manuscript No.
(will be inserted by the editor)

On Binary de Bruijn Sequences from LFSRs with Arbitrary Characteristic Polynomials

Zuling Chang · Martianus Frederic Ezerman · San Ling · Huaxiong Wang

Received: date / Accepted: date

Abstract We propose a construction of de Bruijn sequences by the cycle joining method from linear feedback shift registers (LFSRs) with arbitrary characteristic polynomial $f(x)$. We study in detail the cycle structure of the set $\Omega(f(x))$ that contains all sequences produced by a specific LFSR on distinct inputs and provide a fast way to find a state of each cycle. This leads to an efficient algorithm to find all conjugate pairs between any two cycles, yielding the adjacency graph. The approach is practical to generate a large class of de Bruijn sequences up to order $n \approx 20$. Many previously proposed constructions of de Bruijn sequences are shown to be special cases of our construction.

Keywords Binary periodic sequence · LFSR · de Bruijn sequence · cycle structure · adjacency graph · cyclotomic number

Mathematics Subject Classification (2000) 11B50 · 94A55 · 94A60

1 Introduction

A binary *de Bruijn sequence* of order n has period $N = 2^n$ in which each n-tuple occurs exactly once in each period. There are $2^{2^{n-1}-n}$ of them [5]. Some of their earliest applications are in communication systems. They are generated in a deterministic way, yet satisfy the randomness criteria in [14, Ch. 5] and are balanced, containing the same number of 1s and 0s. In cryptography, they have been used as a source of pseudo-random numbers and in key-sequence generators of stream ciphers [29, Sect. 6.3]. In computational molecular biology, one of the three assembly paradigms in DNA sequencing is the de Bruijn graph assemblers model [31, Box 2]. Some roles of de Bruijn sequences in robust positioning patterns are discussed by Bruckstein *et al.* in [4]. They have numerous applications, *e.g.*, in robotics,

Z. Chang
School of Mathematics and Statistics, Zhengzhou University, Zhengzhou 450001, China
E-mail: zuling_chang@zzu.edu.cn

M. F. Ezerman · S. Ling · H. Wang
Division of Mathematical Sciences, School of Physical and Mathematical Sciences,
Nanyang Technological University, 21 Nanyang Link, Singapore 637371
E-mail: {fredezerman,lingsan,HXWang}@ntu.edu.sg

smart pens, and camera localization. There has also been an increased interest in deploying de Bruijn sequences in spread spectrum [33].

On their construction, two yardsticks are often used to measure the goodness of a method or algorithm, namely, the number of constructed sequences and the efficiency of the construction in terms of both time and memory requirements. In some applications, it is crucial to have a lot of sequences to choose from. Methods to generate all binary de Bruijn sequences have been available in the literature (see, *e.g.*, [12] and [32]). These methods, however, require a large memory space and or long running time.

Fredricksen's survey [12] recalls various properties and construction methods up to the early 1980s. A well-known construction called the *cycle joining (CJ) method* begins with a given Feedback Shift Register (FSR) and joins all cycles produced by the FSR into a single cycle by identifying the conjugate pairs shared by any pair of cycles. The cycle structure of a Linear FSR (LFSR) can be studied using tools from the algebra of polynomial rings. It is then natural to construct de Bruijn sequences by applying the cycle joining method to LFSRs. Some LFSRs with simple cycle structure, such as the maximal length LFSRs, pure cycling registers, and pure summing registers, have been studied in [10,11,12]. Hauge and Helleseth established a connection between the cycles generated by LFSRs and irreducible cyclic codes in [15]. The number of de Bruijn sequences obtained from these LFSRs is related to cyclotomic numbers, which in general are hard to determine precisely.

Recently, C. Li *et al.* studied some classes of de Bruijn sequences. In [22] and [23], respectively, the characteristic polynomials of the LFSRs are $(1+x)^3 p(x)$ and $(1+x^3)p(x)$, where $p(x)$ is a primitive polynomial of degree $n > 2$. Further generalized results are given in [24] to include products of primitive polynomials whose degrees are pairwise coprime, leading to coprime periods of the sequences that form the cycle structure. This generalization yields a relatively small number of de Bruijn sequences when compared to the one we are proposing. M. Li and D. Lin discussed the cycle structure of LFSRs with characteristic polynomial $f(x) = \prod_{k=1}^{s} \ell_k(x)$ where $\gcd(\ell_i(x), \ell_j(x)) = 1$ for $1 \le i \ne j \le s$ and presented some results about the adjacency graph of $\Omega(f(x))$ in more recent work [27]. Each factor $\ell_k(x)$ of $f(x)$ is not necessarily irreducible.

We put forward a construction from LFSRs whose characteristic polynomials are products of two distinct irreducible polynomials and showed that it generates a large number of de Bruijn sequences in [7]. In another work [6], whose preliminary results were presented at SETA 2016, we discussed in detail how to determine the cycle structure and find a state for each cycle for an arbitrary polynomial $f(x) \in \mathbb{F}_q[x]$ for any prime power q. Drawing insights from them, this present work generalizes the construction of de Bruijn sequences from LFSRs with arbitrary polynomials as their characteristic polynomials. The main contributions are as follows.

1. We propose a construction of de Bruijn sequences by the cycle joining method from LFSRs with an arbitrary characteristic polynomial in $\mathbb{F}_2[x]$. The cycle structure and adjacency graph are studied in details. This gives us a fast method to find all conjugate pairs shared by any two cycles. Our construction covers numerous previously-studied constructions of de Bruijn sequences as special cases. While our approach works for the general case of any arbitrary characteristic polynomial, a careful investigation on LFSRs with repeated roots shows that it is not advisable to construct de Bruijn sequences from them. Unless one needs a lot more de Bruijn sequences than those that can be built from LFSRs whose characteristic polynomials are products of distinct irreducible polynomials. In such a situation one must have access to computing resources beyond what is practical for most users.

2. In generalizing the number of irreducible polynomial factors of $f(x)$ from 2 in [7] to $s \geq 2$ we introduce some modifications to stay efficient. Most notably, for an irreducible $g(x)$, a state in $\Omega(g(x))$ is computed based on the mapping φ in [7, Sect. 3], requiring computations over $\mathbb{F}_{2^n}$. This present work determines the state by a simple decimation.
3. Based on our theoretical results, a `python` implementation is written to generate de Bruijn sequences. Here we aim for completeness instead of speed. All conjugate pairs are found and the *full* adjacency graph G is built. Our method is practical for up to $n \approx 20$. Further optimization tricks are possible, depending on a user's particular requirements and available resources.

After this introduction come preliminary notions and known results in Section 2. Section 3 discusses the cycle structure and how to find a state belonging to a given cycle. Section 4 establishes important properties of the conjugate pairs. Two preparatory algorithms are explained in Section 5. These are then used to design the main algorithm that finds all conjugate pairs between any pair of cycles in Section 6. Section 7 discusses the complication of having a characteristic polynomial with repeated roots and highlights some parts of our method that can still be useful in this situation. Section 8 shows how the tools fit together nicely, using the examples presented in the preceding sections to derive a large number of de Bruijn sequences of order 7. A summary of our implementation is given in Section 9. The last section contains a brief conclusion, a table listing prior constructions which can be seen as special cases of ours, and possible future directions.

2 Preliminaries

For convenience, we recall needed definitions and results, mostly from [14, Ch. 4].

An *n-stage shift register* is a circuit consisting of n consecutive storage units, each containing a bit, regulated by a clock. When it pulses, the bit in each storage unit is shifted to the next stage in line. A shift register becomes a binary code generator when one adds a feedback loop which outputs a new bit s_n based on the n bits $\mathbf{s}_0 = (s_0, \ldots, s_{n-1})$ called an *initial state* of the register. The corresponding *feedback function* $h(x_0, \ldots, x_{n-1})$ is the Boolean function that outputs s_n on input $\mathbf{s}_0$. A feedback shift register (FSR) outputs a binary sequence $\mathbf{s} = s_0, s_1, \ldots, s_n, \ldots$ satisfying $s_{n+\ell} = h(s_\ell, s_{\ell+1}, \ldots, s_{\ell+n-1})$ for $\ell = 0, 1, 2, \ldots$. For $N \in \mathbb{N}$, if $s_{i+N} = s_i$ for all $i \geq 0$, then $\mathbf{s}$ is *N-periodic* or *with period N* and one writes $\mathbf{s} = (s_0, s_1, s_2, \ldots, s_{N-1})$. The period of the all zero sequence $\mathbf{0}$ is 1. When the context is clear, $\mathbf{0}$ also denotes a string of zeroes or a zero vector. We call $\mathbf{s}_i = (s_i, s_{i+1}, \ldots, s_{i+n-1})$ *the i-th state* of $\mathbf{s}$ and states $\mathbf{s}_{i-1}$ and $\mathbf{s}_{i+1}$ the *predecessor* and *successor* of $\mathbf{s}_i$, respectively. Given $\mathbf{a} = (a_0, a_1, \ldots, a_{N-1})$ and $\mathbf{b} = (b_0, b_1, \ldots, b_{N-1}) \in \mathbb{F}_2^N$, let $c\mathbf{a} := (ca_0, ca_1, \ldots, ca_{N-1})$ for $c \in \mathbb{F}_2$ and $\mathbf{a} + \mathbf{b} := (a_0 + b_0, a_1 + b_1, \ldots, a_{N-1} + b_{N-1})$.

In an FSR, distinct initial states generate distinct sequences, forming the set $\Omega(h)$ of cardinality 2^n. All sequences in $\Omega(h)$ are periodic if and only if the feedback function h is *nonsingular*, *i.e.*, h can be written as $h(x_0, x_1, \ldots, x_{n-1}) = x_0 + g(x_1, \ldots, x_{n-1})$, where $g(x_1, \ldots, x_{n-1})$ is some Boolean function with domain $\mathbb{F}_2^{n-1}$ [13, p. 116]. In this paper, the feedback functions are all nonsingular. An FSR is called *linear* or an LFSR if its feedback function is linear, and *nonlinear* or an NLFSR otherwise.

The *characteristic polynomial* of an n-stage LFSR with feedback $h(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} c_i x_i$ is $f(x) = x^n + \sum_{i=0}^{n-1} c_i x^i \in \mathbb{F}_2[x]$. A sequence $\mathbf{s}$ may have many characteristic polynomials. The one with the lowest degree is the *minimal polynomial* of $\mathbf{s}$. It represents the LFSR of shortest length that generates $\mathbf{s}$. Given an LFSR with characteristic polynomial $f(x)$, the set $\Omega(h)$ is also denoted by $\Omega(f(x))$. A sequence $\mathbf{v}$ is said to be

a *d-decimation* sequence of $\mathbf{s}$, denoted by $\mathbf{v} = \mathbf{s}^{(d)}$, if $v_j = s_{d \cdot j}$ for all $j \geq 0$. For a sequence $\mathbf{s}$, the *(left) shift operator* L is given by $L\mathbf{s} = L(s_0, s_1, \ldots, s_{N-1}) = (s_1, s_2, \ldots, s_{N-1}, s_0)$ with the convention that $L^0\mathbf{s} = \mathbf{s}$. The set $[\mathbf{s}] := \{\mathbf{s}, L\mathbf{s}, L^2\mathbf{s}, \ldots, L^{N-1}\mathbf{s}\}$ is a *shift equivalent class* or a *cycle*. The set $\Omega(f(x))$ can be partitioned into cycles. If $\Omega(f(x))$ consists of exactly r cycles $[\mathbf{s}_1], [\mathbf{s}_2], \ldots, [\mathbf{s}_r]$ for some $r \in \mathbb{N}$, then its *cycle structure* is $\Omega(f(x)) = [\mathbf{s}_1] \cup [\mathbf{s}_2] \cup \ldots \cup [\mathbf{s}_r]$. A *conjugate pair* consists of a state $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ and its *conjugate* $\widehat{\mathbf{v}} = (v_0 + 1, v_1, \ldots, v_{n-1})$. Cycles C_1 and C_2 are *adjacent* if they are disjoint and there exists $\mathbf{v}$ in C_1 whose conjugate $\widehat{\mathbf{v}}$ is in C_2. Adjacent cycles with the same feedback $h(x_0, x_1, \ldots, x_{n-1})$ can be joined into a single cycle by interchanging the successors of $\mathbf{v}$ and $\widehat{\mathbf{v}}$. The resulting cycle has feedback function

$$g(x_0, \ldots, x_{n-1}) = h(x_0, \ldots, x_{n-1}) + \prod_{i=1}^{n-1} (x_i + v_i + 1). \tag{1}$$

The feedback functions of the resulting de Bruijn sequences are completely determined once the corresponding conjugate pairs are found. It is therefore crucial to find all pairs and to place them in the adjacency graph.

Definition 1 [16] The *adjacency graph* G for an FSR with feedback function h is an undirected multigraph whose vertices correspond to the cycles in $\Omega(h)$. There exists an edge between two vertices if and only if they are adjacent. A conjugate pair labels every edge. The number of edges between any pair of cycles is the number of shared conjugate pairs.

By definition G contains no loops. There is a one-to-one correspondence between the spanning trees of G and the de Bruijn sequences constructed by the cycle joining method [15, 16]. The following result, a variant of the BEST (de **B**ruijn, **E**hrenfest, **S**mith, and **T**utte) Theorem adapted from [1, Sect. 7], provides the counting formula.

Theorem 1 *(BEST) Let G be the adjacency graph of an FSR with vertex set $\{V_1, V_2, \ldots, V_\ell\}$. Let $\mathscr{M} = (m_{i,j})$ be the $\ell \times \ell$ matrix derived from G in which $m_{i,i}$ is the number of edges incident to vertex V_i and $m_{i,j}$ is the negative of the number of edges between vertices V_i and V_j for $i \neq j$. Then the number of the spanning trees of G is the cofactor of any entry of $\mathscr{M}$.*

The *cofactor* of entry $m_{i,j}$ in $\mathscr{M}$ is $(-1)^{i+j}$ times the determinant of the matrix obtained by deleting the i-th row and j-th column of $\mathscr{M}$. A tool we will need later is the generalized Chinese Remainder Theorem (CRT).

Theorem 2 *(Generalized CRT) [8, Thm. 2.4.2]*
Let $2 \leq k \in \mathbb{N}$. Given integers $a_1, \ldots, a_k$ and positive integers $m_1, \ldots, m_k$, there exists $\ell \in \mathbb{N}$ such that $\ell \equiv a_i \pmod{m_i}$ for all $i \in \{1, \ldots, k\}$ if and only if for arbitrary distinct integers $1 \leq i \neq j \leq k$, we have $a_i \equiv a_j \pmod{\gcd(m_i, m_j)}$.

If ℓ is a solution of this system of congruences, then ℓ' is also a solution if and only if $\ell' \equiv \ell \pmod{\operatorname{lcm}(m_1, \ldots, m_k)}$.

Let $\{p_1(x), p_2(x), \ldots, p_s(x)\}$ be a set of s pairwise distinct irreducible polynomials over $\mathbb{F}_2$ and $n_i := \deg(p_i(x))$. From hereon, let

$$f(x) := \prod_{i=1}^{s} p_i(x) \text{ and } n := \sum_{i=1}^{s} n_i.$$

3 The Cycle Structure and a State in Each Cycle

Results needed to determine the cycle structure and a state in each cycle have recently been established in [6]. To make this work self-contained we briefly reproduce the relevant parts in two parts. The first recalls results on the cycle structure of $\Omega(f(x))$. The second determines a state belonging to each of the cycles in $\Omega(f(x))$.

3.1 The Cycle Structure of $\Omega(f(x))$

Let $g(x) = x^n + b_{n-1}x^{n-1} + \ldots + b_0 \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n with a root $\beta \in \mathbb{F}_{2^n}$. Then there exists a primitive element $\alpha \in \mathbb{F}_{2^n}$ such that $\beta = \alpha^t$ for some $t \in \mathbb{N}$ and $e = \frac{2^n-1}{t}$ is the order of β. Using the *Zech logarithmic representation* (see. *e.g.*, [14, p. 39]), write $1 + \alpha^\ell = \alpha^{\tau_n(\ell)}$ where $\tau_n(\ell)$ is the Zech logarithm relative to α. It induces a permutation on $\{1, 2, \ldots, 2^n - 2\}$. Note that $\tau_n(\ell) := \infty$ for $\ell \equiv 0 \pmod{2^n - 1}$ and $\alpha^\infty := 0$.

The *cyclotomic classes* $\mathscr{C}_i \subseteq \mathbb{F}_{2^n}$, for $0 \le i < t$, are

$$\mathscr{C}_i = \{\alpha^{i+s\cdot t} \mid 0 \le s < e\} = \{\alpha^i \beta^s \mid 0 \le s < e\} = \alpha^i \mathscr{C}_0. \tag{2}$$

The *cyclotomic numbers* $(i, j)_t$, for $0 \le i, j < t$, are

$$(i, j)_t = \left|\{\xi \mid \xi \in \mathscr{C}_i, \xi + 1 \in \mathscr{C}_j\}\right|. \tag{3}$$

We know from [14, Ch. 4] that

$$\Omega(g(x)) = [\mathbf{0}] \cup [\mathbf{u}_0] \cup [\mathbf{u}_1] \cup \ldots \cup [\mathbf{u}_{t-1}] \tag{4}$$

with $[\mathbf{u}_i]$ having period e. A way to construct $\Omega(g(x))$ with the property that the cyclotomic classes and the cycles are in a one-to-one correspondence can be found in [15, Thm. 3]. We have proved the following result in [7] by using the properties of cyclotomic numbers.

Lemma 1 *Let $g(x) \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n and order e (making $t = \frac{2^n-1}{e}$) with $\Omega(g(x))$ as presented in (4). Then, for each triple (i, j, k) with $0 \le i, j, k < t$,*

$$(j - i, k - i)_t = \left|\{a \mid \mathbf{u}_i + L^a \mathbf{u}_j = L^b \mathbf{u}_k; 0 \le a, b < e\}\right|. \tag{5}$$

Given $f(x)$, since $p_i(x)$ is irreducible of degree n_i and order e_i, we have $t_i := \frac{2^{n_i}-1}{e_i}$ and

$$\Omega(p_i(x)) = [\mathbf{0}] \cup \left[\mathbf{s}^i_0\right] \cup \left[\mathbf{s}^i_1\right] \cup \ldots \cup \left[\mathbf{s}^i_{t_i-1}\right]. \tag{6}$$

The cycle structure of $\Omega(f(x))$ can be derived from [6, Thm. 1] by restricting q to 2. A similar result was established using some properties of LFSRs in [27].

Lemma 2 *Let $f_i := \begin{cases} e_i & \text{if } a_i = 1 \\ 1 & \text{if } a_i = 0 \end{cases}$ and $\delta := \gcd(f_s, \mathrm{lcm}(f_1, \ldots, f_{s-1}))$. Then*

$$\Omega(f(x)) = \bigcup_{\substack{a_i \in \mathbb{F}_2 \\ 1 \le i \le s}} \bigcup_{j_1=0}^{t_1-1} \cdots \bigcup_{j_s=0}^{t_s-1} \bigcup_{\ell_2=0}^{\gcd(f_2,f_1)-1} \cdots \bigcup_{\ell_s=0}^{\delta-1} \left[a_1 \mathbf{s}^1_{j_1} + a_2 L^{\ell_2} \mathbf{s}^2_{j_2} + \cdots + a_s L^{\ell_s} \mathbf{s}^s_{j_s}\right]. \tag{7}$$

3.2 Finding a State belonging to Each Cycle

Once the number of cycles in $\Omega(f(x))$ is determined, we want to efficiently store them. Recall the state $\mathbf{s}_i$ and its successor $\mathbf{s}_{i+1}$ of an n-stage FSR sequence $\mathbf{s}$ with feedback function $h(x_0,\ldots,x_{n-1})$ from Section 2. A *state operator* T_n turns $\mathbf{s}_i$ into $\mathbf{s}_{i+1}$ with $s_{i+n} = h(s_i,\ldots,s_{i+n-1})$. The subscript n of T_n indicates that $\mathbf{s}_i$ is an n-stage state. If $\mathbf{s}_i \in [\mathbf{s}]$ and e is the period of $\mathbf{s}$, then the e distinct states of $[\mathbf{s}]$ are $\mathbf{s}_i, T\mathbf{s}_i = \mathbf{s}_{i+1},\ldots,T^{e-1}\mathbf{s}_i = \mathbf{s}_{i+e-1}$. It suffices to identify just one state in a given cycle since applying T a suitable number of times generates all e distinct states. To reduce clutters, we use T to denote the state operator for distinct cycles with distinct stages.

Settling this matter is related to [27, Problem 2]. Deciding if two distinct nonzero states of length n belong to the same cycle is hard to determine if the characteristic polynomial is irreducible but non-primitive. We transform the problem into finding an associated primitive polynomial and use decimation to solve the initial problem.

Let $g(x) \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n, order e, and β as a root. Hence, $t = \frac{2^n-1}{e}$ and $\Omega(g(x))$ is as given in (4). To find a state for each nonzero cycle in $\Omega(g(x))$ we start by searching for a primitive polynomial $q(x)$ of degree n with a root α satisfying $\beta = \alpha^t$. We call $q(x)$ an *associated primitive polynomial* of $g(x)$. We know from [6, Lemma 1] that there are $\dfrac{\phi(2^n-1)}{\phi(e)}$ primitive polynomials that can be associated with $g(x)$ and any one of them can be used here. We use LFSR with characteristic polynomial $q(x)$ and a nonzero initial state to generate an m-sequence $\mathbf{m}$. From $\mathbf{m}$, we construct the t distinct t-decimation sequences, each of period e:

$$\mathbf{u}_0 = \mathbf{m}^{(t)}, \mathbf{u}_1 = (L\mathbf{m})^{(t)},\ldots,\mathbf{u}_{t-1} = (L^{t-1}\mathbf{m})^{(t)}. \tag{8}$$

These t distinct sequences are in $\Omega(g(x))$. Starting from an arbitrary $n \cdot t$ consecutive elements of $\mathbf{m}$, one gets t distinct n-stage states by decimation. It is then straightforward to verify that each of the derived states corresponds to one nonzero cycle.

To apply Lemma 1 later, the correspondence between the cycles and the cyclotomic classes is required. The details can be found in [15]. Hence, one must ensure that $(1,\mathbf{0})$ is the initial state of $[\mathbf{u}_0]$, corresponding to $\mathscr{C}_0$. The next proposition shows how to guarantee the correspondence and Algorithm 1 gives the steps to generate the states. If such correspondence is not necessary, then any nonzero vector in $\mathbb{F}_2^n$ can be used as $\mathbf{s}_0$ in Algorithm 1.

Proposition 1 *Let a non-primitive irreducible polynomial $g(x)$ and its associated primitive polynomial $q(x)$ be given. Then there exists an initial state $\mathbf{s}_0$ such that $q(x)$ generates an m-sequence $\mathbf{m}$ with $(1,\mathbf{0}) \in \mathbb{F}_2^n$ as the first n entries in $\mathbf{m}^{(t)}$.*

Proof Using the definition of characteristic polynomial, $\mathbf{s}_0$ can be computed by solving a system of n linear equations. Write $q(x) = x^n + a_{n-1}x^{n-1} + \ldots + a_1x + 1$ and let A be its companion matrix: the $n \times n$ matrix whose first row and last column are respectively $(0,\ldots,0,1)$ and $(1,a_1,a_2,\ldots,a_{n-1})^\intercal$. The remaining entries form the identity matrix I_{n-1}. Then the respective first entry of the state vectors $\mathbf{s}_0, \mathbf{s}_0A^t, \mathbf{s}_0A^{2t},\ldots,\mathbf{s}_0A^{(n-1)t}$ must be $1,0,0,\ldots,0$. Solving the system gives us $\mathbf{s}_0$. □

Example 1 Consider $g(x) = x^4 + x^3 + x^2 + x + 1$, an irreducible polynomial of order 5, with $p(x) = x^4 + x + 1$ as the associated primitive polynomial. The m-sequence that $p(x)$ generates on input $(1,0,0,0)$ is $\mathbf{m} = (1,0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,0,0,0,1,0,0,1,1,0)$. Computing $(L^j\mathbf{m})^{(3)}$ with $j \in \{0,1,2\}$ gives us $[\mathbf{u}_j]$. We take as the respective initial states the first 4 entries of $(L^j\mathbf{m})^{(3)}$: $(1,0,0,0)$, $(0,1,1,1)$, and $(0,0,1,0)$.

Algorithm 1 Finding a State in a Nonzero Cycle in $\Omega(g(x))$

Input: An irreducible polynomial $g(x) \in \mathbb{F}_2[x]$.
Output: A state $\mathbf{v}_j$ of each nonzero cycle $[\mathbf{u}_j] \in \Omega(g(x))$.
1: $e \leftarrow$ order of $g(x)$; $t \leftarrow (2^n-1)/e$.
2: **if** $t = 1$ **then**
3: $\quad \mathbf{v}_0 \leftarrow (1,0,\ldots,0) \in \mathbb{F}_2^n$ and break.
4: **else**
5: $\quad q(x) \leftarrow$ an associated primitive polynomial of $g(x)$.
6: $\quad \mathbf{w} \leftarrow$ the first $n \cdot t$ consecutive entries of the m-sequence generated by $q(x)$ on input $\mathbf{s}_0 \in \mathbb{F}_2^n$.
7: $\quad$ **for** j from 0 to $t-1$ **do**
8: $\quad\quad \mathbf{v}_j \leftarrow (L^j \mathbf{w})^{(t)}$.
9: $\quad$ **end for**
10: **end if**

Based on known states of the cycles in $\Omega(p_i(x))$, we determine a state in each of the cycles in $\Omega(f(x))$. For each $1 \le i \le s$, construct the $n_i \times n$ matrix $\mathscr{P}_i$ in the following manner. The j-th row of $\mathscr{P}_i$ is the first n bits of the sequence generated by the LFSR with characteristic polynomial $p_i(x)$ whose n_i-stage initial state has 1 in the j-th position and 0 elsewhere. We combine the resulting matrices into the full-rank (see [6, Lemma 7]) matrix

$$\mathscr{P}_{n\times n} = \begin{pmatrix} \mathscr{P}_1 \\ \mathscr{P}_2 \\ \vdots \\ \mathscr{P}_s \end{pmatrix}.$$

Let $\mathscr{P}$ be already constructed. Let $\mathbf{v} \in \mathbb{F}_2^n$ and $\mathbf{a}_i \in \mathbb{F}_2^{n_i}$, with $1 \le i \le s$, be respectively the n-stage and n_i-stage states of the sequences in $\Omega(f(x))$ and $\Omega(p_i(x))$. There is a bijection between $\mathbf{v}$ and $(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_s)$ via $\mathbf{v} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_s)\mathscr{P}$. Note that $\mathscr{P}$ and T commute since $T\mathbf{v} = T[(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_s)\mathscr{P}] = (T\mathbf{a}_1, T\mathbf{a}_2, \ldots, T\mathbf{a}_s)\mathscr{P}$. Hence, any sequence $\mathbf{s} \in \Omega(f(x))$ with initial state $\mathbf{v}$ can be written as the sum of sequences $\mathbf{s}_i$ from $\Omega(p_i(x))$ with corresponding initial states $\mathbf{a}_i$ for all i. We can conveniently use $(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_s)$ to represent the state $\mathbf{v}$. A sequence $\mathbf{s}$ generated by $f(x)$ has a longer period and is more complex to study than the *component sequences* $\mathbf{s}_i$. Representing the state $\mathbf{v}$ of $\mathbf{s}$ in terms of states $\mathbf{a}_i$ of corresponding sequences $\mathbf{s}_i$ helps us study the global properties while relying on local properties only. The new state representation offers a significant gain in storage efficiency. In addition, a state belonging to each cycle in $\Omega(f(x))$ can be quickly computed using the representation.

Suppose that we have obtained the set A_i of states corresponding to the $t_i + 1$ distinct cycles in $\Omega(p_i(x))$ given in (6). Letting $\mathbf{0}$ be the state of $[\mathbf{0}]$ and $\mathbf{a}_j^i$ a nonzero state of $[\mathbf{s}_j^i]$, $A_i := \{\mathbf{a}_0^i, \mathbf{a}_1^i, \ldots, \mathbf{a}_{t_i-1}^i, \mathbf{a}_{t_i}^i = \mathbf{0}\}$. For convenience, let each of the states be the initial state of its corresponding sequence. Then one takes

$$\mathbf{v} = \left(\mathbf{a}_{j_1}^1, \mathbf{a}_{j_2}^2, \ldots, \mathbf{a}_{j_s}^s\right)\mathscr{P} \text{ with } \mathbf{a}_{j_i}^i \in A_i \tag{9}$$

as an initial state of a sequence $\mathbf{s} \in \Omega(f(x))$. Note that $\mathbf{s}$ has the form $a_1\mathbf{s}_{j_1}^1 + a_2\mathbf{s}_{j_2}^2 + \ldots + a_s\mathbf{s}_{j_s}^s$, where $a_i\mathbf{s}_{j_i}^i = \mathbf{0}$ if $a_i = 0$. For all other cases, $a_i = 1$. For $1 \le i \le s$, let ℓ_i be a nonnegative integer and let $\mathbf{v}$ be as given in (9). By the properties of $\mathscr{P}$ and T, $\mathbf{w} = \left(T^{\ell_1}\mathbf{a}_{j_1}^1, T^{\ell_2}\mathbf{a}_{j_2}^2, \ldots, T^{\ell_s}\mathbf{a}_{j_s}^s\right)\mathscr{P}$ is a state of cycle $\left[a_1L^{\ell_1}\mathbf{s}_{j_1}^1 + a_2L^{\ell_2}\mathbf{s}_{j_2}^2 + \ldots + a_sL^{\ell_s}\mathbf{s}_{j_s}^s\right] = \left[a_1\mathbf{s}_{j_1}^1 + a_2L^{\ell_2-\ell_1}\mathbf{s}_{j_2}^2 + \ldots + a_sL^{\ell_s-\ell_1}\mathbf{s}_{j_s}^s\right]$. We quickly find a state belonging to any cycle.

Example 2 Let $f(x) = \underbrace{(x+1)}_{=p_1(x)}\underbrace{(x^2+x+1)}_{=p_2(x)}\underbrace{(x^4+x^3+x^2+x+1)}_{=p_3(x)}$. Notice that $p_3(x)$ is not primitive. One gets $\mathscr{P}_1 = [\mathbf{1}]$, $\mathscr{P}_2 = \begin{pmatrix} 1&0&1&1&0&1&1 \\ 0&1&1&0&1&1&0 \end{pmatrix}$, $\mathscr{P}_3 = \begin{pmatrix} 1&0&0&0&1&1&0 \\ 0&1&0&0&1&0&1 \\ 0&0&1&0&1&0&0 \\ 0&0&0&1&1&0&0 \end{pmatrix}$, from which $\mathscr{P}$ and $\mathscr{P}^{-1}$ immediately follow. The relevant cycles and sets of states are

$$\begin{aligned}
\Omega(p_1(x)) =& [\mathbf{0}] \cup [\mathbf{s}_0^1 = \mathbf{1}], A_1 = \{\mathbf{a}_0^1 = (1), \mathbf{a}_1^1 = (0)\}, \\
\Omega(p_2(x)) =& [\mathbf{0}] \cup [\mathbf{s}_0^2 = (1,0,1)], A_2 = \{\mathbf{a}_0^2 = (1,0), \mathbf{a}_1^2 = (0,0)\}, \\
\Omega(p_3(x)) =& [\mathbf{0}] \cup [\mathbf{s}_0^3 = (1,0,0,0,1)] \cup [\mathbf{s}_1^3 = (0,1,1,1,1)] \cup [\mathbf{s}_2^3 = (0,0,1,0,1)], \\
A_3 =& \{\mathbf{a}_0^3 = (1,0,0,0), \mathbf{a}_1^3 = (0,1,1,1), \mathbf{a}_2^3 = (0,0,1,0), \mathbf{a}_3^3 = (0,0,0,0)\}.
\end{aligned}$$

The periods of the nonzero sequences in $\Omega(p_i(x))$ are $1, 3$, and 5. We write each cycle in $\Omega(f(x))$ as $\left[a_1\mathbf{1} + a_2\mathbf{s}_0^2 + a_3\mathbf{s}_j^3\right]$ with $a_i \in \mathbb{F}_2$ and $j \in \{0,1,2\}$. Choosing $\mathbf{a} = \left(\mathbf{a}_0^1, \mathbf{a}_0^2, \mathbf{a}_0^3\right) = (1,1,0,1,0,0,0)$ implies that $\mathbf{v} = \mathbf{a}\mathscr{P} = (1,1,0,0,0,1,0)$ is a state of $\left[\mathbf{1} + \mathbf{s}_0^2 + \mathbf{s}_0^3\right]$. A state of each of the cycles in $\Omega(f(x))$ can be similarly derived. Table 1 lists them down.

Table 1 List of States and Respective Cycles

Apply $\mathscr{P}$ to	State	Cycle	Period
$\left(\mathbf{a}_1^1, \mathbf{a}_1^2, \mathbf{a}_3^3\right)$	$\mathbf{0}$	$[\mathbf{0}]$	1
$\left(\mathbf{a}_1^1, \mathbf{a}_0^2, \mathbf{a}_3^3\right)$	$(1,0,1,1,0,1,1)$	$\left[\mathbf{s}_0^2\right] = [(1,0,1)]$	3
$\left(\mathbf{a}_1^1, \mathbf{a}_1^2, \mathbf{a}_0^3\right)$	$(1,0,0,0,1,1,0)$	$\left[\mathbf{s}_0^3\right] = [(1,0,0,0,1)]$	5
$\left(\mathbf{a}_1^1, \mathbf{a}_1^2, \mathbf{a}_1^3\right)$	$(0,1,1,1,1,0,1)$	$\left[\mathbf{s}_1^3\right] = [(0,1,1,1,1)]$	5
$\left(\mathbf{a}_1^1, \mathbf{a}_1^2, \mathbf{a}_2^3\right)$	$(0,0,1,0,1,0,0)$	$\left[\mathbf{s}_2^3\right] = [(0,0,1,0,1)]$	5
$\left(\mathbf{a}_1^1, \mathbf{a}_0^2, \mathbf{a}_0^3\right)$	$(0,0,1,1,1,0,1)$	$\left[\mathbf{s}_0^2 + \mathbf{s}_0^3\right] = [(0,0,1,1,1,0,1,0,1,0,1,1,1,0,0)]$	15
$\left(\mathbf{a}_1^1, \mathbf{a}_0^2, \mathbf{a}_1^3\right)$	$(1,1,0,0,1,1,0)$	$\left[\mathbf{s}_0^2 + \mathbf{s}_1^3\right] = [(1,1,0,0,1,1,0,1,0,0,0,0,0,1,0)]$	15
$\left(\mathbf{a}_1^1, \mathbf{a}_0^2, \mathbf{a}_2^3\right)$	$(1,0,0,1,1,1,1)$	$\left[\mathbf{s}_0^2 + \mathbf{s}_2^3\right] = [(1,0,0,1,1,1,1,1,1,0,0,1,0,0,0)]$	15
$\left(\mathbf{a}_0^1, \mathbf{a}_1^2, \mathbf{a}_3^3\right)$	$\mathbf{1}$	$[\mathbf{1}]$	1
$\left(\mathbf{a}_0^1, \mathbf{a}_0^2, \mathbf{a}_3^3\right)$	$(0,1,0,0,1,0,0)$	$\left[\mathbf{1} + \mathbf{s}_0^2\right] = [(0,1,0)]$	3
$\left(\mathbf{a}_0^1, \mathbf{a}_1^2, \mathbf{a}_0^3\right)$	$(0,1,1,1,0,0,1)$	$\left[\mathbf{1} + \mathbf{s}_0^3\right] = [(0,1,1,1,0)]$	5
$\left(\mathbf{a}_0^1, \mathbf{a}_1^2, \mathbf{a}_1^3\right)$	$(1,0,0,0,0,1,0)$	$\left[\mathbf{1} + \mathbf{s}_1^3\right] = [(1,0,0,0,0)]$	5
$\left(\mathbf{a}_0^1, \mathbf{a}_1^2, \mathbf{a}_2^3\right)$	$(1,1,0,1,0,1,1)$	$\left[\mathbf{1} + \mathbf{s}_2^3\right] = [(1,1,0,1,0)]$	5
$\left(\mathbf{a}_0^1, \mathbf{a}_0^2, \mathbf{a}_0^3\right)$	$(1,1,0,0,0,1,0)$	$\left[\mathbf{1} + \mathbf{s}_0^2 + \mathbf{s}_0^3\right] = [(1,1,0,0,0,1,0,1,0,1,0,0,0,1,1)]$	15
$\left(\mathbf{a}_0^1, \mathbf{a}_0^2, \mathbf{a}_1^3\right)$	$(0,0,1,1,0,0,1)$	$\left[\mathbf{1} + \mathbf{s}_0^2 + \mathbf{s}_1^3\right] = [(0,0,1,1,0,0,1,0,1,1,1,1,1,0,1)]$	15
$\left(\mathbf{a}_0^1, \mathbf{a}_0^2, \mathbf{a}_2^3\right)$	$(0,1,1,0,0,0,0)$	$\left[\mathbf{1} + \mathbf{s}_0^2 + \mathbf{s}_2^3\right] = [(0,1,1,0,0,0,0,0,0,1,1,0,1,1,1)]$	15

Example 2 demonstrates how our approach quickly finds a state belonging to any cycle in $\Omega(f(x))$. This works in general, not only when the periods are coprime, as will be shown in Section 7 below.

4 Properties of the Adjacency Graph of $\Omega(f(x))$

To use Lemma 1, we assume that each nonzero cycle $[\mathbf{s}_j^i] \in \Omega(p_i(x))$ in (6) corresponds to the suitable cyclotomic class. No generality is loss here. If the two do not correspond, the resulting adjacency graph would be permutation equivalent to the one we obtain.

Let $C_1 = [\mathbf{s}_1]$ and $C_2 = [\mathbf{s}_2]$ be two cycles in $\Omega(f(x))$ and let the special state $\mathbf{S} := (1, \mathbf{0}) \in \mathbb{F}_2^n$ belong to $[\mathbf{s}_0] \in \Omega(f(x))$. If $(\mathbf{v}, \widehat{\mathbf{v}})$ is a conjugate pair between C_1 and C_2, then $\widehat{\mathbf{v}} + \mathbf{v} = \mathbf{S}$. Thus, C_1 and C_2 share at least a conjugate pair if and only if some shift of $\mathbf{s}_1$ plus some shift of $\mathbf{s}_2$ is equal to $\mathbf{s}_0$, *i.e.*, there exist integers ℓ and ℓ' satisfying $L^\ell \mathbf{s}_1 + L^{\ell'} \mathbf{s}_2 = \mathbf{s}_0$. Since $\mathbf{S}$ has $n-1$ consecutive 0s, $f(x)$ is the minimal polynomial of $[\mathbf{s}_0]$. Hence, $[\mathbf{s}_0]$ can be determined with the help of $\mathscr{P}^{-1}$ and Equation (9). Without loss of generality, let

$$[\mathbf{s}_0] = \left[L^{c_1}\mathbf{s}_{d_1}^1 + L^{c_2}\mathbf{s}_{d_2}^2 + \ldots + L^{c_s}\mathbf{s}_{d_s}^s\right] \tag{10}$$

where $c_1, c_2, \ldots, c_s$ and $d_1, d_2, \ldots, d_s$ are some suitable integers. In fact, if the initial states are appropriately chosen, one can make $c_k = d_k = 0$ for $1 \le k \le s$. For our purposes, doing so is not required.

It is clear that $[\mathbf{0}]$ and $[\mathbf{s}_0]$ share a unique conjugate pair. To compute the exact number of conjugate pairs between C_1 and C_2, knowing important properties of the cycles in $\Omega(p_i(x))$ is crucial. Suppose that we have

$$\Gamma_i(j_1, j_2) := \{(u, v) | L^u \mathbf{s}_{j_1}^i + L^v \mathbf{s}_{j_2}^i = \mathbf{s}_{d_i}^i,\ 0 \le u, v < e_i\}. \tag{11}$$

If $\mathbf{s}_{j_1}^i = \mathbf{0}$, then $u = 0$. If $\mathbf{s}_{j_2}^i = \mathbf{0}$, then $v = 0$. If only one of $\mathbf{s}_{j_1}^i$ and $\mathbf{s}_{j_2}^i$ is $\mathbf{0}$, then the other must be $\mathbf{s}_{d_i}^i$ and $\Gamma_i(j_1, j_2) = \{(0,0)\}$. If both $\mathbf{s}_{j_1}^i$ and $\mathbf{s}_{j_2}^i$ are $\neq \mathbf{0}$, then $|\Gamma_i(j_1, j_2)| = (j_1 - d_i, j_2 - d_i)_{t_i} = (j_2 - d_i, j_1 - d_i)_{t_i}$ defined in Lemma 1. Following Equation (7), we let

$$C_1 = [\mathbf{s}_1] = \left[a_1 L^{\ell_1}\mathbf{s}_{j_1}^1 + \ldots + a_s L^{\ell_s}\mathbf{s}_{j_s}^s\right] \text{ and } C_2 = [\mathbf{s}_2] = \left[a_1' L^{\ell_1'}\mathbf{s}_{j_1'}^1 + \ldots + a_s' L^{\ell_s'}\mathbf{s}_{j_s'}^s\right]. \tag{12}$$

Both $\mathbf{s}_1$ and $\mathbf{s}_2$ have period $\mathrm{lcm}(f_1, \ldots, f_s)$. Let $E_1 := \{i \mid a_i = 1, 1 \le i \le s\}$ and $E_2 := \{j \mid a_j' = 1, 1 \le j \le s\}$.

Theorem 3 *Take C_1 and C_2 from (12) and $\Gamma_i(j_i, j_i')$ in (11).*

1. *The following conditions are necessary for C_1 and C_2 to share at least a conjugate pair.*
   (a) *$E_1 \cup E_2 = \{1, 2, \ldots, s\}$.*
   (b) *If $i \in E_1 \cap E_2^{\mathsf{c}}$, then $\mathbf{s}_{j_i}^i = \mathbf{s}_{d_i}^i$. Similarly, if $i \in E_1{}^{\mathsf{c}} \cap E_2$, then $\mathbf{s}_{j_i'}^i = \mathbf{s}_{d_i}^i$.*
   (c) *If $i \in E_1 \cap E_2$, then $(j_i - d_i, j_i' - d_i)_{t_i} > 0$.*
2. *The number of conjugate pairs shared by cycles C_1 and C_2 is equal to the number of tuples $((u_1, v_1), \ldots, (u_s, v_s))$ that satisfy two requirements.*
   (a) *$(u_i, v_i) \in \Gamma_i(j_i, j_i')$ for $1 \le i \le s$.*
   (b) *The following systems of congruences hold modulo $\gcd(e_i, e_j)$.*

$$c_i + u_i - \ell_i \equiv c_j + u_j - \ell_j \text{ for all } i \neq j \in E_1,$$
$$c_i + v_i - \ell_i' \equiv c_j + v_j - \ell_j' \text{ for all } i \neq j \in E_2.$$

3. *The sum of the numbers of conjugate pairs between any two cycles over all possible values for $\ell_1, \ldots, \ell_s$ and $\ell_1', \ldots, \ell_s'$ is equal to $\prod_{i \in E_1 \cap E_2} (j_i - d_i, j_i' - d_i)_{t_i}$.*

   As the ℓs and ℓ's range through all of their respective values, it may happen that $C_1 = C_2$. When this is the case, we count the conjugate pairs $(\mathbf{v}, \widehat{\mathbf{v}})$ and $(\widehat{\mathbf{v}}, \mathbf{v})$ separately even though they are the same.

Proof If C_1 and C_2 share a conjugate pair, then there exist integers ℓ and ℓ' satisfying $L^{\ell}\mathbf{s}_1 + L^{\ell'}\mathbf{s}_2 = \mathbf{s}_0$. By definitions,

$$L^{\ell}\mathbf{s}_1 + L^{\ell'}\mathbf{s}_2 = \sum_{i=1}^{s} a_i L^{\ell+\ell_i}\mathbf{s}^i_{j_i} + a'_i L^{\ell'+\ell'_i}\mathbf{s}^i_{j'_i} = \sum_{i=1}^{s} L^{c_i}\left[a_i L^{\ell+\ell_i-c_i}\mathbf{s}^i_{j_i} + a'_i L^{\ell'+\ell'_i-c_i}\mathbf{s}^i_{j'_i}\right] = \sum_{i=1}^{s} L^{c_i}\mathbf{s}^i_{d_i}.$$

One gets $a_i L^{\ell+\ell_i-c_i}\mathbf{s}^i_{j_i} + a'_i L^{\ell'+\ell'_i-c_i}\mathbf{s}^i_{j'_i} = \mathbf{s}^i_{d_i}$ since sequences $\mathbf{s}^i_{d_i}$ correspond to distinct irreducible characteristic polynomials. Therefore, $a_i \vee a'_i = 1$, $(\ell+\ell_i-c_i, \ell'+\ell'_i-c_i) \in \Gamma_i(j_i, j'_i)$, and $|\Gamma_i(j_i, j'_i)| > 0$, proving Statement 1.

Let (u_i, v_i) be one of the $(j_i - d_i, j'_i - d_i)_{t_i}$ tuples in $\Gamma_i(j_i, j'_i)$. To ensure the existence of ℓ and ℓ', the following systems of congruences must have solutions modulo f_i for all i:

$$\begin{cases} u_i \equiv \ell + \ell_i - c_i \\ v_i \equiv \ell' + \ell'_i - c_i \end{cases} \iff \begin{cases} \ell \equiv c_i + u_i - \ell_i \\ \ell' \equiv c_i + v_i - \ell'_i \end{cases}.$$

By the definition of f_i, with $u_i = v_i = 0$ whenever $i \notin E_1 \cap E_2$, the systems reduce to

$$\begin{cases} \ell \equiv c_i + u_i - \ell_i \pmod{e_i} \text{ for all } i \in E_1 \\ \ell' \equiv c_i + v_i - \ell'_i \pmod{e_i} \text{ for all } i \in E_2 \end{cases}.$$

Theorem 2 says that solutions exist if and only if the following systems of congruences hold modulo $\gcd(e_i, e_j)$:

$$\begin{cases} c_i + u_i - \ell_i \equiv c_j + u_j - \ell_j \text{ for all } i \neq j \in E_1 \\ c_i + v_i - \ell'_i \equiv c_j + v_j - \ell'_j \text{ for all } i \neq j \in E_2 \end{cases}.$$

This completes the proof of Statement 2.

As ℓ_i and ℓ'_i range through their possible values, for a given (u_i, v_i) there exists a corresponding (ℓ, ℓ') satisfying the above systems simultaneously. There are $(j_i - d_i, j'_i - d_i)_{t_i}$ possible choices for (u_i, v_i) if both corresponding sequences are $\neq \mathbf{0}$ but only one choice if one of them is $\mathbf{0}$. Thus, the sum of the numbers of conjugate pairs between C_1 and C_2 over all possible ℓ_is and ℓ'_is is $\prod_{i \in E_1 \cap E_2} (j_i - d_i, j'_i - d_i)_{t_i}$. When $C_1 = C_2$, we double count each of their conjugate pairs since both $(\mathbf{v}, \widehat{\mathbf{v}})$ and $(\widehat{\mathbf{v}}, \mathbf{v})$ appear and are counted separately, despite being exactly the same. Statement 3 is established. □

Remark 1 A similar result to Theorem 3 was given in [27, Thm. 4]. We provide the proof above using our method.

To determine the number of conjugate pairs between two chosen cycles, for each possible (ℓ_i, ℓ'_i) tuple, we check how many $((u_1, v_1), \ldots, (u_s, v_s))$ tuples make the systems solvable. We will provide an efficient procedure to do so. Theorem 3 says that in general it is hard to determine the exact number of conjugate pairs between any two cycles in $\Omega(f(x))$. In some special cases the problem becomes easier. If, for all $1 \leq i \leq s$, the periods of the nonzero sequences in $\Omega(p_i(x))$ are pairwise coprime, then, by Lemma 2, $[\mathbf{s}_0] = \left[\mathbf{s}^1_{d_1} + \mathbf{s}^2_{d_2} + \ldots + \mathbf{s}^s_{d_s}\right]$,

$$C_1 = [\mathbf{s}_1] = \left[a_1\mathbf{s}^1_{j_1} + \ldots + a_s\mathbf{s}^s_{j_s}\right], \text{ and } C_2 = [\mathbf{s}_2] = \left[a'_1\mathbf{s}^1_{j'_1} + \ldots + a'_s\mathbf{s}^s_{j'_s}\right]. \tag{13}$$

We obtain the following corollary to Theorem 3.

Corollary 1 *For all $1 \le i \le s$ let the periods of the nonzero sequences in $\Omega(p_i(x))$ be pairwise coprime. Let C_1 and C_2 be as given in (13). They share at least a conjugate pair if and only if the following requirements are satisfied for all i.*

1. $E_1 \cup E_2 = \{1,2,\ldots,s\}$.
2. *If* $a_i = 1$ *and* $a'_i = 0$, *then* $\mathbf{s}^i_{j_i} = \mathbf{s}^i_{d_i}$.
3. *If* $a_i = 0$ *and* $a'_i = 1$, *then* $\mathbf{s}^i_{j'_i} = \mathbf{s}^i_{d_i}$.
4. *If* $a_i = a'_i = 1$, *then* $(j_i - d_i, j'_i - d_i)_{t_i} > 0$.

Distinct C_1 and C_2 share $\prod_{i \in E_1 \cap E_2} (j_i - d_i, j'_i - d_i)_{t_i}$ *conjugate pairs. We halve the number when* $C_1 = C_2$.

If $p_i(x)$ in Corollary 1 are all primitive, then $t_i = 1$ for all i and there is a conjugate pair between C_1 and C_2 in (13) if and only if $E_1 \cup E_2 = \{1,2,\ldots,s\}$. When $C_1 \neq C_2$, the exact number of conjugate pairs is $\prod_{i \in E_1 \cap E_2} (2^{n_i} - 2)$. When $C_1 = C_2$, we halve the number.

Suppose that we add the assumption that $p_1(x) = x+1$ and keep $p_i(x)$ primitive for $i \ge 2$. If $1 \in E_1 \cap E_2$, *i.e.*, $a_1 = a'_1 = 1$, then there is no conjugate pair between the corresponding pairs of cycles since $n_1 = 1$, making $2^{n_1} - 2 = 0$. Keeping $p_1(x) = x+1$, we prove a result that generalizes [7, Prop. 10].

Proposition 2 *Consider the LFSR with an arbitrary characteristic polynomial $q(x) \in \mathbb{F}_2[x]$. If $(x+1) \mid q(x)$, then $\mathbf{v}$ and $\widehat{\mathbf{v}}$ never belong to the same cycle.*

Proof Let $p_1(x) := (x+1)$ and $q(x) := \prod_{i=1}^{s} p_i^{a_i}(x)$ with $a_i \in \mathbb{N}$. If $[\mathbf{s}] \in \Omega(q(x))$ contains a conjugate pair, then $q(x)$ must be its minimal polynomial. Otherwise, let the minimal polynomial be $p(x)$ with $\deg(p(x)) < \deg(q(x))$. Then there is $k \in \mathbb{Z}$ such that $\mathbf{s} + L^k\mathbf{s}$ contains $\mathbf{S}$ as a state and has characteristic polynomial $p(x)$. The sequence containing $\mathbf{S}$ as a state, however, must have $q(x)$ as its minimal polynomial, which is a contradiction. Hence, $[\mathbf{s}]$ must have the form $\left[\mathbf{s}_1 + L^{i_2}\mathbf{s}_2 + \ldots + L^{i_s}\mathbf{s}_s\right]$ for $i_2,\ldots,i_s \in \mathbb{Z}$ with $p_i^{a_i}(x)$ being the minimal polynomial of $\mathbf{s}_i$ for all i. Thus, $\mathbf{s} + L^k\mathbf{s}$ has the form $L^{i'_1}\mathbf{s}'_1 + L^{i'_2}\mathbf{s}'_s + \ldots + L^{i'_s}\mathbf{s}'_s$ with $p_i^{a_i}(x)$ being the characteristic polynomial of $\mathbf{s}'_i$.

In particular, $L^{i'_1}\mathbf{s}'_1 = \mathbf{s}_1 + L^k\mathbf{s}_1$ must be the sum of two sequences having the same minimal polynomial $(x+1)^{a_1}$. Its period is a power of 2. By [21, Lem. 4.1], the degree of the minimal polynomial of $L^{i'_1}\mathbf{s}'_1$ is $< a_1$. Hence, the degree of the minimal polynomial of the resulting sequence $\mathbf{s} + L^k\mathbf{s}$ must be $< \deg(q(x))$. Thus, it cannot contain $\mathbf{S}$. □

In joining the cycles, the conjugate pairs between any cycle and itself are never used. To take advantage of Proposition 2, let $p_1(x) = x+1$. This implies $\mathbf{s}^1 = \mathbf{1}$. Theorem 3 can still be used to determine the conjugate pairs between any two distinct cycles. Let $\mathbf{s}_0$ be as in (10), $\mathbf{s}_1$ and $\mathbf{s}_2$ in (12), and $\mathbf{s}^1_{d_1} = \mathbf{s}^1_{j_1} = \mathbf{s}^1_{j'_1} = \mathbf{1}$. If $[\mathbf{s}]$ is in $\Omega(f(x))$, then clearly so is $[\mathbf{s}+\mathbf{1}]$.

Corollary 2 *Let $C_1 = [\mathbf{s}_1]$ and $C_2 = [\mathbf{s}_2]$ be two cycles in $\Omega(f(x))$ with $p_1(x) = x+1$.*

1. *For C_1 and C_2 to be adjacent, $a_1 + a'_1 = 1$.*
2. *If $(\mathbf{v}, \widehat{\mathbf{v}})$ is a conjugate pair between $C_1 = [\mathbf{s}_1]$ and $C_2 = [\mathbf{s}_2]$, then $(\mathbf{v}+\mathbf{1}, \widehat{\mathbf{v}}+\mathbf{1})$ is a conjugate pair between $[\mathbf{s}_1+\mathbf{1}]$ and $[\mathbf{s}_2+\mathbf{1}]$.*

These facts simplify the determination of the conjugate pairs. This was exhibited in [24] for $f(x) = (x+1)\prod_{j=2}^{s} p_j(x)$ where the $p_j(x)$s are primitive polynomials with pairwise coprime periods. Corollary 2 tells us that the same applies to a larger class of polynomials.

5 Two Preparatory Algorithms

This section discusses two auxiliary algorithms to prepare for the main algorithm. Any conjugate pair can be written as $(\mathbf{v}, \mathbf{v}+\mathbf{S})$. We have already constructed $\mathscr{P}$ and found a state of each cycle in $\Omega(p_i(x))$ for all $1 \le i \le s$ by Algorithm 1. We now use Algorithm 2 to find the representation

$$\mathbf{S} = \left(T^{c_1}\mathbf{a}^1_{d_1}, T^{c_2}\mathbf{a}^2_{d_2}, \ldots, T^{c_s}\mathbf{a}^s_{d_s}\right)\mathscr{P}. \tag{14}$$

Algorithm 2 Representing the Special State $\mathbf{S}$

Input: $\mathscr{P}$, $\mathbf{S} = (1,0,\ldots,0) \in \mathbb{F}_2^n$.
Output: $((c_1,d_1),\ldots,(c_s,d_s))$ such that (14) holds.

$(\mathbf{v}_1,\ldots,\mathbf{v}_s) \leftarrow \mathbf{S}\mathscr{P}^{-1}$
for i from 1 to s **do**
  for j from 0 to $t_i - 1$ **do**
    for k from 0 to $e_i - 1$ **do**
      if $\mathbf{v}_i = \mathbf{a}^i_j$ **then**
        store $(c_i,d_i) \leftarrow (k,j)$
      else
        $\mathbf{a}^i_j \leftarrow T\mathbf{a}^i_j$
      end if
    end for
  end for
end for
return $((c_1,d_1),\ldots,(c_s,d_s))$

The algorithm is straightforward, using $\mathbf{S}\mathscr{P}^{-1} = (\mathbf{v}_1,\ldots,\mathbf{v}_s)$ to explicitly find a state $\mathbf{v}_i$ for each i belonging to a sequence in $\Omega(p_i(x))$. The tuple (c_i,d_i) satisfying $\mathbf{v}_i = T^{c_i}\mathbf{a}^i_{d_i}$ is found after at most $\sum_{i=1}^{s}(2^{n_i}-1)$ searches. A comparable algorithm accomplishing the same task was presented as [24, Alg. 1]. The latter needs at most $\prod_{i=1}^{s}(2^{n_i}-1)$ searches to find the required representation of $\mathbf{S}$. Algorithm 2 is clearly more efficient.

Algorithm 3 Determining "Conjugate Pairs" between 2 Nonzero Cycles in $\Omega(p_i(x))$

Input: $\mathbf{a}^i_0,\ldots,\mathbf{a}^i_{t_i-1}$, $T^{c_i}\mathbf{a}^i_{d_i}$, e_i.
Output: $\{(\ell_i,-m_i)\}$ satisfying $T^{\ell_i}\mathbf{a}^i_j + T^{-m_i}\mathbf{a}^i_k = T^{c_i}\mathbf{a}^i_{d_i}$.

$K_i \leftarrow \varnothing$ ▷ initiate the list of defining pairs
for j from 0 to $t_i - 2$ **do**
  for k from $j+1$ to $t_i - 1$ **do**
    for $(\ell_i,m_i) \in \{0,1,\ldots,e_i-1\} \times \{0,1,\ldots,e_i-1\}$ **do**
      if $T^{\ell_i}\mathbf{a}^i_j + T^{-m_i}\mathbf{a}^i_k = T^{c_i}\mathbf{a}^i_{d_i}$ **then**
        append $(\ell_i,-m_i)$ to K_i
      end if
    end for
  end for
end for
return K_i and $\lambda_i \triangleq |K_i|$

Next comes a crucial step of dividing the problem of determining "global" conjugate pairs between any two cycles in $\Omega(f(x))$ by first listing all possible "local" candidates between any two cycles in $\Omega(p_i(x))$ for all i. Consider $\Omega(p_i(x))$ for a fixed i and choose any

two cycles in it. Algorithm 3 finds all possible pairs $(\mathbf{w}_1^i, \mathbf{w}_2^i)$ satisfying $\mathbf{w}_1^i + \mathbf{w}_2^i = T^{c_i}\mathbf{a}_{d_i}^i$. Given the representation of $\mathbf{S}$, running this algorithm for all $\Omega(p_i(x))$ yields the required pairs of states in $\Omega(p_i(x))$. The sum of the two states is now the corresponding state in $\mathbf{S}\mathscr{P}^{-1}$. The input consists of t_i states, each corresponding to one of the t_i nonzero cycles in $\Omega(p_i(x))$. The output is the set K_i of all tuples $(\ell_i, -m_i)$ that ensure $T^{\ell_i}\mathbf{a}_j^i + T^{-m_i}\mathbf{a}_k^i = T^{c_i}\mathbf{a}_{d_i}^i$, *i.e.*, the corresponding state pairs sum to $T^{c_i}\mathbf{a}_{d_i}^i$. The specific choice of a state belonging to a cycle affects neither the number of conjugate pairs nor the states being paired in each conjugate pair.

For a chosen pair C_1 and C_2, if Algorithm 3 yields a defining pair (x_1, x_2), then the defining pair for a "conjugate pair" between C_2 and C_1 will be (x_2, x_1). Hence, for each $p_i(x)$, it suffices to take $\binom{t_i}{2}$, instead of t_i^2, distinct (j,k) tuples. The total number λ_i of suitable $(\ell_i, -m_i)$ tuples is a cyclotomic number with specific parameters.

Running the algorithm is unnecessary for $C_1 = [\mathbf{0}]$ and C_2 any nonzero cycle. The pair of cycles $(\mathbf{0}, \mathbf{a}_j^i)$ shares a unique conjugate pair if and only if $\mathbf{a}_j^i = \mathbf{a}_{d_i}^i$, *i.e.*, $K_i = \{(0, c_i)\}$. Similarly, if $p_i(x)$ is primitive, one does not need the algorithm. The desired results can be computed using the Zech logarithm $\tau_n(\ell)$. If $\mathbf{a}$ is an n-stage state of an m-sequence, then the shift-and-add property in [14, Thm. 5.3] says that $\mathbf{a} + T^\ell\mathbf{a} = T^{\tau_n(\ell)}\mathbf{a}$ when $\ell \neq 0$. If, in $(\mathbf{v}_1, \ldots, \mathbf{v}_s) = \mathbf{S}\mathscr{P}^{-1}$, $\mathbf{v}_i = T^c\mathbf{a}$, then an output (y, y') of Algorithm 3 for $p_i(x)$ satisfies $T^y\mathbf{a} + T^{y'}\mathbf{a} = T^c\mathbf{a}$. Hence, $T^{y'}\mathbf{a} = T^c\mathbf{a} + T^y\mathbf{a} = T^c(\mathbf{a} + T^{y-c}\mathbf{a}) = T^{c+\tau_n(y-c)}\mathbf{a}$. Thus, the desired output must be $\{(y, c + \tau_n(y-c)) \mid y \in \{0, 1, \ldots, 2^n - 2\} \setminus \{c\}\}$ and knowing $\tau_n(y)$ is sufficient to find the tuples.

6 The Main Algorithm

The ingredients to construct the adjacency graph G associated with $f(x)$ is now ready. Algorithm 4 uses the results of Algorithm 3 to determine all conjugate pairs between any pair of distinct nonzero cycles. Let us assume that $C_1 = [\mathbf{s}_1] \neq C_2 = [\mathbf{s}_2]$ in $\Omega(f(x))$ are given in the form specified by (12) with the states $\mathbf{v}_1$ of C_1 and $\mathbf{v}_2$ of C_2 written as

$$\mathbf{v}_1 = \left(a_1 T^{\ell_1}\mathbf{a}_{j_1}^1, \ldots, a_s T^{\ell_s}\mathbf{a}_{j_s}^s\right)\mathscr{P} \text{ and } \mathbf{v}_2 = \left(a_1' T^{\ell_1'}\mathbf{a}_{j_1'}^1, \ldots, a_s' T^{\ell_s'}\mathbf{a}_{j_s'}^s\right)\mathscr{P}. \tag{15}$$

Theorem 4 *Algorithm 4 is correct.*

Proof If C_1 and C_2 are adjacent, then there are integers ℓ and ℓ' satisfying $T^\ell\mathbf{v}_1 + T^{\ell'}\mathbf{v}_2 = \mathbf{S}$. In particular, for each i, it holds that $a_i T^{\ell+\ell_i}\mathbf{a}_{j_i}^i + a_i' T^{\ell'+\ell_i'}\mathbf{a}_{j_i'}^i = T^{c_i}\mathbf{a}_{d_i}^i$. Hence, $(\ell + \ell_i, \ell' + \ell_i')$ must be a pair (u_{k_i}, v_{k_i}) obtained from Algorithm 3, *i.e.*, $\ell + \ell_i \equiv u_{k_i} \pmod{f_i}$ and $\ell' + \ell_i' \equiv v_{k_i} \pmod{f_i}$, implying

$$\begin{cases} \ell \equiv u_{k_i} - \ell_i \pmod{e_i} \text{ for all } i \in E_1 \\ \ell' \equiv v_{k_i} - \ell_i' \pmod{e_i} \text{ for all } i \in E_2 \end{cases}.$$

We know that ℓ and ℓ' exist if and only if the systems of congruences

$$\begin{cases} \ell \equiv u_{k_i} - \ell_i \pmod{f_i} \text{ for all } 1 \leq i \leq s \\ \ell' \equiv v_{k_i} - \ell_i' \pmod{f_i} \text{ for all } 1 \leq i \leq s \end{cases}$$

Algorithm 4 All Conjugate Pairs between 2 Nonzero Cycles in $\Omega(f(x))$

Input: $\mathbf{v}_1, \mathbf{v}_2$ defined in (15) and $K_1, K_2, \ldots, K_s$ from Algorithm 3.
Output: All conjugate pairs between distinct nonzero cycles C_1 and C_2.

for i from 1 to s **do**
  if $\lambda_i = 0$ **then**
    return: there is no conjugate pair; break
  end if
end for
$CP \leftarrow \varnothing$ ▷ initiate the set of conjugate pairs
for each $((u_{k_1}, v_{k_1}), \ldots, (u_{k_s}, v_{k_s}))$ in $K \triangleq K_1 \times K_2 \times \ldots \times K_s$ **do** ▷ $1 \le k_i \le \lambda_i$ for $1 \le i \le s$
  if $\begin{cases} u_{k_i} - \ell_i \equiv u_{k_m} - \ell_m \\ v_{k_i} - \ell'_i \equiv v_{k_m} - \ell'_m \end{cases}$ holds in modulo $\gcd(f_i, f_m)$ for all $1 \le m < i \le s$ **then**
    $\mathbf{v} \leftarrow (a_1 T^{u_{k_1}} \mathbf{a}^1_{j_1}, \ldots, a_s T^{u_{k_s}} \mathbf{a}^s_{j_s}) \mathscr{P}$
    append the conjugate pair $(\mathbf{v}, \widehat{\mathbf{v}})$ to CP
  end if
end for
return CP

can be simultaneously solved. For this to happen, Theorem 2 requires that the systems

$$\begin{cases} u_{k_i} - \ell_i \equiv u_{k_j} - \ell_j \pmod{\gcd(f_i, f_j)} \\ v_{k_i} - \ell'_i \equiv v_{k_j} - \ell'_j \pmod{\gcd(f_i, f_j)} \end{cases}$$

hold for $1 \le i \ne j \le s$.

Algorithm 4 checks whether this requirement is met. Given $2 \le i \le s$, the verification is performed for all $1 \le m < i$. All relevant congruences are certified to hold simultaneously. The process is terminated at the first instance when one of the congruences fails to hold. For a chosen set $\{(u_{k_i}, v_{k_i})\}_{i=1}^s$, once the systems of congruences are certified to hold, then ℓ and ℓ' exist, making $(T^\ell \mathbf{v}_1, T^{\ell'} \mathbf{v}_2)$ a conjugate pair. Since $\ell + \ell_i \equiv u_{k_i} \pmod{f_i}$, one confirms that $(\mathbf{v}, \widehat{\mathbf{v}})$ is a conjugate pair with $\mathbf{v} = (a_1 T^{u_{k_1}} \mathbf{a}^1_{j_1}, a_2 T^{u_{k_2}} \mathbf{a}^2_{j_2}, \ldots, a_s T^{u_{k_s}} \mathbf{a}^s_{j_s}) \mathscr{P}$. □

In [24], the $p_i(x)$s are primitive and $e_1, \ldots, e_s$ are pairwise coprime. The conjugate pairs can be found without Algorithm 4. Let $C_1 = [a_1 \mathbf{s}_1 + \ldots + a_s \mathbf{s}_s]$ and $C_2 = [b_1 \mathbf{s}_1 + \ldots + b_s \mathbf{s}_s]$ with $a_i \vee b_i = 1$. Given $\mathbf{S}\mathscr{P}^{-1} = (\mathbf{a}_1, \ldots, \mathbf{a}_s)$, it was shown that $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_s)\mathscr{P}$ in the conjugate pair $(\mathbf{v}, \widehat{\mathbf{v}})$ has

$$\mathbf{v}_i = \begin{cases} \mathbf{0} & \text{if } a_i = 0 \text{ and } b_i = 1, \\ \mathbf{a}_i & \text{if } a_i = 1 \text{ and } b_i = 0, \\ \mathbf{v}_i \in \mathbb{F}_2^{n_i} \setminus \{\mathbf{0}, \mathbf{a}_i\} & \text{if } a_i = b_i = 1. \end{cases}$$

If the periods are not coprime, then the situation is more involved. One must ensure that the required systems of congruences hold simultaneously. Algorithm 4 covers this more general situation. The overall running time can be further improved by using the properties in Section 4 to rule out pairs of cycles with no conjugate pairs prior to running Algorithm 4.

Since Algorithm 4 builds upon the results of Algorithm 3, sufficient storage must be allocated to have them ready at hand. The two algorithms can be merged. In Algorithm 4, at the precise step when the "conjugate pairs" between the specified two cycles in $\Omega(p_i(x))$ are needed, a search for them can be executed to cut down on the storage requirement. Anticipating the need to find all conjugate pairs between any two cycles in $\Omega(f(x))$ in various applications and noting that the data are reasonably small, we prefer storing them.

Algorithm 4 transforms the problem of finding conjugate pairs between two cycles with a more complicated characteristic polynomial into one of finding "conjugate pairs" between two cycles with simpler characteristic polynomials. Once we have determined the "conjugate pairs" for each $p_i(x)$, solving systems of congruences leads us to the desired conjugate pairs for $f(x)$. In contrast, for two cycles with minimal polynomial $f(x)$, exhaustive search requires around $(\mathrm{lcm}(e_1,\ldots,e_s))^2$ computations.

Let ψ denote the number of cycles in $\Omega(f(x))$. The number of pairs of nonzero cycles in Algorithm 4 is $\binom{\psi-1}{2}$. For each of them, the expected cardinality of K is $\prod_{i=1}^{s}\lambda_i = \prod_{i=1}^{s}\left\lceil \frac{e_i}{t_i} \right\rceil$. For any element of K, performing the CRT takes $\mathscr{O}(\log^2(F))$ where $F := \prod_{i=1}^{s} f_i$. Here we use the analysis on the complexity of CRT based on Gardner's Algorithm [19, Sect. 4.3.2]. A similar analysis can also be found in [8, Ch. 3].

7 The Case of Repeated Roots

This section briefly considers characteristic polynomials with repeated roots

$$q(x) := \prod_{i=1}^{s} p_i^{b_i}(x) \text{ with } b_i > 1 \text{ for some } i.$$

We have recently determined the cycle structure of $\Omega(q(x))$ in [6, Thm. 1]. Hence, only a brief outline of the idea is provided here. The number of cycles in $\Omega(p_i^{b_i}(x))$ can be derived from [28, Thm. 8.63].

1. The only cycle with period 1 is $[\mathbf{0}]$.
2. There are t_i cycles containing sequences with period e_i.
3. Let χ_i be the smallest positive integer such that $2^{\chi_i} \geq b_i$ and let $1 \leq j < \chi_i$. Sequences with period $e_i \cdot 2^j$ are partitioned into ρ cycles whenever $b_i > 2$ while those with period $e_i \cdot 2^{\chi_i}$ whenever $b_i \geq 2$ are partitioned into ζ cycles where

$$\rho := \frac{2^{n_i \cdot 2^j} - 2^{n_i \cdot 2^{j-1}}}{e_i \cdot 2^j} \text{ and } \zeta := \frac{2^{n_i \cdot b_i} - 2^{n_i \cdot 2^{\chi_i - 1}}}{e_i \cdot 2^{\chi_i}}.$$

Based on the states of the cycles in $\Omega(p_i(x))$ we give a detailed procedure to derive the states of the remaining cycles in $\Omega(p_i^{b_i}(x))$. For brevity, all states are considered to have the same length $b_i \cdot n_i$. Using the cycles in $\Omega(p_i^{b_i}(x))$ for all i, we can determine all cycles in $\Omega(q(x))$ and prove results similar to the ones in Lemma 2. Once the cycle structure of $\Omega(q(x))$ is known, one can use the methods discussed above to study the properties of the conjugate pairs.

Copying the construction of $\mathscr{P}$, we build $\widetilde{\mathscr{P}}$ by replacing the original polynomial $p_i(x)$ by $p_i^{b_i}(x)$. We know from [6, Lemma 7] that $\widetilde{\mathscr{P}}$ has full rank and can use it to determine the new representation of $\mathbf{S}$ as $(\mathbf{v}_1,\ldots,\mathbf{v}_s)\widetilde{\mathscr{P}}$. Here, $\mathbf{v}_i \in \mathbb{F}_2^{b_i \cdot n_i}$ is a state of a cycle in $\Omega(p_i^{b_i}(x))$ with minimal polynomial $p_i^{b_i}(x)$. Given any two cycles, we deploy Algorithm 3 with inputs the states of all cycles in $\Omega(p_i^{b_i}(x))$ to output pairs of states $(\mathbf{x}_i, \mathbf{y}_i)$ satisfying $\mathbf{x}_i + \mathbf{y}_i = \mathbf{v}_i$. Now, this is where complication arises since we are no longer able to leverage on tools or results from relevant cyclotomic numbers, plus there are likely to be a lot of distinct cycles in $\Omega(p_i^{b_i}(x))$.

The proof of the following result on the pair $(\mathbf{x}_i, \mathbf{y}_i)$ is straightforward. Using the cycle structure of $\Omega(p_i^{b_i}(x))$ and some properties of LFSRs, it may be possible to derive more results on the pair $(\mathbf{x}_i, \mathbf{y}_i)$. We leave them for future investigation.

Proposition 3 *Let $\mathbf{x}_i$ and $\mathbf{y}_i$ be states belonging to respective cycles $C_1 = [\mathbf{s}_1]$ and $C_2 = [\mathbf{s}_2]$ in $\Omega(p_i^{b_i}(x))$. If there exist $a, b \in \mathbb{Z}$ satisfying $T^a\mathbf{x}_i + T^b\mathbf{y}_i = \mathbf{v}_i$, then at least one of $\mathbf{s}_1$ or $\mathbf{s}_2$ must have $p_i^{b_i}(x)$ as minimal polynomial. If $p_1(x) = 1 + x$, then exactly one of $\mathbf{s}_1$ and $\mathbf{s}_2$ has minimal polynomial $(1+x)^{b_i}$.*

Algorithm 4 can also be used to find all conjugate pairs between any two cycles in $\Omega(q(x))$. One needs to exercise greater care here since the cycles in $\Omega(p_i^{b_i}(x))$ may have distinct periods. This affects the application of the generalized CRT (Theorem 2).

Algorithms 3 and 4 perform better for large s, *i.e.*, when $f(x)$ has more distinct irreducible polynomials as its factors. Algorithm 3 is more efficient when there are less cycles and the periods are small. Since $q(x)$ has repeated roots, one has to treat the cycles in $\Omega(p_i^{b_i}(x))$ separately according to their respective periods. A modified version of Algorithm 3 takes more time here since the overall degree is $b_i \cdot n_i$ and there are a lot of cycles to pair up. This typically drives the complexity of finding pairs of states $(\mathbf{x}_i, \mathbf{y}_i)$ satisfying $\mathbf{x}_i + \mathbf{y}_i = \mathbf{v}_i$ much higher than in the case $\Omega\left(\prod_{i=1}^{t} p_i(x)\right)$ where $p_i(x)$ for $1 \le i \le t$ are distinct irreducible polynomails with the same overall degree $b_i \cdot n_i$ where the generalized CRT simplifies the process significantly.

In short, we can slightly modify our approach for $f(x)$ to work on $q(x)$. A characteristic polynomial $q(x)$ with repeated roots can be used to generate de Bruijn sequences using the already mentioned modification on the respective algorithms. Unless one is prepared to commit much more computational resources or is required to produce more de Bruijn sequences that can be constructed based on $f(x)$, using $q(x)$ is generally not advisable.

In the literature, studies on the case of characteristic polynomials with repeated roots have been quite limited, *e.g.*, $q(x) = (x+1)^n$ in [17] and $q(x) = (x+1)^a p(x)$ with $p(x)$ having no repeated roots or is primitive done, respectively, in [22,25]. Prior studies looked into cases with $(x+1)^b \mid q(x)$ for $b \in \mathbb{N}$ because their cycle structures and adjacency graphs had been well-established.

8 Generating the de Bruijn Sequences

After implementing Algorithms 1 to 4 we obtain all of the conjugate pairs between any two adjacent cycles in $\Omega(f(x))$ and, hence, the adjacency graph G. The edges between a specific pair of vertices in G correspond to the conjugate pairs shared by the represented cycles. Derive the graph $\widehat{G}$ from G by bundling together multiple edges incident to the same pair of vertices into one edge. An edge in $\widehat{G}$ corresponds to a set of conjugate pair(s). Next, we use Algorithm 5 to generate all spanning trees in $\widehat{G}$. Line 5 requires that each identified tree contains all vertices in $\widehat{G}$. The `for loop` in Lines 11 to 15 ensures that all vertices are visited and eventually checked and that no cycle occurs. Note that the set B and, hence, X may be empty. Depending on the specifics of the input graph and the choice of $\overline{V}$, given the current sets *VList* and *VCheck*, there are $2^{|B|} < 2^{\deg(\overline{V})}$ choices for X.

Our implementation is in `python`. Readers who prefer to code in `C` may opt to use D. Knuth's implementation of [20, Alg. S, pp. 464 ff.] named `grayspspan` [18]. Its running time is roughly estimated to be $\mathcal{O}(\mu + \psi + \zeta_{\widehat{G}})$ where μ is the total number of edges in $\widehat{G}$, ψ is the number of cycles in $\Omega(f(x))$, and $\zeta_{\widehat{G}}$ is the number of the spanning trees in

Algorithm 5 Finding all Spanning Trees in $\widehat{G}$

Input: $\widehat{G}$ with $V(\widehat{G}) := \{V_1, \ldots, V_\psi\}$ and edge set $E(\widehat{G})$.
Output: All spanning trees in $\widehat{G}$.

$VList \leftarrow \{V_1\}$ ▷ initiate the list of visited vertices; for convenience, $V_1 := [\mathbf{0}]$
$VCheck \leftarrow \varnothing$ ▷ initiate the list of checked vertices
$EList \leftarrow \varnothing$ ▷ initiate the list of collected edges
procedure SPANTREES ($ST(VList, VCheck, EList)$)
  if $|EList| = \psi - 1$ **then**
    return $EList$
  else
    take any vertex $\overline{V} \in (VList \setminus VCheck)$
    append $\overline{V}$ to $VCheck$
    $B \leftarrow \{V_j$ satisfying $(\overline{V}, V_j) \in E(\widehat{G})$ with $V_j \notin VList\}$
    for each $X \subseteq B$ **do**
      $VList \leftarrow VList \cup X$
      $EList \leftarrow EList \cup \{(\overline{V}, V_k)$ with $V_k \in X\}$
      $ST(VList, VCheck, EList)$
    end for
  end if
end procedure

$\widehat{G}$. Notice that $\mu < 2^{n-1}$, $\psi \leq \dfrac{1}{n}\displaystyle\sum_{d|n} \phi(d) 2^{\frac{n}{d}}$ (see, *e.g.*, [30]), and $\zeta_{\widehat{G}}$ is much larger than both μ and ψ. Hence, $\mathscr{O}(\mu + \psi + \zeta_{\widehat{G}}) = \mathscr{O}(\zeta_{\widehat{G}})$. We ran numerous simulations and came to the conclusion that the running time of Algorithm 5 is the same as that of Algorithm S. We prefer the procedure in Algorithm 5 since it leads to a simpler mechanism, both deterministic and random, on how to pick a particular spanning tree to use in the actual generation of the sequences. Next comes the cycle joining procedure.

Let Υ_G be a chosen spanning tree in G and $\{(\mathbf{v}_1, \widehat{\mathbf{v}}_1), \ldots, (\mathbf{v}_{\psi-1}, \widehat{\mathbf{v}}_{\psi-1})\}$ be its edge set. Let $\overline{\mathbf{v}}$ denote the last $n-1$ bits of any $\mathbf{v} \in \mathbb{F}_2^n$. Define a new set $E(\Upsilon_G) := \{\overline{\mathbf{v}}_1, \ldots, \overline{\mathbf{v}}_{\psi-1}\}$. From an arbitrary initial state (any vector in $\mathbb{F}_2^n$), use the LFSR with feedback function $h(x_0, \ldots, x_{n-1})$, *i.e.*, with characteristic polynomial $f(x)$, to generate $a_0, a_1, \ldots, a_{n-1}, a_n, \ldots$. For each input $(a_i, \ldots, a_{i+n-1})_{i \geq 0}$, let

$$\begin{cases} a_{i+n} = h(a_i, \ldots, a_{i+n-1}) & \text{if } (a_{i+1}, \ldots, a_{i+n-1}) \notin E(\Upsilon_G), \\ a_{i+n} = h(a_i, \ldots, a_{i+n-1}) + 1 & \text{if } (a_{i+1}, \ldots, a_{i+n-1}) \in E(\Upsilon_G). \end{cases}$$

The resulting sequence is de Bruijn with feedback function

$$h(x_0, \ldots, x_{n-1}) + \sum_{\mathbf{w} \in E(\Upsilon_G)} \prod_{i=1}^{n-1} (x_i + w_i + 1).$$

Performing the cycle joining procedure to all spanning trees in G gives us all de Bruijn sequences in this class.

To tie up all examples pertaining to $f(x) = (x+1)(x^2+x+1)(x^4+x^3+x^2+x+1)$, label the vertices in G as $V_1, V_2, \ldots, V_{16}$ according to the ordering in Table 1. Since $\mathbf{S}\mathscr{P}^{-1} = (1,1,1,1,0,1,0) = \left(\mathbf{1}, L^2\mathbf{s}_0^2, L^2\mathbf{s}_2^3\right)$ is in $\left[\mathbf{1} + \mathbf{s}_0^2 + \mathbf{s}_2^3\right]$, $\mathbf{S}$ is the initial state of $L^2\left(\mathbf{1} + \mathbf{s}_0^2 + \mathbf{s}_2^3\right)$. Using the appropriate results from Example 2 and running Algorithm 4 yield all the conjugate pairs between any two cycles. We summarize the count in Table 2. The complete adjacency graph G has edges labeled by the conjugate pairs.

Table 2 Number of Conjugate Pairs

Edge	#	Edge	#	Edge	#	Edge	#	Edge	#
$\{V_1,V_{16}\}$	1	$\{V_2,V_{13}\}$	1	$\{V_2,V_{16}\}$	2	$\{V_3,V_{14}\}$	2	$\{V_3,V_{15}\}$	1
$\{V_3,V_{16}\}$	2	$\{V_4,V_{14}\}$	1	$\{V_4,V_{15}\}$	2	$\{V_4,V_{16}\}$	2	$\{V_5,V_{10}\}$	1
$\{V_5,V_{14}\}$	2	$\{V_5,V_{15}\}$	2	$\{V_6,V_{11}\}$	2	$\{V_6,V_{12}\}$	1	$\{V_6,V_{13}\}$	2
$\{V_6,V_{14}\}$	4	$\{V_6,V_{15}\}$	2	$\{V_6,V_{16}\}$	4	$\{V_7,V_{11}\}$	1	$\{V_7,V_{12}\}$	2
$\{V_7,V_{13}\}$	2	$\{V_7,V_{14}\}$	2	$\{V_7,V_{15}\}$	4	$\{V_7,V_{16}\}$	4	$\{V_8,V_9\}$	1
$\{V_8,V_{10}\}$	2	$\{V_8,V_{11}\}$	2	$\{V_8,V_{12}\}$	2	$\{V_8,V_{14}\}$	4	$\{V_8,V_{15}\}$	4

Let $\mathscr{M}_1$ be the diagonal 8×8 matrix with entries, in order, $1,3,5,5,5,15,15,15$. Then

$$\mathscr{M} = \begin{pmatrix} \mathscr{M}_1 & \mathscr{M}_2 \\ \mathscr{M}_2 & \mathscr{M}_1 \end{pmatrix} \text{ with } \mathscr{M}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & -2 & -1 & -2 \\ 0 & 0 & 0 & 0 & 0 & -1 & -2 & -2 \\ 0 & -1 & 0 & 0 & 0 & -2 & -2 & 0 \\ 0 & 0 & -2 & -1 & -2 & -4 & -2 & -4 \\ 0 & 0 & -1 & -2 & -2 & -2 & -4 & -4 \\ -1 & -2 & -2 & -2 & 0 & -4 & -4 & 0 \end{pmatrix}.$$

The cofactor of any entry in $\mathscr{M}$ gives $12,485,394,432 \approx 2^{33.54}$ as the number of constructed de Bruijn sequences of order 7. The matrix representation $\widehat{\mathscr{M}}$ of $\widehat{G}$ has main diagonal entries $1,2,3,3,3,6,6,6,1,2,3,3,3,6,6,6$. For $i \neq j$, the (i,j) entries is -1 whenever V_i and V_j are adjacent and 0 otherwise. Hence, $\zeta_{\widehat{G}} = 1,451,520 \approx 2^{20.47}$, which is easier to process computationally than the number $\zeta_G \approx 2^{33.54}$ of spanning trees in G.

The polynomial $f(x)$ studied in [24] is the product of $x+1$ and some primitive polynomials with coprime periods. Such a choice greatly simplifies the procedure since the resulting cycle structure is simple and the conjugate pairs can be deduced directly. Their main motivation was to be able to generate the sequences quickly. The main drawback is the relatively low number of sequences generated. The construction in [24, Sect. IV], for example, only produces sequences from a special spanning tree named *maximal spanning tree*. The number of de Bruijn sequences generated by [24, Alg. 1] is $\mathscr{O}(2^{(2^{s-1}-1)n})$. From $f(x) = (x+1)p_1(x)p_2(x)$, the number of sequences generated by [24, Alg. 2] is $\mathscr{O}(2^{3n})$. To illustrate the point more concretely, the number of all de Bruijn sequences of order 8 that can be generated from $f(x) = (x+1)(x^3+x^2+1)(x^4+x^3+1)$ is 926,016. Using the same $f(x)$, [24, Alg. 2] produces only 592,704 of them.

9 Implementation

A basic software implementation was written in `python` 2.7 and is available online at `https://github.com/adamasstokhorst/debruijn` [9]. The platform was a laptop having 11.6 GB available memory with Windows 10 operating system powered by an Intel i7-7500U CPU 2.70GHz. Table 3 presents some of the implementation results.

The program determine exactly the number ψ of distinct cycles in $\Omega(f(x))$, the adjacency graph G and its corresponding matrix $\mathscr{M}$, the number ζ_G of de Bruijn sequences that can be generated, and the number $\zeta_{\widehat{G}}$ of all spanning trees in $\widehat{G}$. It then outputs 100 de Bruijn sequences with initial state $\mathbf{0}$. The total running time is denoted by *Run*. For brevity, 2^k with

Table 3 Some Implementation Results

No.	n	$\{p_i(x)\}$	$\{e_i\}$	ψ	ζ_G	$\zeta_{\widehat{G}}$	*Run*
1	6	$\{1011,1101\}$	$\{7,7\}$	10	393 216	51 984	3.57s
2	7	$\{11,111,11111\}$	$\{1,3,5\}$	16	$2^{33.5}$	$2^{20.5}$	1.89s
3	8	$\{11,1101,11001\}$	$\{1,7,15\}$	8	926 016	15	2.77s
4	8	$\{10011,11111\}$	$\{15,5\}$	20	$2^{60.8}$	$2^{53.0}$	16.37s
5	9	$\{111,1011,11111\}$	$\{3,7,5\}$	16	$2^{54.4}$	$2^{28.8}$	5.47s
6	9	$\{11,100111001\}$	$\{1,17\}$	32	$2^{113.4}$	$2^{86.7}$	7.20s
7	10	$\{11,111,1011,11111\}$	$\{1,3,7,5\}$	32	$2^{116.0}$	$2^{61.1}$	16.47s
8	10	$\{11111111111\}$	$\{11\}$	94	$2^{304.9}$	$2^{299.1}$	59.30s
9	11	$\{111,1011,1001001\}$	$\{3,7,9\}$	60	$2^{251.9}$	$2^{190.0}$	2m 32s
10	11	$\{101011100011\}$	$\{23\}$	90	$2^{388.8}$	$2^{373.8}$	1m 27s
11	12	$\{1001001,1010111\}$	$\{9,21\}$	74	$2^{398.7}$	$2^{350.7}$	6m 09s
12	13	$\{11,111,11111,1001001\}$	$\{1,3,5,9\}$	240	$2^{1114.6}$	$2^{853.8}$	24m 12s
13	14	$\{111,11111,100111001\}$	$\{3,5,9\}$	128	$2^{800.2}$	$2^{583.7}$	5m 04s
14	15	$\{1001001,1000000011\}$	$\{9,73\}$	64	$2^{508.6}$	$2^{277.3}$	5m 06s
15	16	$\{1001001,10000001111\}$	$\{9,341\}$	32	$2^{274.2}$	$2^{97.0}$	7m 58s
16	16	$\{11,111,1011,11111,1001001\}$	$\{1,3,7,5,9\}$	480	$2^{2925.8}$	$2^{1966.8}$	15hr 23m
17	17	$\{100111111,1000000011\}$	$\{85,73\}$	32	$2^{310.1}$	$2^{111.3}$	15m 01s
18	18	$\{111010111,10001000111\}$	$\{17,341\}$	64	$2^{630.6}$	$2^{261.5}$	37m 48s
19	19	$\{1001100101,10000110101\}$	$\{73,93\}$	96	$2^{1076.3}$	$2^{530.7}$	1hr 41m
20	20	$\{11111,1001001,10000001111\}$	$\{5,9,341\}$	128	$2^{1365.0}$	$2^{564.4}$	4hr 36m

$k \geq 20$ rounded to one decimal place approximates ζ_G and $\zeta_{\widehat{G}}$ and we remove the $\approx$ sign from the table. Entry 2 is our example while Entry 3 is an example in [24].

To build a library of de Bruijn sequences of order n, one can store the graphs G and generate $\widehat{G}$. Once a tree in $\widehat{G}$ has been identified, the corresponding set of tree(s) in G is listed down. For each of these trees, one uses the cycle joining method to generate all de Bruijn sequences in this class. To generate a random de Bruijn sequence, we can chose a random tree $\Upsilon_{\widehat{G}}$ before selecting one among the corresponding trees in G randomly, say Υ_G. We then use an arbitrary $\mathbf{v} \in \mathbb{F}_2^n$ as the initial state in the cycle joining routine. The randomization does not significantly alter the running time.

A user may want to output one de Bruijn sequence chosen uniformly at random from among all of the constructible sequences. One simply applies Broder's elegant algorithm [3, Alg. Generate] on G (not on $\widehat{G}$) to get a uniformly random tree Υ_G. The algorithm's expected running time per generated tree is $\mathscr{O}(\psi \log \psi)$ for most graphs. The worst case value is $\mathscr{O}(\psi^3)$. One then picks an arbitrary initial state $\mathbf{v}$ from $\mathbb{F}_2^n$ and performs the cycle joining routine with Υ_G and $\mathbf{v}$ as ingredients. Once G has been determined, the rest of the process is very fast.

For large n or s, the required memory and running time can quickly exceed available resources if we insist on building G completely. To generate a few (at least one) de Bruijn sequences of large order, we perform the routine up to generating all the cycles and finding a state in each cycle. We can then proceed to identify a simple connected subgraph containing all cycles in $V(G)$. One way to do this is to index the cycles as $C_1, C_2, \ldots, C_\psi$ before collecting all neighbours of C_1 in a set $\mathscr{N}_{C_1}$. Only one conjugate pair between C_1 and each member of $\mathscr{N}_{C_1}$ needs to be found. We pick the cycle with the smallest index in $\mathscr{N}_{C_1}$ and

determine all of its neighbours in the set $V(G) \setminus (\mathscr{N}_{C_1} \cup C_1)$, listing a single conjugate pair for each pair of cycles. The procedure is repeated until a connected graph containing $V(G)$ is found. The spanning trees in the resulting graph generates de Bruijn sequences.

10 Conclusion and Future Directions

We put forward a method to generate a large class of binary de Bruijn sequences from LFSRs with an arbitrary characteristic polynomial, paying special attention to the case where $f(x)$ is the product of s pairwise distinct irreducible polynomials. The related structures are studied in details. Our approach covers numerous prior constructions as special cases. Table 4 lists their relevant parameters and results arranged chronologically by publication dates. In addition to the cycle structure and the adjacency graph, we note if the feedback functions of the maximum-length NLFSRs are explicitly derived. Discussions on the number of de Bruijn sequences that can be constructed and on the algorithmic steps to generate them are also considered in the list.

Table 4 List of Prior Cycle-Joining Constructions

No.	The Characteristic Polynomial $f(x)$	Ref.	Structure $\Omega(f(x))$	Graph G	Feedback Function	Number of Sequences	Generating Algorithm(s)
1	$(1+x)^n$	[17]	Yes	No	No	Not discussed in the paper	
2	$f(x)$ is irreducible	[15]	Yes	Yes	No	Bounds	Not discussed
3	$(1+x)^m\ p(x)$, $p(x)$ is primitive	[22]	Yes (All m)	Yes (Only applicable for $m=3$)	Yes	Estimated	Implicit, without analysis
4	$(1+x^3)\ p(x)$, $p(x)$ is primitive	[23]	Yes	Yes	Yes	Estimated	For some, not all, sequences
5	$\prod_{i=1}^{s} p_i(x)$, $p_i(x)$ primitive, degrees increasing and pairwise coprime	[24]	Yes (All)	Yes (These results apply only for $p_1(x)=1+x$)	Yes	Estimated	Yes, with analysis
6	Product of 2 irreducibles	[7]	Yes	Yes	Yes	Estimated	Yes, without analysis
7	$\ell(x)\ p(x)$, $p(x)$ primitive, $\ell(x)$ has small degree	[26]	Yes (If periods of $\ell(x)=1+x+x^3+x^4$ and $p(x)$ are coprime)	Yes	Yes	Exact	Implicit, with analysis
8	$\prod_{i=1}^{s} \ell_i(x)$, factors are pairwise coprime	[27]	Yes	Yes	Not discussed		Implicit, without analysis

The followings are useful details on each entry in Table 4.

1. The number of conjugate pairs between any two cycles in [17] is shown to be ≤ 2 but without any method to determine them. A complete adjacency graph is not provided.
2. A lower bound on the number of resulting sequences is stated in [15, Thm. 6]. Exact values in some specific cases are given in [15, Thm. 7 and Thm. 8].
3. The estimated number $\mathscr{O}(2^{4n})$ of the resulting sequences in [22] applies for $m=3, n\geq 5$.
4. Not all possible de Bruijn sequences are constructed in [23] since only special types of spanning trees are used in the cycle joining process. Using this well chosen adjacency subgraph, the time as well as memory complexity of the algorithm that generate $\mathscr{O}(2^{4n})$ and $\mathscr{O}(2^{8n})$ de Bruijn sequences for odd and even n, respectively, is $\mathscr{O}(n)$.
5. Recall that n_s is the degree of $p_s(x)$. When $p_1(x)=1+x$, the number of de Bruijn sequences that can be constructed in [24] is estimated to be $\mathscr{O}\left(2^{(2^k-1)n}\right)$. The algorithm

runs in memory complexity $\mathscr{O}\left(2^{s+1}(s+1)n\right)$ and time complexity $\mathscr{O}\left(2^{n-n_s}(s+1)n\right)$. If $s=3$ and $n\geq 8$, the time complexity can be reduced to $\mathscr{O}\left(n^{\log\log(n)}\right)$.

6. The exact number of de Bruijn sequences that can be constructed and the discussion on the time complexity $\mathscr{O}(n)$ required to completely generate one such sequence are given in [26, Sect. VII].
7. Only a rough estimate on the number of constructed sequences is given in [7, Eq. (21)]. An exact count is provided in [7, Thm. 4] for $f(x)=(1+x+x^2)p(x)$ where $p(x)$ is any primitive polynomial of degree ≥ 3.
8. The main focus in [27] is the adjacency graph. The number of sequences that can be built and their feedback functions are out of the scope and the discussion on how to generate them is limited to some illustrative cases.

In this work we build a de Bruijn sequence generator and back our claims with implementation results. Our basic implementation scenarios can be extended to cover more application-dependent purposes. Using the results established in this paper, one can write an implementation software to identify enough conjugate pairs that yield a spanning tree, from which a de Bruijn sequence of a large order can be built quickly.

Many interesting directions remain open for investigation. A partial list includes the following problems.

1. Optimize the algorithms above to practically generate de Bruijn sequences for larger n.
2. Derive a reasonably tight lower bound on the number of de Bruijn sequences of a very large order that can be generated by cycle joining method based on a given LFSR.
3. Generalize our results to any finite field $\mathbb{F}_q$. This is nontrivial since the presence of multiple nonzero elements changes the definition of a conjugate pair.
4. Determine good lower and upper bounds on the linear complexity of the resulting sequences or their modified version, *i.e.*, those obtained by deleting a zero from the consecutive string of n zeroes. We believe that this problem is very challenging.
5. Use the cycle joining method to generate de Bruijn sequences from NLFSRs.
6. Find really simple rules to quickly generate de Bruijn sequences in the spirit of the *prefer-one method* given in [12].

Acknowledgements Adamas Aqsa Fahreza wrote the `python` implementation code. The work of Z. Chang is supported by the National Natural Science Foundation of China under Grant 61772476 and the Key Scientific Research Projects of Colleges and Universities in Henan Province under Grant 18A110029. Research Grants TL-9014101684-01 and MOE2013-T2-1-041 support the research carried out by M. F. Ezerman, S. Ling, and H. Wang.

References

1. van Aardenne-Ehrenfest, T., de Bruijn, N.G.: Circuits and trees in oriented linear graphs. Simon Stevin **28**, 203–217 (1951)
2. Arndt, J.: Matters Computational: Ideas, Algorithms, Source Code, 1st edn. Springer-Verlag, New York, NY, USA (2010)
3. Broder, A.: Generating random spanning trees. In: Proc. 30th Annual Symposium on Foundations of Computer Science, pp. 442–447 (1989)
4. Bruckstein, A.M., Etzion, T., Giryes, R., Gordon, N., Holt, R.J., Shuldiner, D.: Simple and robust binary self-location patterns. IEEE Trans. Inform. Theory **58**(7), 4884–4889 (2012)
5. de Bruijn, N.G.: A combinatorial problem. Koninklijke Nederlandse Akademie v. Wetenschappen **49**, 758–764 (1946)

6. Chang, Z., Ezerman, M.F., Ling, S., Wang, H.: The cycle structure of LFSR with arbitrary characteristic polynomial over finite fields. Cryptogr. Commun. (2017). DOI 10.1007/s12095-017-0273-2, Online First 20 Dec. 2017
7. Chang, Z., Ezerman, M.F., Ling, S., Wang, H.: Construction of de Bruijn sequences from product of two irreducible polynomials. Cryptogr. Commun. **10**(2), 251–275 (2018)
8. Ding, C., Pei, D., Salomaa, A.: Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1996)
9. Ezerman, M. F., Fahreza, A. A.: A binary de Bruijn sequence generator from product of irreducible polynomials. URL`github.com/adamasstokhorst/debruijn`
10. Etzion, T., Lempel, A.: Algorithms for the generation of full-length shift-register sequences. IEEE Trans. Inform. Theory **30**(3), 480–484 (1984)
11. Fredricksen, H.: A class of nonlinear de Bruijn cycles. J. Combinat. Theory, Ser. A **19**(2), 192 – 199 (1975)
12. Fredricksen, H.: A survey of full length nonlinear shift register cycle algorithms. SIAM Review **24**(2), 195–221 (1982)
13. Golomb, S.W.: Shift Register Sequences. Aegean Park Press, Laguna Hills (1981)
14. Golomb, S.W., Gong, G.: Signal Design for Good Correlation: for Wireless Communication, Cryptography, and Radar. Cambridge Univ. Press, New York (2004)
15. Hauge, E.R., Helleseth, T.: De Bruijn sequences, irreducible codes and cyclotomy. Discrete Math. **159**(1-3), 143–154 (1996)
16. Hauge, E.R., Mykkeltveit, J.: On the classification of de Bruijn sequences. Discrete Math. **148**(13), 65 – 83 (1996)
17. Hemmati, F., Schilling, D.L., Eichmann, G.: Adjacencies between the cycles of a shift register with characteristic polynomial $(1+x)^n$. IEEE Trans. Comput. **33**(7), 675–677 (1984)
18. Knuth, D.E.: Grayspspan. URL`http://www-cs-faculty.stanford.edu/~uno/programs/grayspspan.w`
19. Knuth, D.E.: The Art of Computer Programming. Vol. 2 (3rd Ed.)., Seminumerical Algorithms. Addison-Wesley, Longman Publishing Co., Inc., Boston (1997)
20. Knuth, D.E.: The Art of Computer Programming. Vol. 4A., Combinatorial Algorithms. Part 1. Addison-Wesley, Upple Saddle River (N.J.), London, Paris (2011)
21. Kurosawa, K., Sato, F., Sakata, T., Kishimoto, W.: A relationship between linear complexity and k-error linear complexity. IEEE Trans. Inform. Theory **46**(2), 694–698 (2000)
22. Li, C., Zeng, X., Helleseth, T., Li, C., Hu, L.: The properties of a class of linear FSRs and their applications to the construction of nonlinear FSRs. IEEE Trans. Inform. Theory **60**(5), 3052–3061 (2014)
23. Li, C., Zeng, X., Li, C., Helleseth, T.: A class of de Bruijn sequences. IEEE Trans. Inform. Theory **60**(12), 7955–7969 (2014)
24. Li, C., Zeng, X., Li, C., Helleseth, T., Li, M.: Construction of de Bruijn sequences from LFSRs with reducible characteristic polynomials. IEEE Trans. Inform. Theory **62**(1), 610–624 (2016)
25. Li, M., Jiang, Y., Lin, D.: The adjacency graphs of some feedback shift registers. Des. Codes Cryptogr. **82**(3), 695–713 (2017)
26. Li, M., Lin, D.: The adjacency graphs of LFSRs with primitive-like characteristic polynomials. IEEE Trans. Inform. Theory **63**(2), 1325–1335 (2017).
27. Li, M., Lin, D.: De Bruijn sequences, adjacency graphs and cyclotomy. IEEE Trans. Inform. Theory **64**(4), 2941–2952 (2018)
28. Lidl, R., Niederreiter, H.: Finite Fields. Encyclopaedia of Mathematics and Its Applications. Cambridge Univ. Press, New York (1997)
29. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography, 1st edn. CRC Press, Inc., Boca Raton, FL, USA (1996)
30. Mykkeltveit, J.: A proof of Golomb's conjecture for the de Bruijn graph. J. of Combinat. Theory, Ser. B **13**(1), 40 – 45 (1972)
31. Nagarajan, N., Pop, M.: Sequence assembly demystified. Nature Rev. Genet. **14**(3), 157–167 (2013)
32. Ralston, A.: De Bruijn sequences - a model example of the interaction of discrete mathematics and computer science. Math. Magazine **55**(3), 131–143 (1982)
33. Spinsante, S., Gambi, E.: De Bruijn binary sequences and spread spectrum applications: A marriage possible? IEEE Trans. Aerosp. Electron. Syst. **28**(11), 28–39 (2013)