A Method for Constructing Decodable de Bruijn Sequences

Chris J. Mitchell, *Member IEEE*, Tuvi Etzion, *Member IEEE*, and Kenneth G. Paterson

Abstract— **In this paper we present two related methods of construction for de Bruijn sequences, both based on interleaving "smaller" de Bruijn sequences. Sequences obtained using these construction methods have the advantage that they can be "decoded" very efficiently, i.e., the position within the sequence of any particular "window" can be found very simply. Sequences with simple decoding algorithms are of considerable practical importance in position location applications.**

Index Terms— **Decoding, de Bruijn graph, window sequence, de Bruijn sequence.**

I. Introduction

A. De Bruijn Sequences and the Decoding Problem

De Bruijn sequences, i.e., periodic sequences with elements taken from a finite alphabet in which every possible v-tuple of elements appears precisely once in a period (for some v), have been well studied for many years, see, for example, [1], [2]. Many constructions are known, and a useful survey has been given by Fredricksen [2].

However, the decoding problem, i.e., the problem of discovering the position within a particular sequence of any specified v-tuple, has been much less well studied. This is notwithstanding the fact that for certain well-known practical applications of de Bruijn sequences, including their use for position location (see, for example, [3]–[5]), the decoding problem is an important one. Over and above its practical significance, the decoding problem has been listed by Chung, Diaconis, and Graham [6] as one of the "fundamental questions" for the study of de Bruijn sequences.

Previous work on the decoding problem can be summarized as follows.

- The obvious approach is the "brute-force" method of storing a lookup table of the positions in the sequence of all possible v-tuples. Alternatively, successive states of the sequence can be generated until the desired v-tuple is found. These methods are too inefficient for use with anything except relatively short sequences.
- A "milestone" approach to the decoding of the de Bruijn sequences derived from m-sequences can be derived from the work of Petriu [5]. The idea is a simple development of the brute-force approach of stepping a linear feedback shift register, equipped with feedbacks which generate the m-sequence, through all possible states until the desired tuple is obtained. The milestone idea is to store every nth shift register state (i.e., every nth tuple in the sequence) for some n; these form the milestone values. A particular v-tuple to be "decoded" is then used to define the initial state of the register, which is stepped until a stored value is obtained. This approach does not reduce the computational complexity of decoding below the brute-force value; it is simply a time/space tradeoff (albeit not without practical merit for relatively small values of v).
- Another decoding method also applies to the de Bruijn sequences derived from binary m-sequences. Consider the successive states of a v-stage "Galois" feedback register, equipped with feedbacks corresponding to a primitive polynomial. It is a well-established fact (see, for example, [7]) that, if these states (binary v-tuples) are regarded as binary vectors with respect to an appropriate basis, then the successive states are simply successive powers of a primitive element in the finite field GF(2^v) when regarded as a v-dimensional vector space over GF(2). Hence finding the position of any given state in this sequence of states is precisely equivalent to finding discrete logarithms in this field.

  Massey and Liu [8] showed that there always exists a linear transformation mapping the sequence of states of a Galois register into the corresponding sequence of states of a "conventional" feedback register. This means that the decoding problem for m-sequences (and de Bruijn sequences derived from them) is equivalent to the discrete logarithm problem over GF(2^v). Although finding discrete logarithms is nontrivial, algorithms considerably more efficient than the brute-force approach are known; see, for example, [9].
- The only other method known to the authors applies to a different class of de Bruijn sequences; namely, those derived by repeated application of the inverse of Lempel's homomorphism [10]. Paterson and Robshaw [11] have shown that such sequences can be decoded recursively. Although this technique achieves a time/space tradeoff, its overall complexity remains essentially the same as the

Manuscript received July 14, 1995; revised April 2, 1996. The work of T. Etzion was supported in part by the EPSRC under Grant GR/K38847. The work of K. G. Paterson was supported by a Lloyds of London Tercentenary Foundation Research Fellowship.

C. J. Mitchell is with the Computer Science Department, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England.

T. Etzion is with the Computer Science Department, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England, on leave from the Computer Science Department, Technion—Israel Institute of Technology, 32000 Haifa, Israel.

K. G. Paterson is with the Mathematics Department, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England.

Publisher Item Identifier S 0018-9448(96)05757-4.

brute-force methods, unless some information is already available about the approximate location of the v-tuple in the sequence.

It should be clear that all the existing approaches have significant limitations; even the discrete logarithm method is computationally complex, and is nontrivial to implement.

In this paper we present two related methods of construction for de Bruijn sequences. We also describe algorithms which can be used to decode these sequences much more efficiently than any of the previously known techniques.

B. Preliminary Definitions and Notation

We first set up some notation which we will use throughout the paper.

We are concerned here with c-ary periodic sequences, where by the term c-ary we mean sequences whose elements are drawn from the set $\{0, 1, \cdots, c-1\}$. We refer throughout to c-ary cycles of period n, by which we mean periodic sequences $(s_0, s_1, \cdots, s_{n-1})$ where $s_i \in \{0, 1, \cdots, c-1\}$ for every $i, (0 \le i < n)$.

If $\boldsymbol{t} = (t_0, t_1, \cdots, t_{v-1})$ is a c-ary v-tuple (i.e., $t_i \in \{0, 1, \cdots, c-1\}$ for every i, $(0 \le i < v)$), and $\boldsymbol{s} = (s_0, s_1, \cdots, s_{n-1})$ is a c-ary cycle of period n $(n \ge v)$, then we say that $\boldsymbol{t}$ *occurs in* $\boldsymbol{s}$ *at position* j if and only if

$$t_i = s_{i+j}$$

for every $i, (0 \le i < v)$, where $i + j$ is computed modulo n.

Throughout we will write $\mathbf{0}^i$ for the i-tuple of all zeros and $\mathbf{1}^i$ for the i-tuple of all ones.

If $\boldsymbol{s} = (s_0, s_1, \cdots, s_{n-1})$ is a c-ary cycle of period n, then we say that $\boldsymbol{s}$ is a *v-window sequence* if no c-ary v-tuple occurs in $\boldsymbol{s}$ in two distinct positions within a period of $\boldsymbol{s}$. Equivalently, $\boldsymbol{s}$ contains n distinct v-tuples in a period of the cycle. A c-ary *de Bruijn sequence of span* v is then simply a v-window sequence of period equal to c^v; equivalently, every possible c-ary v-tuple occurs precisely once in a period of a de Bruijn sequence.

A c-ary *punctured de Bruijn sequence of span* v (sometimes called a pseudorandom sequence) is a v-window sequence in which every c-ary v-tuple except for $\mathbf{0}^v$ occurs, and so a punctured de Bruijn sequence has period $c^v - 1$. A span-v de Bruijn sequence can be "punctured" by deleting one of the zeros in $\mathbf{0}^v$, and a punctured de Bruijn sequence can be transformed into a de Bruijn sequence by adding a zero to any one of the $c - 1$ occurrences of $\mathbf{0}^{v-1}$. Similarly, a c-ary *doubly punctured de Bruijn sequence of span* v is a v-window sequence in which every c-ary v-tuple occurs except for $\mathbf{0}^v$ and $\mathbf{1}^v$, and hence a doubly punctured de Bruijn sequence has period $c^v - 2$. A de Bruijn sequence can be "doubly punctured" by first puncturing it and then deleting one of the ones in $\mathbf{1}^v$, and a doubly punctured de Bruijn sequence can be transformed into a de Bruijn sequence by adding a zero to any of the $c - 1$ occurrences of $\mathbf{0}^{v-1}$, and adding a one to any of the $c - 1$ occurrences of $\mathbf{1}^{v-1}$.

If $\boldsymbol{s} = (s_0, s_1, \cdots, s_{n-1})$ and $\boldsymbol{t} = (t_0, t_1, \cdots, t_{n-1})$ are two cycles of the same length, n say, then the interleaving of these cycles, denoted $\mathcal{I}(\boldsymbol{s}, \boldsymbol{t})$, is defined to be the following cycle of length $2n$:

$$(s_0, t_0, s_1, t_1, \cdots, s_{n-1}, t_{n-1}).$$

II. An Interleaving Construction for Window Sequences

We now present a method for constructing a c-ary cycle with the window property.

A. The Construction Method

Before describing the method of construction we need the following definition.

Definition 1: If $\boldsymbol{a}$ is a c-ary v-window sequence of period n, then $\boldsymbol{a}$ is said to satisfy Condition A if and only if the following three conditions are met:

- n is even,
- $\boldsymbol{a}$ does not contain the all-zero v-tuple, and
- $\boldsymbol{a}$ contains $\mathbf{0}^{v-1}$ at position 0 (it may contain other occurrences of $\mathbf{0}^{v-1}$).

Construction 2: Suppose n, c, v are positive integers $(c \ge 2)$. Moreover, suppose that $\boldsymbol{a} = (a_0, a_1 \cdots, a_{n-1})$ is a c-ary v-window sequence of period n satisfying Condition A.

Let $\boldsymbol{b}$ be the cycle of period $n + 2$ obtained from $\boldsymbol{a}$ by inserting two extra zeros at the start of the cycle; this has the effect of replacing $\mathbf{0}^{v-1}$ at position 0 with $\mathbf{0}^{v+1}$. (Observe that $\boldsymbol{b}$ is "almost" a v-window sequence, with the single exception that $\mathbf{0}^v$ occurs twice in consecutive positions—this fact is of key importance to the construction method).

Now let

$$\boldsymbol{d} = \mathcal{I}(\boldsymbol{b}^{n/2}, \boldsymbol{a}^{n/2+1})$$

where, as throughout, $\boldsymbol{a}^s$ denotes the cycle obtained by concatenating s copies of cycle $\boldsymbol{a}$.

We can now state and prove the following result.

Theorem 3: Suppose n, c, v and $\boldsymbol{a}$ satisfy the conditions of Construction 2. If $\boldsymbol{d}$ is constructed from $\boldsymbol{a}$ using Construction 2 then $\boldsymbol{d}$ is a $2v$-window sequence of period $n(n + 2)$ which satisfies Condition A.

Proof: If $0 \le i < n$, let $\boldsymbol{a}_i$ be the v-tuple occurring in $\boldsymbol{a}$ at position i; similarly, if $0 \le j < n + 2$, let $\boldsymbol{b}_j$ be the v-tuple occurring in $\boldsymbol{b}$ at position j. Then the $2v$-tuples of $\boldsymbol{d}$ are precisely

$$\begin{array}{rll}
\mathcal{I}(\boldsymbol{b}_{2i}, \boldsymbol{a}_{2j}) & 0 \le i \le n/2, & 0 \le j \le n/2 - 1 \\
\mathcal{I}(\boldsymbol{b}_{2i+1}, \boldsymbol{a}_{2j+1}) & 0 \le i \le n/2, & 0 \le j \le n/2 - 1 \\
\mathcal{I}(\boldsymbol{a}_{2i}, \boldsymbol{b}_{2j+1}) & 0 \le i \le n/2 - 1, & 0 \le j \le n/2 \\
\mathcal{I}(\boldsymbol{a}_{2i+1}, \boldsymbol{b}_{2j}) & 0 \le i \le n/2 - 1, & 0 \le j \le n/2.
\end{array}$$

All tuples of this form occur exactly once in $\boldsymbol{d}$, because $(n, n + 2) = 2$.

Next observe that each of the above four classes consists of $n/2(n/2 + 1)$ distinct $2v$-tuples, and the four classes are pairwise-disjoint. This latter point follows because the set of tuples $\{\boldsymbol{a}_{2i}\}$ is precisely the same as the set of tuples $\{\boldsymbol{b}_{2i}\}$ (with the exception of $\mathbf{0}^v$), and the set of tuples $\{\boldsymbol{a}_{2i+1}\}$ is precisely the same as the set of tuples $\{\boldsymbol{b}_{2i+1}\}$ (again with the exception of $\mathbf{0}^v$). Hence $\boldsymbol{d}$ is a $2v$-window sequence.

It remains for us to show that $\boldsymbol{d}$ satisfies Condition A. First observe that $\boldsymbol{d}$ has period $n(n+2)$ which is even since n is even. Next observe that since $\boldsymbol{a}$ does not contain $\mathbf{0}^v$ it follows immediately that $\boldsymbol{d}$ cannot contain $\mathbf{0}^{2v}$. Finally, observe that, since $\mathbf{0}^{v+1}$ occurs at position 0 in $\boldsymbol{b}$ and $\mathbf{0}^{v-1}$ occurs at position 0 in $\boldsymbol{a}$, then $\mathbf{0}^{2v-1}$ occurs at position 0 in $\boldsymbol{d}$. The result follows. □

B. Application of the Construction Method

First observe that if c is odd then an example of a v-window sequence $\boldsymbol{a}$ of period $c^v - 1$ satisfying Condition A can be obtained by taking an appropriate cyclic shift of a c-ary span v punctured de Bruijn sequence. The cycle $\boldsymbol{d}$ resulting from an application of Construction 2 to $\boldsymbol{a}$ will then have period $(c^v - 1)(c^v + 1) = c^{2v} - 1$, and will be a c-ary span $2v$ punctured de Bruijn sequence (enabling the construction to be applied recursively).

The situation is not so convenient in the case where c is even (which obviously includes the binary case), where the best that can be done is to note that an appropriately shifted c-ary span v doubly punctured de Bruijn sequence (of period $c^v - 2$) is an example of a sequence satisfying Condition A. We will address this case in Sections III and IV below.

C. Example

Before proceeding we consider two simple examples of the construction method.

Example 4: Let $n = 6$, $c = 2$, and $v = 3$. Let $\boldsymbol{a}$ be the following cycle:

$$(0\quad 0\quad 1\quad 0\quad 1\quad 1).$$

Note that $\boldsymbol{a}$ is a 2-ary span 3 doubly punctured de Bruijn sequence. The cycle $\boldsymbol{b}$ is as follows:

$$(0\quad 0\quad 0\quad 0\quad 1\quad 0\quad 1\quad 1).$$

We then have

$$\begin{array}{lccccccccccccccc}
\boldsymbol{d} = \mathcal{I}(\boldsymbol{b}^3, \boldsymbol{a}^4) & & & & & & & & & & & & & & & \\
= & (0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
 & & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
 & & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
 & & 0 & 1 & 1 & 1 & 1). & & & & & & & & &
\end{array}$$

$\boldsymbol{d}$ is a binary 6-window sequence of period 48.

Example 5: Let $n = 8$, $c = 3$, and $v = 2$. Let $\boldsymbol{a}$ be the following cycle:

$$(0\quad 1\quad 1\quad 0\quad 2\quad 1\quad 2\quad 2).$$

Note that $\boldsymbol{a}$ is a 3-ary span 2 punctured de Bruijn sequence. The cycle $\boldsymbol{b}$ is as follows:

$$(0\quad 0\quad 0\quad 1\quad 1\quad 0\quad 2\quad 1\quad 2\quad 2).$$

Then $\boldsymbol{b}^{n/2}$ is

$$\begin{array}{ccccccccccccc}
(0 & 0 & 0 & 1 & 1 & 0 & 2 & 1 & 2 & 2 & 0 & 0 & 0 \\
 & 1 & 1 & 0 & 2 & 1 & 2 & 2 & 0 & 0 & 0 & 1 & 1 \\
 & 0 & 2 & 1 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 2 \\
 & 1 & 2 & 2) & & & & & & & & &
\end{array}$$

and $\boldsymbol{a}^{n/2+1}$ is

$$\begin{array}{ccccccccccccc}
(0 & 1 & 1 & 0 & 2 & 1 & 2 & 2 & 0 & 1 & 1 & 0 & 2 \\
 & 1 & 2 & 2 & 0 & 1 & 1 & 0 & 2 & 1 & 2 & 2 & 0 \\
 & 1 & 1 & 0 & 2 & 1 & 2 & 2 & 0 & 1 & 1 & 0 & 2 \\
 & 1 & 2 & 2). & & & & & & & & &
\end{array}$$

Hence $\boldsymbol{d}$ is

$$\begin{array}{cccccccccccccc}
(0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 2 & 0 & 1 & 2 & 2 \\
 & 1 & 2 & 2 & 0 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 2 & 1 \\
 & 1 & 1 & 2 & 0 & 2 & 2 & 0 & 1 & 1 & 2 & 1 & 2 & 0 \\
 & 0 & 2 & 0 & 1 & 0 & 2 & 1 & 2 & 1 & 0 & 0 & 1 & 2 \\
 & 1 & 1 & 0 & 2 & 2 & 2 & 1 & 0 & 2 & 0 & 2 & 0 & 0 \\
 & 1 & 1 & 1 & 1 & 0 & 0 & 2 & 2 & 1 & 1 & 2 & 2 & 2 \\
 & 2) & & & & & & & & & & & &
\end{array}$$

and $\boldsymbol{d}$ is a 3-ary span 4 punctured de Bruijn sequence.

D. A Decoding Algorithm

We now present a simple algorithm for decoding cycles which have been derived using Construction 2. This algorithm makes use of a decoder for the cycle $\boldsymbol{a}$ used as input to the construction.

Algorithm 6: Suppose n, c, v, and $\boldsymbol{a}$ satisfy the conditions of Construction 2, and that $\boldsymbol{d}$ has been constructed from $\boldsymbol{a}$ using this construction. Suppose also that the function E is a decoder for $\boldsymbol{a}$, i.e., if $\boldsymbol{x}$ is some v-tuple occurring in $\boldsymbol{a}$ then $0 \le E(\boldsymbol{x}) < n$ and $\boldsymbol{x}$ occurs in $\boldsymbol{a}$ at position $E(\boldsymbol{x})$.

Define the function $F: T \to \{0, 1, \cdots, n(n+2) - 1\}$ as follows, where T is the set of all c-ary $(2v)$-tuples which occur in $\boldsymbol{d}$. First suppose $\boldsymbol{x} \in T$, and let

$$\boldsymbol{x} = \mathcal{I}(\boldsymbol{y}, \boldsymbol{z}).$$

Let m be the unique solution $(\bmod\, n(n+2)/2)$ to the simultaneous congruences

$$m \equiv \begin{cases} E(\boldsymbol{z}), & \text{if } \boldsymbol{y} = \mathbf{0}^v \text{ or } E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is even} \\ E(\boldsymbol{y}), & \text{if } \boldsymbol{z} = \mathbf{0}^v \text{ or } E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is odd} \end{cases} \quad \bmod n$$

$$m \equiv \begin{cases} 0 \text{ or } 1, & \text{if } \boldsymbol{y} = \mathbf{0}^v \\ n+1 \text{ or } 0, & \text{if } \boldsymbol{z} = \mathbf{0}^v \\ E(\boldsymbol{y}) + 2, & \text{if } E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is even} \\ E(\boldsymbol{z}) + 1, & \text{if } E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is odd.} \end{cases} \quad \bmod n+2$$

Then let

$$F(\boldsymbol{x}) = \begin{cases} 2m, & \text{if } \boldsymbol{y} = \mathbf{0}^v \text{ or } E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is even} \\ 2m+1, & \text{if } \boldsymbol{z} = \mathbf{0}^v \text{ or } E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is odd} \end{cases}$$

Theorem 7: If $n, c, v, \boldsymbol{d}$ and F are defined as in Algorithm 6, then F is a decoder for $\boldsymbol{d}$.

Proof: It should be immediately clear that every c-ary $(2v)$-tuple will be covered by one of the four "cases" of the algorithm. We now consider each case in turn. We suppose throughout that $\boldsymbol{x}$ occurs at position p in $\boldsymbol{d}$, where $0 \le p < n(n+2)$ (we know that p is well-defined by Theorem 3).

If $\boldsymbol{y} = \mathbf{0}^v$ then $\boldsymbol{y}$ must occur in $\boldsymbol{b}$, and hence $\boldsymbol{x}$ occurs at an even position in $\boldsymbol{d}$, i.e., $p = 2q$ for some integer q. Thus given that $\mathbf{0}^v$ occurs at positions 0 and 1 in $\boldsymbol{b}$, we have that

$$q \equiv 0 \text{ or } 1 \pmod{n+2}.$$

Now $\boldsymbol{z}$ occurs at position $E(\boldsymbol{z})$ in $\boldsymbol{a}$ and hence

$$q \equiv E(\boldsymbol{z}) \pmod{n}.$$

Given that p is well-defined, F must be well-defined, and the correctness of the first case is established.

If $\boldsymbol{z} = \mathbf{0}^v$ then $\boldsymbol{z}$ must occur in $\boldsymbol{b}$, and hence p is odd, say $p = 2q + 1$ for some integer q. Thus given that $\mathbf{0}^v$ occurs at positions 0 and 1 in $\boldsymbol{b}$, we have that

$$p \equiv 2n + 3 \text{ or } 1 \pmod{2(n+2)}$$

and hence

$$q + 1 \equiv 0 \text{ or } 1 \pmod{n+2}.$$

In addition, $\boldsymbol{y}$ occurs at position $E(\boldsymbol{y})$ in $\boldsymbol{a}$, and hence

$$p \equiv 2E(\boldsymbol{y}) + 1 \pmod{2n}$$

and so

$$q \equiv E(\boldsymbol{y}) \pmod{n}.$$

As before, given that p is well-defined, F must be well-defined, and the correctness of the second case is established.

If $\boldsymbol{y}$ and $\boldsymbol{z}$ are both nonzero, then either

a) p is even, $p = 2q$ say, $\boldsymbol{y}$ occurs in $\boldsymbol{b}$ at position $E(\boldsymbol{y}) + 2$ and $\boldsymbol{z}$ occurs in $\boldsymbol{a}$ at position $E(\boldsymbol{z})$, or

b) p is odd, $p = 2q + 1$ say, $\boldsymbol{z}$ occurs in $\boldsymbol{b}$ at position $E(\boldsymbol{z}) + 2$ and $\boldsymbol{y}$ occurs in $\boldsymbol{a}$ at position $E(\boldsymbol{y})$.

In case a) we have

$$q \equiv E(\boldsymbol{y}) + 2 \pmod{n+2}$$

and

$$q \equiv E(\boldsymbol{z}) \pmod{n}.$$

In case b) we have

$$p \equiv 2E(\boldsymbol{z}) + 3 \pmod{2(n+2)}$$

i.e.,

$$q + 1 \equiv E(\boldsymbol{z}) + 2 \pmod{n+2}$$

and

$$p \equiv 2E(\boldsymbol{y}) \pmod{2n}$$

i.e.,

$$q \equiv E(\boldsymbol{y}) \pmod{n}.$$

The above discussion covers the final two cases, and the result follows. □

E. Complexity of Decoding

We now consider the complexity of the decoding method of Algorithm 6, when applied to a c-ary $(2v)$-window sequence $\boldsymbol{d}$ constructed from a c-ary v-window sequence $\boldsymbol{a}$ of period n using the technique of Construction 2.

Suppose it takes e arithmetic operations to find the position of a c-ary v-tuple in $\boldsymbol{a}$. Then it is not difficult to see that the number of arithmetic operations involved in decoding a single c-ary v-tuple is bounded above by $2e + \mathrm{EA}(n) + k$, where $\mathrm{EA}(n)$ is the number of operations required to find the unique solution (modulo $n(n + 2)/2$) to a pair of simultaneous congruences (modulo n and $n + 2$), and k is a small constant. Solving a pair of simultaneous congruences can be achieved using the well-known (and simple) Euclidean Algorithm.

III. A Related Construction Method for Window Sequences

We now present a second method for constructing a c-ary cycle with the window property. This method is a variant of the method presented in the previous section—it has advantages for cycles with even-size alphabets.

A. The Construction Method

Before describing the method of construction we need the following definition.

Definition 8: If $\boldsymbol{a}$ is a c-ary v-window sequence of period n, then $\boldsymbol{a}$ is said to satisfy Condition B if and only if the following three conditions are met:

- $2 \mid n$ and $4 \nmid n$,
- $\boldsymbol{a}$ does not contain $\mathbf{0}^v$ or $\mathbf{1}^v$, and
- $\boldsymbol{a}$ does contain at least one occurrence of $\mathbf{0}^{v-1}$ and at least one occurrence of $\mathbf{1}^{v-1}$.

Construction 9: Suppose n, c, v are positive integers $(c \geq 2)$. Moreover, suppose that $\boldsymbol{a} = (a_0, a_1 \cdots, a_{n-1})$ is a c-ary v-window sequence of period n satisfying Condition B. Let $\boldsymbol{b}$ be a cycle of period $n + 4$ obtained from $\boldsymbol{a}$ by replacing an occurrence of $\mathbf{0}^{v-1}$ with $\mathbf{0}^{v+1}$ and an occurrence of $\mathbf{1}^{v-1}$ with $\mathbf{1}^{v+1}$. Observe that $\boldsymbol{b}$ is "almost" a v-window sequence, with the two exceptions that $\mathbf{0}^v$ and $\mathbf{1}^v$ both occur twice in consecutive positions—this fact is of key importance to the construction method.

Now let

$$\boldsymbol{d}' = \mathcal{I}(\boldsymbol{b}^{n/2}, \boldsymbol{a}^{n/2+2}).$$

Finally, derive $\boldsymbol{d}$ from $\boldsymbol{d}'$ by inserting a zero followed by a one after one of the $(2v - 1)$-tuples equal to $\mathcal{I}(\mathbf{1}^{v-1}, \mathbf{0}^{v-1})$ followed by a one, in order to make it into a $(2v + 1)$-tuple equal to $\mathcal{I}(\mathbf{1}^v, \mathbf{0}^v)$ followed by a one.

We can now state and prove the following result.

Theorem 10: Suppose n, c, v, and $\boldsymbol{a}$ satisfy the conditions of Construction 9. If $\boldsymbol{d}$ is constructed from $\boldsymbol{a}$ using Construction 9 then $\boldsymbol{d}$ is a $(2v)$-window sequence of period $n(n+4)+2$ which satisfies Condition B.

Proof: The proof that this construction works is almost identical to that of Theorem 7. Using the same argument as given in that proof, it is straightforward to see that $\boldsymbol{d}'$ is a $2v$-window sequence. We therefore need only establish that

- the derivation of $\boldsymbol{d}$ from $\boldsymbol{d}'$ is well-defined,
- $\boldsymbol{d}$ is a $2v$-window sequence, and
- $\boldsymbol{d}$ satisfies Condition B.

The first point follows from the observation that, since $\boldsymbol{a}$ contains $\mathbf{0}^{v-1}$, and $\boldsymbol{b}$ contains $\mathbf{1}^{v+1}$, then $\boldsymbol{d}$ must contain the $(2v-1)$-tuple: $\mathcal{I}(\mathbf{1}^{v-1}, \mathbf{0}^{v-1})$ followed by a one. To establish the second point we note that the two "extra" $2v$-tuples which $\boldsymbol{d}$ contains are $\mathcal{I}(\mathbf{1}^{v}, \mathbf{0}^{v})$ and $\mathcal{I}(\mathbf{0}^{v}, \mathbf{1}^{v})$. Since $\boldsymbol{a}$ does not contain $\mathbf{0}^v$ or $\mathbf{1}^v$, then neither of these $2v$-tuples occurs in $\boldsymbol{d}'$. Hence, since $\boldsymbol{d}'$ is a $2v$-window sequence then so is $\boldsymbol{d}$.

Finally, we need to show that $\boldsymbol{d}$ satisfies Condition B. First note that $\boldsymbol{d}$ has period $n(n+4)+2$, which, since $2 \mid n$, satisfies $2 | n(n+4)+2$ and $4 \nmid n(n+4)+2$. Secondly, since $\boldsymbol{a}$ does not contain $\mathbf{0}^v$ or $\mathbf{1}^v$, then $\boldsymbol{d}$ does not contain $\mathbf{0}^{2v}$ or $\mathbf{1}^{2v}$. Third, $\boldsymbol{d}$ contains at least one occurrence of $\mathbf{0}^{2v-1}$ and at least one occurrence of $\mathbf{1}^{2v-1}$ by an exactly analogous argument to that used to show that $\boldsymbol{d}'$ contains an alternating $(2v-1)$-tuple.

The result now follows. □

B. Application of the Construction Method

Analogously to Section III-B, if c is even then an example of a v-window sequence $\boldsymbol{a}$ of period c^v-2 satisfying Condition B can be obtained by taking a c-ary span v doubly punctured de Bruijn sequence. The cycle $\boldsymbol{d}$ resulting from the application of Construction 9 to $\boldsymbol{a}$ will have period $(c^v-2)(c^v+2)+2 = c^{2v}-2$, and will also be a doubly punctured de Bruijn sequence (since it satisfies Condition B and has period $c^{2v}-2$), thus enabling the construction to be applied recursively.

Hence Constructions 2 and 9 provide a pair of methods for recursively generating de Bruijn sequences for all alphabet sizes; both methods double the window length at each iteration. For the odd-size alphabet case we have already seen how a computationally very simple decoder for a double-length window sequence can be derived from a decoder for the single-length window sequence used to construct it. In the sequel, we will demonstrate a corresponding simple recursive decoder for the even-size alphabet case.

C. Example

Before proceeding we consider a simple example of the construction method (and how the cycle produced can be made into a de Bruijn sequence).

Example 11: Let $n=6$, $c=2$, and $v=3$. Let $\boldsymbol{a}$ be the following cycle:

$$(0\quad 0\quad 1\quad 0\quad 1\quad 1).$$

Note that $\boldsymbol{a}$ is a 2-ary span 3 doubly punctured de Bruijn sequence. The cycle $\boldsymbol{b}$ is as follows:

$$(0\quad 0\quad 0\quad 0\quad 1\quad 0\quad 1\quad 1\quad 1\quad 1).$$

We then have

$$\begin{array}{rl} \boldsymbol{d}' &= \mathcal{I}(\boldsymbol{b}^3, \boldsymbol{a}^5) \\ &= (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1 \\ & \quad\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0 \\ & \quad\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1 \\ & \quad\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0 \\ & \quad\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0 \\ & \quad\ \ 1\ \ 1\ \ 1\ \ 1). \end{array}$$

$\boldsymbol{d}'$ is a binary 6-window sequence of period 60, and the only 6-tuples missing are

$$(0\quad 0\quad 0\quad 0\quad 0\quad 0)$$
$$(1\quad 1\quad 1\quad 1\quad 1\quad 1)$$
$$(0\quad 1\quad 0\quad 1\quad 0\quad 1)$$

and

$$(1\quad 0\quad 1\quad 0\quad 1\quad 0).$$

The sequence $\boldsymbol{d}$ is obtained from $\boldsymbol{d}'$ by inserting an extra zero and one following the tuple "10101." Underlining the inserted bits, we obtain

$$\begin{array}{rl} \boldsymbol{d} = & (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0 \\ & \quad\ \ 1\ \ 0\ \ 1\ \ \underline{0}\ \ \underline{1}\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0 \\ & \quad\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0 \\ & \quad\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0 \\ & \quad\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1) \end{array}$$

which is a binary, span 6, doubly punctured de Bruijn sequence. To make $\boldsymbol{d}$ into a de Bruijn sequence we insert an extra zero and one to make the (unique) all-zero and all-one 5-tuples into 6-tuples, yielding the following binary de Bruijn sequence of period 64 (the two added bits are underlined):

$$\begin{array}{l} (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ \underline{0}\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0 \\ \quad\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0 \\ \quad\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ \underline{1}\ \ 0\ \ 1 \\ \quad\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1 \\ \quad\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1). \end{array}$$

D. A Decoding Algorithm

We next present a simple algorithm for decoding cycles which have been derived using Construction 9; this decoding method is very similar to that presented in Algorithm 6 above. As with that algorithm, use is made of a decoder for the cycle $\boldsymbol{a}$ used as input to the construction.

Algorithm 12: Suppose n, c, v, and $\boldsymbol{a}$ satisfy the conditions of Construction 9, and that $\boldsymbol{d}$ has been constructed from $\boldsymbol{a}$ using this construction. Suppose also that the function E is a decoder for $\boldsymbol{a}$, i.e., if $\boldsymbol{x}$ is some v-tuple occurring in $\boldsymbol{a}$ then $0 \le E(\boldsymbol{x}) < n$ and $\boldsymbol{x}$ occurs in $\boldsymbol{a}$ at position $E(\boldsymbol{x})$. Similarly, suppose that the function E' is a decoder for $\boldsymbol{b}$ (defined only for the tuples which occur in $\boldsymbol{a}$, and hence E' is well-defined).

Suppose also that the particular occurrences of $\mathbf{0}^{v-1}$ and $\mathbf{1}^{v-1}$ which are modified in deriving $\boldsymbol{b}$ from $\boldsymbol{a}$, occur at positions s and s' in $\boldsymbol{a}$, respectively. Suppose also, without loss of generality, that $s < s'$. Finally, suppose that the $(2v-1)$-tuple of alternating zeros and ones, which is augmented to obtain $\boldsymbol{d}$ from $\boldsymbol{d}'$, occurs at position t in $\boldsymbol{d}'$.

Define the functions $F'\colon T' \to \{0, 1, \cdots, n(n+2)-1\}$ and $F\colon T \to \{0, 1, \cdots, n(n+2)+1\}$ as follows, where T is the set of all c-ary $(2v)$-tuples which occur in $\boldsymbol{d}$ and T' is the set of all c-ary $(2v)$-tuples which occur in $\boldsymbol{d}'$. First suppose $\boldsymbol{x} \in T'$, and let

$$\boldsymbol{x} = \mathcal{I}(\boldsymbol{y}, \boldsymbol{z}).$$

Let m be the unique solution $(\operatorname{mod} n(n+4)/2)$ to the simultaneous congruences

$$m \equiv \begin{cases} E(\boldsymbol{z}), & \text{if } \boldsymbol{y} = \mathbf{0}^v \text{ or } \boldsymbol{y} = \mathbf{1}^v \text{ or} \\ & \quad E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is even} \\ E(\boldsymbol{y}), & \text{if } \boldsymbol{z} = \mathbf{0}^v \text{ or } \boldsymbol{z} = \mathbf{1}^v \text{ or} \\ & \quad E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is odd} \end{cases} \pmod{n}$$

$$m \equiv \begin{cases} s \text{ or } s+1, & \text{if } \boldsymbol{y} = \mathbf{0}^v \\ s'+2 \text{ or } s'+3, & \text{if } \boldsymbol{y} = \mathbf{1}^v \\ s-1 \text{ or } s, & \text{if } \boldsymbol{z} = \mathbf{0}^v \\ s'+1 \text{ or } s'+2, & \text{if } \boldsymbol{z} = \mathbf{1}^v \\ E'(\boldsymbol{y}), & \text{if } E(\boldsymbol{z}) - E(\boldsymbol{y}) \\ & \quad \text{is even} \\ E'(\boldsymbol{z}) - 1, & \text{if } E(\boldsymbol{z}) - E(\boldsymbol{y}) \\ & \quad \text{is odd.} \end{cases} \pmod{n+4}$$

Then let

$$F'(\boldsymbol{x}) = \begin{cases} 2m, & \text{if } \boldsymbol{y} = \mathbf{0}^v \text{ or } \boldsymbol{y} = \mathbf{1}^v \text{ or} \\ & \quad E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is even} \\ 2m+1, & \text{if } \boldsymbol{z} = \mathbf{0}^v \text{ or } \boldsymbol{z} = \mathbf{1}^v \text{ or} \\ & \quad E(\boldsymbol{z}) - E(\boldsymbol{y}) \text{ is odd.} \end{cases}$$

Finally, if $\boldsymbol{x} \in T$, let

$$F(\boldsymbol{x}) = \begin{cases} F'(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in T' \text{ and } F'(\boldsymbol{x}) < t \\ t, & \text{if } \boldsymbol{x} = \mathcal{I}(\mathbf{1}^v, \mathbf{0}^v) \\ t+1, & \text{if } \boldsymbol{x} = \mathcal{I}(\mathbf{0}^v, \mathbf{1}^v) \\ F'(\boldsymbol{x}) + 2, & \text{if } \boldsymbol{x} \in T' \text{ and } F'(\boldsymbol{x}) \geq t. \end{cases}$$

Remark 13: It is important to note that the function E' can very simply be derived from E as follows. Suppose $\boldsymbol{x}$ is a v-tuple occurring in $\boldsymbol{a}$. Suppose also that s and s' are as defined in Algorithm 12 (and $s < s'$). Then

- if $0 \leq E(\boldsymbol{x}) \leq s$ then $E'(\boldsymbol{x}) = E(\boldsymbol{x})$,
- if $s < E(\boldsymbol{x}) \leq s'$ then $E'(\boldsymbol{x}) = E(\boldsymbol{x}) + 2$, and
- if $s' < E(\boldsymbol{x}) < n$ then $E'(\boldsymbol{x}) = E(\boldsymbol{x}) + 4$.

Theorem 14: If $n, c, v, \boldsymbol{d}$, and F are defined as in Algorithm 12, then F is a decoder for $\boldsymbol{d}$.

Proof: Rather than go through the proof in great detail we observe that it follows using a very similar argument to that used to establish Theorem 7. It should be immediately clear that every c-ary $(2v)$-tuple will be covered by one of the six 'cases' of the algorithm. The six individual cases then follow using exactly analogous arguments to those employed to deal with the four cases in the proof of Theorem 7. The change from F' to F is necessary to "correct" for the addition of the extra 0 and 1 to derive $\boldsymbol{d}$ from $\boldsymbol{d}'$. □

E. Complexity of Decoding

We complete this section by briefly considering the complexity of the decoding method of Algorithm 12, when applied to a c-ary $(2v)$-window sequence $\boldsymbol{d}$ constructed from a c-ary v-window sequence $\boldsymbol{a}$ of period n using the technique of Construction 9. It should be clear that, because of the great similarity between the two algorithms, the complexity of Algorithm 12 is approximately the same as that of Algorithm 6, with the exception that, for each iteration, there is a need to store the values of s, s', and t.

Hence, if it takes e arithmetic operations to find the position of a c-ary v-tuple in $\boldsymbol{a}$, then the number of arithmetic operations involved in decoding a single c-ary v-tuple is bounded above by $2e + \mathrm{EA}'(n) + k'$, where $\mathrm{EA}'(n)$ is the number of operations required to find the unique solution (modulo $n(n+4)/2$) to a pair of simultaneous congruences (modulo n and $n+4$), and k' is a small constant. Storage space is also required for the three values s, s', and t.

IV. An Alternative Approach for Even-Size Alphabets

Because Construction 2 only enabled the recursive construction of de Bruijn sequences with odd-size alphabets, Construction 9 was devised to deal with the even-size alphabet case. However, an alternative approach exists for recursively constructing de Bruijn sequences with even-size alphabets using Construction 2 directly. We sketch that approach here.

Suppose $c > 1$ is even and $\boldsymbol{a}$ is a c-ary span v de Bruijn sequence which ends with $\mathbf{1}^v$ and begins with $\mathbf{0}^v$ (there are always such de Bruijn sequences—for example, the "prefer ones" sequence [2]). By deleting a zero from $\mathbf{0}^v$ and a one from $\mathbf{1}^v$ we obtain a doubly punctured de Bruijn sequence $\boldsymbol{a}'$ with the property that the sequence ends with $\mathbf{1}^{v-1}$ and begins with $\mathbf{0}^{v-1}$; we call this *Property C.*

We give a method involving Construction 2 which produces a new doubly punctured de Bruijn sequence also having Property C, and thus the method can be iterated.

We first apply Construction 2 to $\boldsymbol{a}'$ to obtain a sequence $\boldsymbol{d}$ of period $c^{2v} - 2c^v$. The following properties of $\boldsymbol{d}$ are a consequence of Theorem 3 and the construction method.

1) $\boldsymbol{d}$ is a $2v$-window sequence.
2) $\boldsymbol{d}$ begins with $\mathbf{0}^{2v-1}$ and ends with $\mathbf{1}^{2v-2}$.
3) $\boldsymbol{d}$ contains all $2v$-tuples, except for $\mathbf{0}^{2v}$ and the tuples $\mathcal{I}(\boldsymbol{w}, \mathbf{1}^v)$ and $\mathcal{I}(\mathbf{1}^v, \boldsymbol{w})$, where $\boldsymbol{w}$ is an arbitrary v-tuple.

Now let $\boldsymbol{e}$ denote the sequence obtained from $\boldsymbol{a}$ by deleting a one from $\mathbf{1}^v$ and shifting the resulting sequence right by $(v-1)$ places. Thus $\boldsymbol{e}$ has period $c^v - 1$ and begins with $\mathbf{1}^{v-1}$ followed by a zero. Let

$$\boldsymbol{f} = \mathcal{I}(\boldsymbol{e}, \mathbf{1}^{c^v - 1}).$$

Then $\boldsymbol{f}$ has period $2c^v - 2$ and begins with the $2v$-tuple $\mathbf{1}^{2v-2}$ followed by a zero and a one.

It is easily checked that, for each $\boldsymbol{w} \neq \mathbf{1}^v$, $\boldsymbol{f}$ contains as $2v$-tuples both $\mathcal{I}(\boldsymbol{w}, \mathbf{1}^v)$ and $\mathcal{I}(\mathbf{1}^v, \boldsymbol{w})$, neither of which occurs in $\boldsymbol{d}$.

From Property 2 above, $\boldsymbol{d}$ contains an occurrence of the $2v$-tuple made up of $\mathbf{1}^{2v-2}$ followed by two zeros (i.e., the "conjugate" of the first $2v$-tuple of $\boldsymbol{f}$) at position $c^{2v} -$

$2c^v - 2v + 2$. Hence $\boldsymbol{f}$ can be joined into $\boldsymbol{d}$ at position $c^{2v} - 2c^v - 2v + 2$ using Lempel's cycle joining method [10]. We obtain a new sequence $\boldsymbol{d}'$ which is a doubly punctured de Bruijn sequence satisfying Condition C.

It is not difficult to see how the decoding of $\boldsymbol{d}'$ can be reduced to the decoding of $\boldsymbol{a}'$. We leave the details to the reader, noting only that the complexity of the resulting decoding algorithm is marginally greater than that of Algorithm 6.

V. Summary and Conclusions

A. Decoding Long de Bruijn Sequences

We now briefly consider how de Bruijn sequences can be recursively constructed using Constructions 2 and 9, and, in addition, how they can be recursively decoded. Suppose we wish to construct and subsequently decode a span v de Bruijn sequence over a c-ary alphabet. Suppose also that $v = 2^h v'$, where v' is odd.

1) Odd-Size Alphabets: We start by considering use of Construction 2, and hence suppose c is odd. First construct, by some means, a c-ary span v' de Bruijn sequence $\boldsymbol{a}'$. In addition, a decoding algorithm needs to be provided for this cycle. Note that if c is composite then, using a special case of [12, Lemma 5.1], a c-ary span v' cycle can be constructed by combining span v' cycles over alphabets of sizes equal to the prime factors of c, and decoding the combined cycle can be reduced to decoding the component cycles.

Next derive a punctured de Bruijn sequence $\boldsymbol{a}$ from $\boldsymbol{a}'$ by deleting a single zero from $\mathbf{0}^{v'}$. We can now recursively apply Construction 2 h times to $\boldsymbol{a}$, obtaining a punctured de Bruijn sequence after each iteration. The final output will be a punctured c-ary de Bruijn sequence of span $v = 2^h v'$. This cycle can then be decoded by recursively applying Algorithm 6 h times, which (by the discussion in Section III-E) will involve at most 2^h decodings of the span v' cycle $\boldsymbol{a}$, together with the solution to $2^h - 1$ pairs of simultaneous congruences.

2) Even-Size Alphabets: We next consider use of Construction 9, and hence suppose c is even. As previously, construct, by some means, a c-ary span v' de Bruijn sequence $\boldsymbol{a}'$ with a decoding method. Again as previously, if c is composite then the methods of [12] can be used to simplify the decoding of the span v' cycle.

Next derive a doubly punctured de Bruijn sequence $\boldsymbol{a}$ from $\boldsymbol{a}'$ by deleting a single zero from $\mathbf{0}^{v'}$ and a single one from $\mathbf{1}^{v'}$. We can now recursively apply Construction 9 h times to $\boldsymbol{a}$.

The final output will be a doubly punctured c-ary de Bruijn sequence of span $v = 2^h v'$. This cycle can then be decoded by recursively applying Algorithm 12 h times, which (by the discussion in Section III-E) will involve at most 2^h decodings of the span v' cycle $\boldsymbol{a}$, together with the solution to $2^h - 1$ pairs of simultaneous congruences and the storage of $3h$ values.

Alternatively, the approach of Section IV can be used to produce a doubly punctured de Bruijn sequence of span $2^h v'$. This cycle can be decoded using an algorithm based on Algorithm 6, and having complexity roughly the same as for the case covered in Section V-A1.

3) Decoding Complexity: It should be clear that, given $h > 0$, the described approaches are far more efficient than any of the previously known methods for both odd- and even-size alphabets. In the "best case," where $v = 2^h$, decoding requires the solution of $v - 1$ pairs of simultaneous congruences (involving numbers of size at most c^v) and v decodings of the trivial sequence; hence the complexity of decoding is $O(v^2)$, i.e., it is polynomial in the span of the de Bruijn sequence.

B. Future Work

We conclude by briefly noting two areas for further work.

- Similar constructions to those described can be used to recursively construct Perfect Factors in the de Bruijn graph, with corresponding simple decoding algorithms. Perfect Factors have previously been studied because of their importance in constructing Perfect Maps, see, for example [12]–[15], and readily decoded Perfect Factors will enable the construction of Perfect Maps with simpler decoding algorithms (see [16]).
- Decoding cycles with large odd window length is still nontrivial, and further refinement of existing techniques (possibly combined with new techniques) remains a desirable goal.

Acknowledgment

The authors wish to thank an anonymous referee for pointing out that, in an unpublished paper, H. Fredricksen has shown how to decode the "prefer ones," "prefer sames," and "lexicographic composition" de Bruijn sequences using techniques like the milestone results of Petriu.

References

[1] N. de Bruijn, "A combinatorial problem," *Proc. Nederlandse Akademie van Wetenschappen*, vol. 49, pp. 758–764, 1946.

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

[3] J. Bondy and U. Murty, *Graph Theory with Applications*. Amsterdam, The Netherlands: Elsevier, 1976.

[4] J. Burns and C. Mitchell, "Coding schemes for two-dimensional position sensing," in *Cryptography and Coding III*, M. Ganley, Ed. Oxford, UK: Oxford Univ. Press, 1993, pp. 31–66.

[5] E. Petriu, "New pseudorandom/natural code conversion method," *Electron. Lett.*, vol. 24, pp. 1358–1359, 1988.

[6] F. Chung, P. Diaconis, and R. Graham, "Universal cycles for combinatorial structures," *Discr. Math.*, vol. 110, pp. 43–59, 1992.

[7] S. Golomb, *Shift Register Sequences*. San Francisco, CA: Holden-Day, 1967.

[8] J. Massey and R. Liu, "Equivalence of nonlinear shift-registers," *IEEE Trans. Inform. Theory*, vol. IT-10, pp. 378–379, 1964.

[9] A. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance," in *Advances in Cryptology: Proc. EUROCRYPT '84*, T. Beth, N. Cot, and I. Ingemarsson, Eds. Berlin, Germany: Springer-Verlag, 1985, pp. 224–314.

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

[11] K. Paterson and M. Robshaw, "Storage efficient decoding for a class of binary de Bruijn sequences," *Discr. Math.*, vol. 138, pp. 327–341, 1995.

[12] K. Paterson, "Perfect factors in the de Bruijn graph," *Des., Codes Cryptogr.*, vol. 5, pp. 115–138, 1995.

[13] T. Etzion, "Constructions for perfect maps and pseudo-random arrays," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1308–1316, 1988.

[14] C. Mitchell, "Constructing c-ary perfect factors," *Des., Codes Cryptogr.*, vol. 4, pp. 341–368, 1994.

[15] ——, "New c-ary perfect factors in the de Bruijn graph," in *Codes and Cyphers* (Proc. 4th IMA Conf. on Cryptography and Coding, Cirencester, UK, Dec. 1993), P. Farrell, Ed. Southend, UK: Formara Ltd., 1995, pp. 299–313.

[16] C. Mitchell and K. Paterson, "Decoding perfect maps," *Des., Codes Cryptogr.*, vol. 4, pp. 11–30, 1994.