# Covering Sequences for $\ell$-Tuples

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

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

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

***Abstract*—de Bruijn sequences of order $\ell$, i.e., sequences that contain each $\ell$-tuple as a window exactly once, have found many diverse applications in information theory and most recently in DNA storage. This family of binary sequences has asymptotic rate of $1/2$. To overcome this low rate, we study $\ell$-tuples covering sequences, which impose that each $\ell$-tuple appears at least once as a window in the sequence. The cardinality of this family of sequences is analyzed while assuming that $\ell$ is a function of the sequence length $n$. Lower and upper bounds on the asymptotic rate of this family are given. Moreover, we study an upper bound for $\ell$ such that the redundancy of the set of $\ell$-tuples covering sequences is at most a single symbol. We present an efficient encoding and decoding schemes for $\ell$-tuples covering sequences that meet this bound.**

## I. Introduction

The binary de Bruijn graph of order $\ell$, $G_\ell$, was introduced in 1946 by de Bruijn [2]. His target in introducing this graph was to find a recursive method to enumerate the number of cyclic binary sequences of length $2^\ell$ such that each $\ell$-tuple appears as a window exactly once in each sequence. These sequences were later called *de Bruijn sequences*. These results were later generalized in [20] for any alphabet of finite size $q$, using a $q$-ary generalization of the de Bruijn graph of order $\ell$, $G_{q,\ell}$.

The vertices of $G_{q,\ell}$ are the $q$-ary $(\ell-1)$-tuples, and its edges correspond to the $q$-ary $\ell$-tuples. There is an edge $u \to v$ if $v$ can be obtained from $u$ by shifting one entry left and appending a symbol. Eulerian cycles in de Bruijn graphs, i.e., cycles that visit all the edges of $G_{q,\ell}$ exactly once, are called *de Bruijn cycles*. It was proved that the number of de Bruijn cycles in $G_{q,\ell}$ is $(q!)^{q^{\ell-1}}/q^\ell$ [20].

Each de Bruijn cycle induces a single (cyclic) de Bruijn sequence of length $q^\ell$, by picking any edge in the cycle as a starting point, considering its first entry and appending the first entry of each consecutive edge in the cycle. All sequences that can be generated in this way are considered as the same sequence. Contrary to this, each de Bruijn cycle induces $q^\ell$ distinct *acyclic de Bruijn sequences*, i.e., sequences of length $q^\ell+\ell-1$ that contain each $\ell$-tuple as a window exactly once, using a similar method with the exception of appending the $(\ell-1)$-suffix of the last edge as well; each sequence corresponds to a choice of different starting edge from $G_{q,\ell}$. Hence, the number of such acyclic de Bruijn sequences is $(q!)^{q^{\ell-1}}$ and their asymptotic rate is $\log_q(q!)/q$ (equals $1/2$ for $q=2$). One of the first applications of the de Bruijn graph was in the introduction of shift-register sequences and linear feedback shift registers [10]. Throughout the years, an extensive number of papers have studied the de Bruijn sequences and their applications, several of those include [3], [5], [7], [8], [12], [15], [17]. Most recently, DNA storage has brought fresh interest to this family of sequences; for more information on such applications the reader is referred to [1], [11], [19].

This paper studies a novel generalization of de Bruijn sequences (for the rest of this paper we refer only to acyclic sequences). We say that a sequence is an *$\ell$-tuples covering sequence* if it contains each $q$-ary $\ell$-tuple as a window at least once. This work follows recent generalizations of de Bruijn sequences that proposed unique variations regarding the appearances of $\ell$-tuples in the sequence: *$\ell$-repeat free sequences* [6], [9] require each $\ell$-tuple to appear at most once, *$(b,\ell)$-locally-constrained de Bruijn sequences* [4] require each $\ell$-tuple to appear at most once in every window of length $b$, and *$(\ell,\mu)$-balanced de Bruijn sequences* [14] require each $\ell$-tuple to appear exactly $\mu$ times in the sequence.

Notice that for sequences of length $q^\ell+\ell-1$, all $\ell$-tuples covering sequences are simply the de Bruijn sequences deduced from the de Bruijn graph $G_{q,\ell}$; as a result, their asymptotic rate is $\log_q(q!)/q$. Our main goal is to efficiently construct codes of $\ell$-tuples covering sequences with higher rates (specifically larger than $1/2$ for binary sequences) and fixed number of redundancy symbols. We study the cardinality for the set of $\ell$-tuples covering sequences and present lower bounds on its asymptotic rate for various values of $\ell$. Additionally, we present an upper bound for $\ell$ such that the redundancy of a set of $\ell$-tuples covering sequences is at most one symbol. Later, we present an encoding algorithm for the set of binary $\ell$-tuples covering sequences that uses a single redundancy bit and meets this bound on $\ell$. Finally, we use a generalization of de Bruijn graph to develop an upper bound for the cardinality of this set of sequences.

Another interesting family of sequences is introduced as a building block to our analysis of $\ell$-tuples covering sequences. For some $\ell$-tuple $\boldsymbol{v}$, we say that a sequence is a *$\boldsymbol{v}$-avoiding sequence* if it does not contain $\boldsymbol{v}$ as a window. Note that if $\boldsymbol{v}$ is the all-zero $\ell$-tuple, then this family of sequences is known as RLL sequences and was studied before, for example in [13], [18]. We study this family of sequences for any $\ell$-tuple $\boldsymbol{v}$.

The rest of this paper is organized as follows. In Section II we formally define the families of sequences studied in this paper and review several previous results. In Section III, we study the family of $\boldsymbol{v}$-avoiding sequences for any $\ell$-tuple $\boldsymbol{v}$. Based on these results, in Section IV we analyze the cardinality of $\ell$-tuples covering sequences and present an encoding scheme for $q=2$ which uses a single redundancy bit. Due to the lack of space, some of the proofs in this paper are omitted.

## II. Definitions and Preliminaries

For two integers $i,k \in \mathbb{N}$ such that $i \leqslant k$ we denote by $[i,k]$ the set $\{i,\ldots,k\}$ and use $[k]$ as a shorthand for $[0,k-1]$. We use the notation $\Sigma_q = \{0,1,\ldots,q-1\}$ as the alphabet of finite size $q$. For simplicity, when $q=2$, we omit the parameter $q$ from this notation and similar ones.

Let $n \in \mathbb{N}$ and let $\boldsymbol{w} = (w_0,\ldots,w_{n-1}) \in \Sigma_q^n$ denote a sequence of length $n$. For two positive integers $i$

978-1-6654-2159-1/22/$31.00 ©2022 IEEE

and $k$ such that $i+k-1 \leqslant n$, let $\boldsymbol{w}_{i,k}$ denote the substring $(w_i, \ldots, w_{i+k-1})$. Additionally, let $\mathrm{Pref}_k(\boldsymbol{w}) \triangleq \boldsymbol{w}_{0,k}$, $\mathrm{Suff}_k(\boldsymbol{w}) \triangleq \boldsymbol{w}_{n-k,k}$ denote the *k-prefix*, *k-suffix* of $\boldsymbol{w}$, respectively. The notation $\boldsymbol{w} \circ \boldsymbol{v}$ is the concatenation of $\boldsymbol{w}$ and another sequence $\boldsymbol{v}$, and $\boldsymbol{w}^i$ denotes the concatenation of $\boldsymbol{w}$ $i$ times, i.e., $\boldsymbol{w}^i = \boldsymbol{w} \circ \boldsymbol{w}^{i-1}$. Finally, the redundancy of a set $A \subseteq \Sigma_q^n$ is defined as $\mathrm{red}(A) \triangleq n - \log_q |A|$.

**Definition 1.** *The $\ell$-th order $q$-ary **de Bruijn graph** $G_{q,\ell}$ is the digraph $(V, E)$, where $V = \Sigma_q^{\ell-1}$ and*

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

Note that the edges of $G_{q,\ell}$ correspond to the set of $q$-ary $\ell$-tuples, $\Sigma_q^\ell$.

**Definition 2.** *Let $\ell > 1$ be an integer and $n = q^\ell + \ell - 1$. A sequence $\boldsymbol{s} \in \Sigma_q^n$ is called a **de Bruijn sequence** of order $\ell$ if $\boldsymbol{s}$ contains each $q$-ary $\ell$-tuple as a window exactly once.*

Let $\mathcal{S}_q(\ell)$ denote the set of $q$-ary de Bruijn sequences of order $\ell$. The connection between Eulerian cycles in $G_{q,\ell}$ to de Bruijn sequences is as follows. In order to generate a sequence from a cycle, we pick any edge in the cycle and set its first entry as the start of the sequence. Then, we append to the sequence the first entry of each consecutive edge in the cycle. Finally, we append the $(\ell-1)$-suffix of the last edge of the cycle to form the whole sequence. Note that since each edge of $G_{q,\ell}$ can be picked as the first edge of the Eulerian cycle, a single cycle generates $q^\ell$ unique de Bruijn sequences.

**Example 1.** Let $q = 2, \ell = 3, n = 10$. The sequence $\boldsymbol{s} = 0001011100$ is a de Bruijn sequence. $\boldsymbol{s}$ can be generated from $G_{q,\ell}$ using the Eulerian cycle

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

Recall that the number of de Bruijn cycles in $G_{q,\ell}$ is $(q!)^{q^{\ell-1}}/q^\ell$. Since each de Bruijn cycle generates $q^\ell$ unique de Bruijn sequences of length $q^\ell + \ell - 1$, it follows that $|\mathcal{S}_q(\ell)| = (q!)^{q^{\ell-1}}$. Therefore, the asymptotic rate of $\mathcal{S}_q(\ell)$ is

$$\limsup_{\ell\to\infty} \frac{\log_q |\mathcal{S}_q(\ell)|}{q^\ell + \ell - 1} = \frac{\log_q(q!)}{q}.$$

Note that when $q = 2$, this asymptotic rate equals $1/2$. However, for $q \to \infty$, it approaches 1.

Next, we introduce the main family of sequences that is discussed in this paper.

**Definition 3.** *Let $n, \ell$ be integers. A sequence $\boldsymbol{w} \in \Sigma_q^n$ is called an $\ell$-**tuples covering sequence** if $\boldsymbol{w}$ contains each $q$-ary $\ell$-tuple as a window at least once, i.e., for each $\boldsymbol{v} \in \Sigma_q^\ell$, there exists $i \in [n-\ell+1]$ such that $\boldsymbol{w}_{i,\ell} = \boldsymbol{v}$.*

**Example 2.** Let $q = 2, \ell = 3, n = 13$. The sequence $\boldsymbol{w}_1 = 0001001110101$ is an $\ell$-tuples covering sequence. However, the sequence $\boldsymbol{w}_2 = 1001001110101$ is not an $\ell$-tuples covering sequence, since it does not contain the 3-tuple 000.

We denote the set of all $q$-ary $\ell$-tuples covering sequence over $\Sigma_q^n$ by $\mathcal{R}_q(n,\ell)$ and notate the size of such code by $r_q(n,\ell) \triangleq |\mathcal{R}_q(n,\ell)|$. For a window length that is a function of $n$, that is $\ell = f(n)$, we denote the asymptotic rate of $\mathcal{R}_q(n, f(n))$ by

$$\mathbb{R}_q(\ell) \triangleq \limsup_{n\to\infty} \frac{\log_q r_q(n, f(n))}{n}.$$

Note the following connection between $\ell$-tuples covering sequences and de Bruijn sequences; if $n = q^\ell + \ell - 1$, then the set $\mathcal{R}_q(n,\ell)$ is exactly the set of de Bruijn sequences $\mathcal{S}_q(\ell)$. Therefore, $\mathbb{R}_q(\ell) = \log_q(q!)/q$ in this case. In Section IV we study the cardinality of $\mathcal{R}_q(n,\ell)$ for various sizes of $\ell$, i.e., for various functions $f(n)$. Moreover, we present an encoding algorithm for $q = 2$ that uses a single redundancy bit.

## III. $\boldsymbol{v}$-Avoiding Sequences

In this section, we present the auxiliary family of *$\boldsymbol{v}$-avoiding sequences* that is used later in our analysis of $\ell$-tuples covering sequences in Section IV.

**Definition 4.** *Let $\ell$ be an integer and $\boldsymbol{v} \in \Sigma_q^\ell$ a fixed $\ell$-tuple. The set of **$\boldsymbol{v}$-avoiding sequences** over $\Sigma_q^n$, denoted by $\mathcal{A}_q(n,\boldsymbol{v})$ contains all $q$-ary sequences of length $n$ that do not contain $\boldsymbol{v}$ as a window. Namely,*

$$\mathcal{A}_q(n,\boldsymbol{v}) = \{\boldsymbol{w} \in \Sigma_q^n \mid \forall i \in [n-\ell+1], \boldsymbol{w}_{i,\ell} \neq \boldsymbol{v}\}$$

For a given $\ell$-tuple $\boldsymbol{v}$, we notate the size of this code by $a_q(n,\boldsymbol{v}) \triangleq |\mathcal{A}_q(n,\boldsymbol{v})|$. Note that for $\boldsymbol{v} = 0^\ell$, this family of sequences is the family of $(0,\ell-1)$-RLL sequences [18] (for integers $d,k$, a $(d,k)$-RLL sequence satisfies that the number of zeros between two consecutive ones is in the range $[d,k]$). These sequences were studied extensively in [13] for different functions $\ell = f(n)$.

We are motivated to study this family of sequences due to the following connection to the family of $\ell$-tuples covering sequences; a sequence $\boldsymbol{s}$ is an $\ell$-tuples covering sequence if and only if for every $\boldsymbol{v} \in \Sigma_q^\ell$, $\boldsymbol{s}$ is not a $\boldsymbol{v}$-avoiding sequence. This connection will be utilized later in order to encode and analyze the cardinality of $\mathcal{R}_q(n,\ell)$.

First, we give an upper bound for $a_q(n,\boldsymbol{v})$ for any $\boldsymbol{v} \in \Sigma_q^\ell$ in order to use it later to estimate the cardinality of $\mathcal{R}_q(n,\ell)$.

**Lemma 5.** *Let $n, \ell$ be positive integers such that $\ell \leqslant n$, and let $\boldsymbol{v} \in \Sigma^\ell$ be any $\ell$-tuple. Then,*

$$a_q(n,\boldsymbol{v}) \leqslant q^{n - c_1 \frac{n-2\ell}{q^\ell}},$$

*where $c_1 = \frac{(q-1)^2 \log_q e}{4q^2}$ ($e$ is the base of the natural logarithm).*

Next, we focus on binary sequences, i.e., $q = 2$, and present a $\boldsymbol{v}$-avoiding sequences compression algorithm for any $\boldsymbol{v}$ of length $\ell \leqslant \log n - 6$ and $n$ large enough. The algorithm receives a $\boldsymbol{v}$-avoiding sequence of length $n$ and outputs a unique unconstrained sequence of length $n-1$. Clearly, this algorithm can be used for any $\ell' < \ell$ by padding $\boldsymbol{v}$ to size $\ell$ and continuing regularly; hence we assume from now on that $\ell = \log n - 6$. This compression algorithm will be utilized in Section IV to encode binary $\ell$-tuples covering sequences.

First, we present some useful notations. For a sequence $\boldsymbol{s} \in \Sigma^n$, let $p(\boldsymbol{s})$ denote its *period*, that is, the smallest positive integer that satisfies $s_i = s_{i+p(\boldsymbol{s})}$ for every $i \in [n - p(\boldsymbol{s})]$. For every $\boldsymbol{v} \in \Sigma^\ell$, we denote two functions,

$$f_1(\boldsymbol{v}) = \boldsymbol{v} \circ (1 - v_{|\boldsymbol{v}| \bmod p(\boldsymbol{v})})$$

$$f_2(\boldsymbol{v}) = \mathrm{Pref}_{\lfloor |\boldsymbol{v}|/2 \rfloor + 3}(\boldsymbol{v}) \circ f_1(\mathrm{Suff}_{\lceil |\boldsymbol{v}|/2 \rceil - 3}(\boldsymbol{v})).$$

Note that both functions append a single bit to $\boldsymbol{v}$. We have the following lemma,

**Lemma 6.** *For every $\boldsymbol{v} \in \Sigma^\ell$, $p(f_1(\boldsymbol{v})) \geqslant \lceil (\ell+1)/2 \rceil$.*

We say that a sequence has a *long period* if its period is at least half its length, i.e., $p(\boldsymbol{v}) \geqslant \lceil |\boldsymbol{v}|/2 \rceil$. Hence, from Lemma 6, for every $\boldsymbol{v} \in \Sigma^\ell$, $f_1(\boldsymbol{v})$ has a long period, and $f_2(\boldsymbol{v})$ satisfies that its $(\lceil \ell/2 \rceil - 2)$-suffix has a long period. These functions are utilized in the following compression algorithm.

The $\boldsymbol{v}$-avoiding compression algorithm (Algorithm 1) receives a sequence $\boldsymbol{s} \in \mathcal{A}(n, \boldsymbol{v})$ for $\boldsymbol{v} \in \Sigma^\ell$ and compresses it to some uniquely decodable sequence $\boldsymbol{x} \in \Sigma^{n-1}$. Initially, the algorithm checks the first bit of $\boldsymbol{s}$. If it is zero, then the rest of $\boldsymbol{s}$ is returned as the result (see Figure 1). Otherwise, an index $i$ is decoded from the subsequent $\log n - 1$ bits of $\boldsymbol{s}$ (by converting this binary sequence to its integer representation) and the algorithm will construct $\boldsymbol{x}$ by inserting an occurrence of $\boldsymbol{v}$ at this index. However, since such an insertion might create new instances of $\boldsymbol{v}$ in the sequence $\boldsymbol{x}$, 5 additional bits are appended to $\boldsymbol{v}$ in order to ensure that the insertion index can always be deduced by the decoder (see Figure 2). The redundancy bits are added as follows; first, two bits are appended to $\boldsymbol{v}$ (independently of the input sequence $\boldsymbol{s}$) to construct $\boldsymbol{u}$, a sequence that satisfies that both $\boldsymbol{u}$ and $\mathrm{Suff}_{\lceil (\ell+1)/2 \rceil - 2}(\boldsymbol{u})$ have long periods. As a result, when $\boldsymbol{u}$ is inserted at position $i$, at most three new occurrences can be created to the right of it (see Lemma 7 which follows). These cases are eliminated using the 3 remaining bits appended to $\boldsymbol{u}$, denoted by $\boldsymbol{a}$. The result is a sequence $\boldsymbol{x} \in \Sigma^{n-1}$ with its rightmost occurrence of $\boldsymbol{u}$ at position $i$.

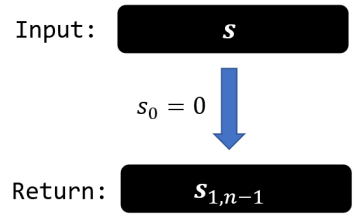


**Figure 1**. Algorithm 1 illustration for the case $s_0 = 0$.

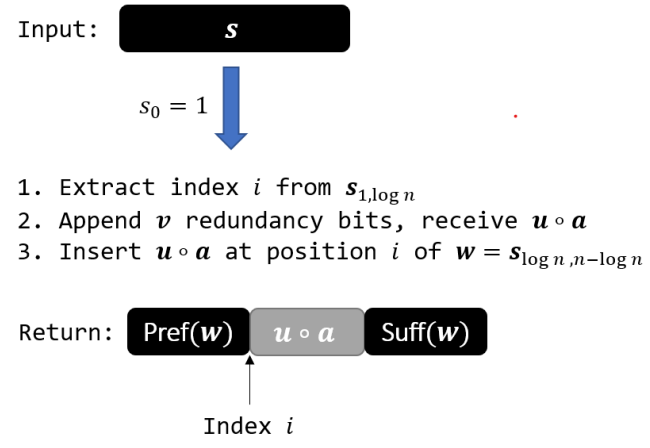


**Figure 2**. Algorithm 1 illustration for the case $s_0 = 1$.

**Algorithm 1** $\boldsymbol{v}$-avoiding compression algorithm

**Input:** A sequence $\boldsymbol{s} \in \mathcal{A}(n, \boldsymbol{v})$
**Output:** A sequence $\boldsymbol{x} \in \Sigma^{n-1}$

1: If $s_0 = 0$, return $\boldsymbol{s}_{1,n-1}$. Otherwise, decode index $i$ from $\boldsymbol{s}_{1,\log n-1}$ and set $\boldsymbol{w} = \mathrm{Suff}_{n-\log n}(\boldsymbol{s})$
2: Construct $\boldsymbol{u} = f_2(f_1(\boldsymbol{v}))$
3: Let

$$A = \{p(\boldsymbol{u}) - 3 \leqslant m \leqslant |\boldsymbol{u}| - 1 : \boldsymbol{w}_{i+1,m} = \mathrm{Suff}_m(\boldsymbol{u})\}$$

4: Set $\boldsymbol{a} = 000$, notate the elements of $A$ in decreasing order $m_2 > m_1 > m_0 \in A$ (starting from $m_2$)
5: **for** every $m_k \in A$ **do**
6: Set $a_k = 1 - u_{\ell-1-m_k+k}$
7: **end for**
8: Return $\boldsymbol{x} = \mathrm{Pref}_i(\boldsymbol{w}) \circ \boldsymbol{u} \circ \boldsymbol{a} \circ \mathrm{Suff}_{|\boldsymbol{w}|-i}(\boldsymbol{w})$

**Lemma 7.** *At Step 3 of Algorithm 1, we have that $|A| \leqslant 3$ .*

**Lemma 8.** *After Step 8 of Algorithm 1, $\boldsymbol{x}$ has its rightmost occurrence of $\boldsymbol{u}$ at position $i$.*

**Theorem 9.** *Algorithm 1 compresses a sequence $\boldsymbol{s} \in \mathcal{A}(n, \boldsymbol{v})$ to a sequence $\boldsymbol{x} \in \Sigma^{n-1}$ that can be uniquely decoded to its input sequence $\boldsymbol{s}$. The time complexity of the algorithm is $\mathcal{O}(\log n)$ and the time complexity of its decoder is $\mathcal{O}(n)$.*

A decoder for this algorithm constructs $\boldsymbol{u} = f_2(f_1(\boldsymbol{v}))$ and looks for the rightmost occurrence of $\boldsymbol{u}$ in $\boldsymbol{x}$. If such occurrence is found, then it must be at the insertion index $i$ from Lemma 8. From here, reconstructing $\boldsymbol{s}$ is straightforward.

**Remark 1.** Note that Algorithm 1 is generic and fits any $\boldsymbol{v} \in \Sigma^\ell$. Clearly, with some knowledge of the tuple $\boldsymbol{v}$ some steps can be skipped and the supported tuple length $\ell$ can be larger. For example, if $\boldsymbol{v}$ has a long period it is unnecessary to invoke $f_1$ at Step 2 and $\ell = \log n - 5$ can be used. If $p(\boldsymbol{v}) = \ell$, i.e., $\boldsymbol{v}$ is aperiodic, then no additional bits are necessary (the algorithm returns $\mathrm{Pref}_i(\boldsymbol{w}) \circ \boldsymbol{v} \circ \mathrm{Suff}_{|\boldsymbol{w}|-i}(\boldsymbol{w})$) and the algorithm is applicable to $\ell = \log n - 1$.

**Example 3.** Let $\ell = 9$, $\boldsymbol{v} = 010101010$. Notice that $p(\boldsymbol{v}) = 2$. We can construct from advance $f_1(\boldsymbol{v}) = 0101010100$, and since $|f_1(\boldsymbol{v})|/2 - 3 = 2$,

$$\boldsymbol{u} = f_2(f_1(\boldsymbol{v})) = 01010101 \circ f_1(00) = 01010101001.$$

Notice that $p(\boldsymbol{u}) = 9$.

Let $n = 2^{15}$ and assume the input sequence $\boldsymbol{s}_1 = 0^n \in \mathcal{A}(n, \boldsymbol{v})$. In this case, the algorithm simply returns $\boldsymbol{x}_1 = 0^{n-1}$ at Step 1. The decoder receives $\boldsymbol{x}_1$ and notices that no occurrences of $\boldsymbol{v}$ exist in the sequence, thus correctly returns $0 \circ \boldsymbol{x}_1 = \boldsymbol{s}_1$.

Next, assume $\boldsymbol{s}_2 = (10101001)^{2^{12}} \in \mathcal{A}(n, \boldsymbol{v})$. One can easily verify that $|\boldsymbol{s}_2| = n$ and that $\boldsymbol{v}$ does not appear as a window in $\boldsymbol{s}_2$. At Step 1, since the first bit of the sequence is 1, the algorithm decodes an index $i$ from the integer value of $(\boldsymbol{s}_2)_{1,14} = 01010011010100$, that is $i = 5332$, and denotes by $\boldsymbol{w}$ the unused part of $\boldsymbol{s}_2$. Soon, the algorithm will insert the tuple $\boldsymbol{u}$ at position $i$. However, we need to ensure that this

insertion does not create additional occurrences of $\boldsymbol{u}$ that start to the right of $i$.

Since $\boldsymbol{w}_{i+1,10} = 1010100110$, the set of indices that is constructed at Step 3 is $A = \{10\}$. Note that we have $|A| = 1$ although by Lemma 7 the maximal size of $A$ is 3. At Step 6 the algorithm constructs $\boldsymbol{a} = 001$ where $a_2 = 1$ ensures that a new occurrence of $\boldsymbol{u}$ is not created when concatenating $\boldsymbol{u}$ and $\boldsymbol{w}_{i+1,10}$. Finally, at Step 8 the algorithm returns the sequence

$$\boldsymbol{x}_2 = \mathrm{Pref}_i(\boldsymbol{w}) \circ 01010101001 \circ 001 \circ \mathrm{Suff}_{|\boldsymbol{w}|-i}(\boldsymbol{w}).$$

The decoder receives $\boldsymbol{x}_2$ and identifies the rightmost occurrence of $\boldsymbol{v}$ at position $i = 5332$. Thus, it constructs

$$\boldsymbol{s}_2 = 1 \circ b(i) \circ \mathrm{Pref}_i(\boldsymbol{x}_2) \circ \mathrm{Suff}_{|\boldsymbol{x}_2|-i-5}(\boldsymbol{x}_2).$$

## IV. $\ell$-Tuples Covering Sequences

In this section, we focus on the main family of sequences discussed in this paper, $\ell$-tuples covering sequences. We study the cardinality of this set of sequences and present a lower bound on its asymptotic rate for various values of $\ell$. Then, we present an upper bound for $\ell$ such that the redundancy of a set of $\ell$-tuples covering sequences is at most one symbol. Later, we present an encoding algorithm for binary $\ell$-tuples covering sequences that uses a single redundancy bit that meets this bound on $\ell$. Finally, we use a generalization of de Bruijn graph to develop an upper bound for the cardinality of this set of sequences.

### A. Rate Lower Bound Analysis

We begin the discussion of $\ell$-tuples covering sequences for any alphabet of size $q$. For integers $n, \ell$, it is clear that if $n < q^\ell + \ell - 1$ then $r_q(n,\ell) = 0$ since a sequence of such length can contain at most $n - \ell + 1$ unique $\ell$-tuples where $n - \ell + 1 < q^\ell$. In the case where $n = q^\ell + \ell - 1$ the set $\mathcal{R}_q(n,\ell)$ is exactly the set of de Bruijn sequences of order $\ell$, and hence $r_q(n,\ell) = |\mathcal{S}_q(\ell)|$. For larger values of $n$, we have the following lemma.

**Lemma 10.** *Let $n = q^\ell + \ell - 1 + k$ for $\ell, k \in \mathbb{N}$. Then,*

$$r_q(n,\ell) \geqslant (q!)^{q^{\ell-1}} \cdot q^k.$$

***Proof.*** We construct $\ell$-tuples covering sequences of length $n$ using any de Bruijn sequence of order $\ell$ followed by any $k$ symbols from $\Sigma_q$. The result follows immediately from the size of $\mathcal{S}_q(\ell)$. ∎

Therefore, we have the next results.

**Corollary 11.** *Let $n = q^\ell + \ell - 1 + f(n)$ for $\ell \in \mathbb{N}$. Then,*

$$\mathbb{R}_q(\ell) \geqslant \begin{cases} \frac{\log_q(q!)}{q} & f(n) = o(q^\ell) \\ \frac{q^{-1}\log_q(q!)+\alpha}{1+\alpha} & f(n) = \alpha q^\ell + o(q^\ell) \text{ for } \alpha > 0 \\ 1 & f(n) = \omega(q^\ell) \end{cases}$$

Alternatively, we have

**Corollary 12.** *Let $n \in \mathbb{N}$ and let $\ell = \log n - g(n)$. Then,*

$$\mathbb{R}_q(\ell) \geqslant \begin{cases} \frac{\log_q(q!)}{q} & g(n) = o(1) \\ 1 + \frac{1}{q^{c+1}}\log_q(q!) - \frac{1}{q^c} & g(n) = c \text{ for } c > 0 \\ 1 & g(n) = \omega(1) \end{cases}$$

For convenience, we use both representations of $n$ and $\ell$ throughout this section.

### B. Single Symbol Redundancy Analysis

Next, we present an upper bound on $\ell$, where the redundancy of $\mathcal{R}_q(n,\ell)$ is at most a single symbol. This bound uses the upper bound on the cardinality of $\boldsymbol{v}$-avoiding sequences presented in Section III.

**Theorem 13.** *If $n, \ell$ are integers such that $\ell \leqslant \log_q n - \log_q \log_q n - \mathcal{O}(1)$, then for $n$ large enough, red$(\mathcal{R}_q(n,\ell)) \leqslant 1$.*

*Proof:* For every sequence $\boldsymbol{w} \in \Sigma_q^n$ that is not an $\ell$-tuples covering sequence, there exists $\boldsymbol{v} \in \Sigma_q^\ell$ such that $\boldsymbol{w}$ is a $\boldsymbol{v}$-avoiding sequence, i.e., $\boldsymbol{w} \in \mathcal{A}_q(n,\boldsymbol{v})$. Thus, using Lemma 5, the number of sequences that are not $\ell$-tuples covering sequences can be bounded above by

$$\sum_{\boldsymbol{v}\in\Sigma_q^\ell} a_q(n,\boldsymbol{v}) \leqslant q^{n - c_1 \frac{n-2\ell}{q^\ell} + \ell}, \tag{1}$$

where $c_1 = \frac{(q-1)^2 \log_q e}{4q^2}$. Therefore, in order to have red$(\mathcal{R}_q(n,\ell)) \leqslant 1$, i.e., $r_q(n,\ell) \geqslant q^{n-1}$, we require that the right-hand side of equation (1) is bounded above by $(q-1)q^{n-1}$, which is satisfied by applying $\ell \leqslant \log_q n - \log_q \log_q n - c_2$ for a constant $c_2 > -\log_q c_1$. ∎

### C. Encoding Algorithm for the Binary Case

For the rest of this section, we assume that $q = 2$. We present an encoder for $\ell$-tuples covering sequences over $\Sigma^n$. This encoder uses a single redundancy bit and handles $\ell$-tuples of length $\ell \leqslant \log n - \log\log n - 6$ for $n$ large enough. Note that this value of $\ell$ is associated with the bound presented in Theorem 13.

This algorithm is based on the compressor of $\boldsymbol{v}$-avoiding sequences presented in Algorithm 1. For $\boldsymbol{v} \in \Sigma^\ell$, let $\mathcal{E}_{\boldsymbol{v}}$ denote this compressor, i.e., $\mathcal{E}_{\boldsymbol{v}}$ receives a $\boldsymbol{v}$-avoiding sequence of length $n$ such that $\ell \leqslant \log n - 6$ and outputs an unconstrained and uniquely decodable sequence over $\Sigma^{n-1}$. Let $n_{\mathcal{E}}$ denote the maximal sequence length that can be compressed with $\mathcal{E}_{\boldsymbol{v}}$ such that $|\boldsymbol{v}| = \ell$, that is, $n_{\mathcal{E}} = 2^{\ell+6} = n/\log n$. Moreover, $\mathcal{E}_{\boldsymbol{v}}$ can be used to efficiently compress $\boldsymbol{v}$-avoiding sequences of length $n \geqslant n_{\mathcal{E}}$ as well; the input sequence is split to consecutive segments of length $n_{\mathcal{E}}$ and each of them is compressed separately. If there is a remainder smaller than $n_{\mathcal{E}}$, it is not compressed at all. This way, a $\boldsymbol{v}$-avoiding sequence of length $n$ can be compressed to a uniquely decodable sequence of length $n - \lfloor n/n_{\mathcal{E}} \rfloor$. We abuse the notation $\mathcal{E}_{\boldsymbol{v}}$ to denote this generalized compressor as well. Similarly, let $\mathcal{D}_{\boldsymbol{v}}$ denote the matching decoder of $\mathcal{E}_{\boldsymbol{v}}$ for any $\boldsymbol{v} \in \Sigma^\ell$.

Algorithm 2 receives as an input $\boldsymbol{w}$, a sequence of length $n-1$ and outputs $\boldsymbol{x}$, an $\ell$-tuples covering sequence of length $n$. The goal of the algorithm is to shorten the input sequence enough in order to make room for appending a de Bruijn sequence of order $\ell$ at its end. The shortening procedure uses the family of compressors $\{\mathcal{E}_{\boldsymbol{v}} \mid \boldsymbol{v} \in \Sigma^\ell\}$, based on the observation that as long as the sequence is not an $\ell$-tuples covering sequence, then it is $\boldsymbol{v}$-avoiding for some tuple $\boldsymbol{v} \in \Sigma^\ell$. Let $\boldsymbol{s}$ denote a fixed de Bruijn sequence of length $2^\ell + \ell - 1$; $\boldsymbol{s}$ can be produces with time complexity of $\mathcal{O}(\ell 2^\ell)$, see [8]. The algorithm first sets $\boldsymbol{x} = 0 \circ \boldsymbol{w}$ in order to mark the start of the encoding process for the decoder. Then, as long as $\boldsymbol{x}$ is not $\ell$-tuples covering, the algorithm repeatedly shortens $\boldsymbol{x}$ by finding an $\ell$-tuple $\boldsymbol{v}$ that

does not appear in $\boldsymbol{x}$. The algorithm encodes the occurrence and compresses $\boldsymbol{x}$ using $\mathcal{E}_v$. This process ends when $\boldsymbol{x}$ is either an $\ell$-tuples covering sequence or it is short enough to be appended by $\boldsymbol{s}$. Either way, this results with an $\ell$-tuples covering sequence which is returned after being padded to length $n$.

**Algorithm 2** $\ell$-tuples covering sequences encoding

**Input:** A sequence $\boldsymbol{w} \in \Sigma^{n-1}$
**Output:** A sequence $\boldsymbol{x} \in \mathcal{R}(n,\ell)$

1: Set $\boldsymbol{x} = 0 \circ \boldsymbol{w}$
2: **while** $\boldsymbol{x}$ is not $\ell$-tuples covering and $|\boldsymbol{x}| > n - |\boldsymbol{s}|$ **do**
3: Pick $\boldsymbol{v} \in \Sigma^{\ell} \setminus \{\boldsymbol{x}_{i,\ell} : i \in [|\boldsymbol{x}| - \ell + 1]\}$
4: Set $\boldsymbol{x} = 1 \circ \boldsymbol{v} \circ \mathcal{E}_v(\boldsymbol{x})$
5: **end while**
6: Return $\mathrm{Pref}_n(\boldsymbol{x} \circ \boldsymbol{s} \circ 1^n)$

**Theorem 14.** *Algorithm 2 successfully outputs a uniquely decodable $\ell$-tuples covering sequence of length $n$. The time complexity of the algorithm and its decoder is $\mathcal{O}\left(\frac{n^2}{\log n \log\log n}\right)$.*

In order to decode $\boldsymbol{w} \in \Sigma^{n-1}$ given $\boldsymbol{x}$, an output of Algorithm 2, we iteratively inverse the operation of the while loop using the set of decoders $\{\mathcal{D}_v \mid \boldsymbol{v} \in \Sigma^{\ell}\}$. As long as $x_0 = 1$, we repeatedly extract $\boldsymbol{v} = \boldsymbol{x}_{1,\ell}$ and decode the rest of $\boldsymbol{x}$ using $\mathcal{D}_v$. This process ends when $x_0 = 0$, where the decoder returns $\boldsymbol{w} = \mathrm{Pref}_{n-1}(\boldsymbol{x})$.

Note that as a result of the possible concatenation of $\boldsymbol{s}$ and $1^n$ to $\boldsymbol{x}$ at Step 6, in some cases the values of $\boldsymbol{x}$ in the matching iterations of the encoder and the decoder are not equal. In those cases, the decoded sequence contains an additional suffix and $\mathcal{D}_v$ can be invoked on segments that were not outputs of $\mathcal{E}_v$ in Algorithm 2. However, this does not impact the correctness of the decoder since those segments will be trimmed from $\boldsymbol{x}$ at the end of the decoding process.

**Remark 2.** Algorithm 2 can be generalized to any $q > 2$ using a $q$-ary generalization of the compressor $\mathcal{E}_v$. In this case, the algorithm can encode $q$-ary $\ell$-tuples covering sequences of length $\ell \leqslant \log_q n - \log_q \log_q n - 4$ for $n$ large enough.

### *D. Upper Bound Using de Bruijn Graph*

In this section we use an enumeration technique which was first used to enumerate de Bruijn sequences using the de Bruijn graph [16]. It was recently used to enumerate another generalization of de Bruijn sequences [14]. In this paper, this technique is used to derive an upper bound on the cardinality of $\ell$-tuple covering sequences. For simplicity, we focus on binary sequences, although this technique can be extended to any alphabet of finite size.

Let $n = 2^{\ell} + \ell - 1 + k$, for $k = f(n)$. For every selection of $k$ $\ell$-tuples (with repetitions) we construct a graph $G$ which is a generalization of the de Bruijn graph $G_\ell$. The vertices of each graph are the same vertices of $G_\ell$ which are represented by the binary $(\ell-1)$-tuples. The edges are the edges of $G_\ell$ with additional parallel edges corresponding to each of the $k$ $\ell$-tuples picked. There are $\binom{2^{\ell}+k-1}{k}$ different graphs which are generated this way.

A *reverse spanning tree* $T$ of a generalized graph $G$ is a graph whose underlying graph is a tree rooted at some $\boldsymbol{r} \in \Sigma^{\ell-1}$, it contains all the vertices of $G$, and there is a unique directed path from each vertex $v$ of $T$ to $\boldsymbol{r}$. Clearly, all the generalized graphs share the same set of reverse spanning trees as $G_\ell$, and there are $2^{2^{\ell-1}-\ell}$ such trees for each $\boldsymbol{r} \in \Sigma^{\ell-1}$ [16].

For each graph $G$, reverse spanning tree $T$ and root vertex $\boldsymbol{r}$, we use a nondeterministic algorithm to attempt traversing all the edges of $G$ exactly once, starting from the root vertex $\boldsymbol{r}$. In order to ensure the uniqueness of the path with respect to $T$, its edges are notated as *starred*, and the algorithm leaves a vertex $v$ on a starred edge only if it is the last outgoing edge of $v$ that was not traversed before (this algorithm is defined formally in [14], we omit its formal definition due to the lack of space). If the algorithm succeeded traversing all the edges of $G$, the result is a path that corresponds to an $\ell$-tuples covering sequence of length $n$. Notice that for some graphs, the algorithm might fail to traverse all the edges and to generate sequences. However, all the sequences of $\mathcal{R}(n,\ell)$ are produced by this method and thus enumerating the paths constructed by such algorithm derives an upper bound for $r(n,\ell)$.

**Lemma 15.** *For each reverse spanning tree, at most $2^{k+\ell}\binom{2^{\ell}+k-1}{k}$ distinct acyclic sequences are constructed by the algorithm.*

**Corollary 16.** *The total number of distinct acyclic sequences of length $n$ formed by the algorithm is at most $2^{2^{\ell-1}+k}\binom{2^{\ell}+k-1}{k}$.*

The value presented in Corollary 16 provides an upper bound on $r(n,\ell)$. Next, we derive an upper bound for the asymptotic rate of $\ell$-tuples covering sequences, for different values of $\ell$.

**Theorem 17.** *If $n = 2^{\ell} + \ell - 1 + f(n)$ where $f(n) = o(2^{\ell})$, then the asymptotic rate of $\ell$-tuples covering sequences satisfies*

$$\mathbb{R}(\ell) = \frac{1}{2}.$$

**Theorem 18.** *If $n = 2^{\ell} + \ell - 1 + f(n)$ where $f(n) = \alpha 2^{\ell} + o(2^{\ell})$ for $\alpha > 0$, then the asymptotic rate of $\ell$-tuples covering sequences satisfies*

$$\mathbb{R}(\ell) \leqslant H\left(\frac{\alpha}{\alpha+1}\right) + \frac{2\alpha+1}{2\alpha+2}.$$

Note that the result of Theorem 18 is useful only for small values of $\alpha$ in the range $0 < \alpha < 1$. Other values of $\alpha$ are subject for future research.

Table I summarizes the results presented in this paper regarding the asymptotic rate of $\ell$-tuples covering sequences.

TABLE I
ASYMPTOTIC RATE OF BINARY $\ell$- TUPLES COVERING SEQUENCES
$n = 2^{\ell} + \ell - 1 + f(n)$

| Case | Result |
|---|---|
| $f(n) = o(2^{\ell})$ | $\mathbb{R}(\ell) = 1/2$ |
| $f(n) = \alpha 2^{\ell} + o(2^{\ell})$ , $\alpha > 0$ | $\frac{2\alpha+1}{2\alpha+2} \leqslant \mathbb{R}(\ell) \leqslant \min\left\{H(\frac{\alpha}{\alpha+1}) + \frac{2\alpha+1}{2\alpha+2}, 1\right\}$ |
| $f(n) = \omega(2^{\ell})$ | $\mathbb{R}(\ell) = 1$ |

## REFERENCES


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

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

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

[4] Y. M. Chee, T. Etzion, H. M. Kiah, S. Marcovich, A. Vardy, V. Khu Vu, and E. Yaakobi, "Locally-constrained de Bruijn codes: Properties, enumeration, code constructions, and applications," *IEEE Transactions on Information Theory*, vol. 67, no. 12, pp. 7857–7875, 2021.

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

[6] O. Elishco, R. Gabrys, E. Yaakobi, and M. Mèdard, "Repeat-free codes," *IEEE Transactions on Information Theory*, vol. 67, no. 9, pp. 5749–5764, 2021.

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

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

[9] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded sequences from multiset substring spectra," in *Proc. of the IEEE International Symposium on Information Theory*, Vail, Colorado, USA, 2018, pp. 2540–2544.

[10] S. W. Golomb, *Shift register sequences*. World Scientific, Singapore, 2017.

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

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

[13] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3671–3691, 2019.

[14] S. Marcovich, T. Etzion, and E. Yaakobi, "Balanced de Bruijn sequences," in *Proc. of the IEEE International Symposium on Information Theory*, Melbourne, Australia, 2021, pp. 1528–1533.

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

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

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

[18] K. Schouhamer Immink, *Coding techniques for digital recorders*. Prentice-Hall, 1991.

[19] L. Song, F. Geng, Z. Gong, B. Li, and Y. Yuan, "Robust data storage in DNA by de Bruijn graph-based decoding," 2020, [online]. Available: bioRxiv:423642.

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