balance its derivatives, from $\boldsymbol{s}^{(k-1)}$ to $\boldsymbol{s}$ and constructs the balancing indices sequence $\boldsymbol{v}$. Then, the process is repeated with $\boldsymbol{v}$ as an input to create $\boldsymbol{u}$, a second balancing indices sequence. Next, $\boldsymbol{u}$ is transformed to a k-order balanced sequence using a fixed encoding map. Then, $\boldsymbol{s}$ and the indices sequences are concatenated to construct $\boldsymbol{x}$, a sequence that is almost k-order balanced. At the last step, the algorithm manually balances $\boldsymbol{x}^{(k-1)}, \ldots, \boldsymbol{x}$ while ensuring that when balancing a derivative, its higher derivatives remain balanced.

Observe that flipping a prefix of a sequence $\boldsymbol{x}$ changes at most two bits in $\boldsymbol{x}'$. Moreover, changing a bit of $\boldsymbol{x}$ affects at most two bits of $\boldsymbol{x}'$. We can infer that flipping a prefix to balance $\boldsymbol{x}$ imbalances the first k derivatives of $\boldsymbol{x}$ by a constant number of bits. For convenience, similarly to Algorithm 1 we can operate on a derivative of a sequence instead of the actual sequence $\boldsymbol{x}$. The same principle applies to any higher derivative of $\boldsymbol{x}$.

Algorithm 3 Encoding k-order balanced sequences

Input: A sequence $\boldsymbol{s} \in \mathbb{F}_2^{n-\rho}$
Output: A sequence $\boldsymbol{x} \in \mathcal{D}(n,k)$

1: For every $t = k-1, \ldots, 0$, use $\mathcal{E}_K$ to find i_t, the balancing index of $\boldsymbol{s}^{(t)}$. Flip the i_t-prefix of $\boldsymbol{s}^{(t)}$ without adding the encoding of the index. Let $\widehat{\boldsymbol{s}}$ denote the result of this process.
2: Construct the indices sequence $\boldsymbol{v} = b(i_0) \circ \cdots \circ b(i_{k-1})$ of length $k\log n$.
3: Repeat Steps 1-2 with $\boldsymbol{v}$ as an input to receive $\widehat{\boldsymbol{v}}$ and the indices sequence $\boldsymbol{u}$ of length $k\log(k\log n)$.
4: Use a fixed encoding map to translate $\boldsymbol{u}$ to $\widehat{\boldsymbol{u}}$, a k-order balanced sequence of length $\Theta(\log\log n)$.
5: Let $\boldsymbol{x} = 0^{k-1} \circ \widehat{\boldsymbol{s}} \circ \widehat{\boldsymbol{v}} \circ \widehat{\boldsymbol{u}}$.
6: For every $t = k-1, \ldots, 0$, append to $\boldsymbol{x}^{(t)}$ a word $\boldsymbol{y}_t$ of length m_t such that: (Note that $\boldsymbol{x}$ is changed throughout the process)
  a: $\mathrm{Pref}_{k-t-1}(\boldsymbol{y}_t) = \boldsymbol{0}$,
  b: $\boldsymbol{x}^{(t)} \circ \boldsymbol{y}_t$ is balanced,
  c: $\boldsymbol{y}'_t$ is $(k-t-1)$-order balanced.
7: Return $\boldsymbol{x}$.

The suitable words $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{k-1}$ that satisfy the constraints mentioned in Step 6 can be generated using a method that uses similar techniques to those used in Section III in order to enumerate (ℓ,μ)-BdB sequences. Their lengths $m_0, \ldots, m_{k-1}$ are a function of the imbalance of the derivatives of $\boldsymbol{x}$ which can be upper bounded and set in advance independently of the input sequence.

Theorem 22. *Algorithm 3 returns a sequence $\boldsymbol{x} \in \mathcal{D}(n,k)$ that can be uniquely decoded to its input sequence $\boldsymbol{s}$. The algorithm uses $\rho = k\log n + \Theta(\log\log n)$ redundancy bits and its time complexity is $\mathcal{O}(n)$.*

Acknowledgment

S. Marcovich and E. Yaakobi were supported by the United States-Israel BSF grant no. 2018048. T. Etzion was supported by ISF grant no. 222/19.

REFERENCES

[1] S. Al-Bassam and B. Bose, "On balanced codes," *IEEE Transactions on Information Theory*, vol. 36, no. 2, pp. 406–408, 1990.

[2] N. Alon, E. E. Bergmann, D. Coppersmith, and A. M. Odlyzko, "Balancing sets of vectors," *IEEE Transactions on Information Theory*, vol. 34, no. 1, pp. 128–130, 1988.

[3] N. Alon, J. Bruck, F. F. Hassanzadeh, and S. Jain, "Duplication distance to the root for binary sequences," *IEEE Transactions on Information Theory*, vol. 63, no. 12, pp. 7793–7803, 2017.

[4] N. G. D. Bruijn, "A combinatorial problem," *Koninklijke Nederlandse Akademie v. Wetenschappen*, vol. 49, no. 49, pp. 758–764, 1946.

[5] A. H. Chan, R. A. Games, and E. L. Key, "On the complexities of de bruijn sequences," *Journal of Combinatorial Theory, Series A*, vol. 33, no. 3, pp. 233 – 246, 1982.

[6] P. Compeau, P. Pevzner, and G. Tesler, "How to apply de bruijn graphs to genome assembly," *Nature biotechnology*, no. 11, pp. 987–991, 2011.

[7] T. Cover, "Enumerative source encoding," *IEEE Transactions on Information Theory*, vol. 19, no. 1, pp. 73–77, 1973.

[8] S. Dolinar, T.-M. Ko, and R. McEliece, "Some vlsi decompositions of the de Bruijn graph," *Discrete Mathematics*, vol. 106-107, pp. 189 – 198, 1992.

[9] T. Etzion, "The depth distribution-a new characterization for linear codes," *IEEE Transactions on Information Theory*, vol. 43, no. 4, pp. 1361–1363, 1997.

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

[11] H. M. Fredricksen, "A class of nonlinear de Bruijn cycles," *Journal of Combinatorial Theory*, vol. 19, pp. 192–199, 1975.

[12] T. Goka, "An operator on binary sequences," *SIAM Review*, vol. 12, pp. 264–266, 1970.

[13] S. W. Golomb, "Shift register sequences," *San Francisco: Holden-Day*, 1967.

[14] I. Good, "Normal recurring decimals," *Journal of the London Mathematical Society*, vol. 21, pp. 167–169, 1946.

[15] T. Hasunuma and Y. Shibata, "Embedding de Bruijn, kautz and shuffle-exchange networks in books," *Discrete Applied Mathematics*, vol. 78, no. 1, pp. 103 – 116, 1997.

[16] H. D. L. Hollmann and K. A. S. Immink, "Performance of efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 913–918, 1991.

[17] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for dna sequence profiles," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3125–3146, 2016.

[18] D. Knuth, "Efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 51–53, 1986.

[19] F. T. Leighton, M. Lepley, and G. L. Miller, "Layouts for the shuffle-exchange graph based on the complex plane diagram," *SIAM Journal on Algebraic Discrete Methods*, vol. 5, no. 2, pp. 202–215, 1984.

[20] A. Lempel, "On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers," *IEEE Transactions on Computers*, vol. C-19, no. 12, pp. 1204–1209, 1970.

[21] U. M. Maurer, "Asymptotically-tight bounds on the number of cycles in generalized de Bruijn-Good graphs," *Discrete Applied Mathematics*, vol. 37-38, pp. 421 – 436, 1992.

[22] F. J. Mowle, "Relations between p_n cycles and stable feedback shift registers," *IEEE Transactions on Computers*, vol. C-15, pp. 375–378, 1966.

[23] ——, "An algorithm for generating stable shift registers of order n," *J. Assoc. Comput. Mach.*, vol. 14, pp. 529–542, 1967.

[24] M. B. Nathanson, "Derivatives of binary sequences," *SIAM Journal on Applied Mathematics*, vol. 21, no. 3, pp. 407–412, 1970.

[25] A. Ralston, "A new memoryless algorithm for de Bruijn sequences," *Journal of Algorithms*, vol. 2, pp. 50–62, 1981.

[26] R. M. Roth, P. H. Siegel, and A. Vardy, "High-order spectral-null codes-constructions and bounds," *IEEE Transactions on Information Theory*, vol. 40, no. 6, pp. 1826–1840, 1994.

[27] K. A. Schouhamer Immink and J. H. Weber, "Very efficient balanced codes," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 188–192, 2010.

[28] V. Skachek, T. Etzion, and R. M. Roth, "Efficient encoding algorithm for third-order spectral-null codes," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 846–851, 1998.

[29] L. Song, F. Geng, Z. Gong, B. Li, and Y. Yuan, "Super-robust data storage in DNA by de bruijn graph-based decoding," 2020.

[30] M. A. Sridhar and C. S. Raghavendra, "Fault-tolerant networks based on the de Bruijn graph," *IEEE Transactions on Computers*, vol. 40, no. 10, pp. 1167–1174, 1991.

[31] L. G. Tallini and B. Bose, "Balanced codes with parallel encoding and decoding," *IEEE Transactions on Computers*, vol. 48, no. 8, pp. 794–814, 1999.

[32] ——, "On efficient high-order spectral-null codes," *IEEE Transactions on Information Theory*, vol. 45, no. 7, pp. 2594–2601, 1999.

[33] L. G. Tallini, R. M. Capocelli, and B. Bose, "Design of some new efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 790–802, 1996.

[34] T. van Aardenne-Ehrenfest and N. G. de Bruijn, "Circuits and trees in oriented linear graphs," *Simon Stevin : Wis-en Natuurkundig Tijdschrif*, vol. 28, pp. 203–217, 1951.

[35] J. H. Weber and K. A. Schouhamer Immink, "Knuth's balanced codes revisited," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1673–1679, 2010.