ELSEVIER Discrete Mathematics 138 (1995) 327–341

DISCRETE MATHEMATICS

Storage efficient decoding for a class of binary be Bruijn sequences

K.G. Paterson*, M.J.B. Robshaw[1]

Department of Mathematics, Royal Holloway, University of London, Egham, Surrey, TW20 OEX, UK

Received 7 July 1993; revised 5 April 1994

Abstract

A binary span k de Bruijn sequence is a binary sequence of period 2^k such that each k-tuple of bits occurs exactly once as a subsequence in a period of the sequence. The de Bruijn sequences have applications in position sensing and range finding, as well as in other areas, by virtue of this subsequence property. To make use of a particular de Bruijn sequence, we need to be able to determine the position of an arbitrary k-tuple in the sequence. We refer to this as 'decoding' the sequence. In this paper we use the structure of a class of de Bruijn sequences, originally constructed by Lempel, to give an algorithm for decoding that is fast, yet more space efficient than a complete look-up table of subsequences and their positions in a sequence. We also consider the case where the displacement between consecutive subsequences whose positions are decoded is restricted and obtain a further marked improvement in storage.

Keywords: de Bruijn sequence; Position sensing

1. Introduction

A binary sequence $(s) = \ldots s_{-1}, s_0, s_1, \ldots$ is said to have period p if p is the least positive integer such that for any integer i, $s_{i+p} = s_i$. Any p consecutive bits of (s) is said to be a generating period of the sequence and is denoted by $[s_i, s_{i+1}, \ldots, s_{i+p-1}]$. Any k successive bits is known as a k-tuple or a k-window. If a sequence of period p has the property that each k-tuple, starting at each of the p possible starting points in the same generating period of (s), is distinct, then (s) is said to have the window property for window size k. A class of sequences with the size k window property are the well-known span k de Bruijn sequences which are typically obtained as the output of a k-stage non-linear feedback shift register [5].

*Corresponding author. Present address: Institute for Signal and Information Processing, Swiss Federal Institute of Technology, Zurich, ISI ETZ F90, ETH Zentrum, CH-8092, Zurich, Switzerland. E-mail: paterson@isi.ee.ethz.ch.

[1] Present address: RSA Laboratories, 100 Marine Parkway, Redwood City, CA 94117, USA.

A sequence (s) with period p possessing the window property for window size k clearly has the property that each k-window uniquely identifies one out of p positions in a generating period of (s). Consequently, various classes of sequences with the window property have been proposed for use in position sensing applications (see, for example, [1, 2, 7, 8]). In a typical application of this type, a sequence is encoded in some way onto a surface. Then a device capable of examining a window of the appropriate size, such as an industrial automaton or an electronic pen, can in principal determine its position on the surface. Notice that the span k de Bruijn sequences are optimal for such applications in the sense that their period is maximal amongst sequences having the window property for window size k.

We refer to the process of decducing the position of an arbitrary window in the sequence as *decoding* the sequence. The success of any application depends on finding an efficient procedure for decoding. Two obvious techniques suggest themselves for de Bruijn sequences. Firstly, there is the 'brute force' method of storing a complete look-up table of the positions of k-tuples. Secondly, we could generate the sequence by clocking the corresponding feedback shift register from the given k-tuple until the register contains a k-tuple whose position is known; the position of the k-tuple of interest is then given by a simple subtraction. By analogy with [7, 8], we could combine these schemes to produce a hybrid scheme consisting of a set of 'milestone' tuples whose equally spaced positions are known; a k-tuple to be decoded is loaded into the register which is then clocked until its contents are equal to one of the milestone tuples.

In this paper we present an alternative solution. We consider a class of binary de Bruijn sequences [6] and show how the structure inherent in their construction can be exploited in a simple way to produce a recursive decoding algorithm. We show that this solution uses significantly less storage than the straightforward look-up table, yet is still fast. We then consider the problem when the movement between decodings is limited. By this we mean that the distance between the current position of the device and the previous position is limited to less than some value. Such a restriction will occur quite naturally in many applications. With this restriction in place we show how certain de Bruijn sequences can be decoded with reduced storage requirements.

2. Notation and definitions

Throughout we follow the notation of Lempel [6]. Additionally we write the k-tuples $(0, \ldots, 0)$ as 0^k and $(1, \ldots, 1)$ as 1^k. Since the tuples 0^k and 1^k occur only once in a period of the de Bruijn sequence the k-tuple preceding 0^k is $(1, 0, \ldots, 0)$ and the k-tuple following 0^k is $(0, \ldots, 0, 1)$. Similar results hold for the k-tuple 1^k.

The kth order de Bruijn graph G_k is a directed graph with 2^k vertices, labelled by the binary k-tuples. Throughout we denote a vertex of G_k in bold type and the components of the vertex in plain type. Vertices $\boldsymbol{x} = (x_1, \ldots, x_k)$ and $\boldsymbol{y} = (y_1, \ldots, y_k)$ of G_k are joined by an edge from $\boldsymbol{x}$ to $\boldsymbol{y}$ if $(x_2, \ldots, x_k) = (y_1, \ldots, y_{k-1})$. We write $\boldsymbol{x} \rightarrow \boldsymbol{y}$ and we say that y is a *successor* of x; each vertex has two successors. Similarly, x is called a *predecessor* of y; each vertex has two predecessors.

We define the *conjugate* of x by $\hat{x}=(\bar{x}_1,x_2,\ldots,x_k)$ and the *dual* of x, by $\bar{x}=(\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_k)$ where $\bar{x}_i$ denotes the binary complement of x_i. We also define the *companion* vertex of x to be the vertex $x'=(x_1,x_2,\ldots,x_{k-1},\bar{x}_k)$. Note that $x\to y$ if and only if $\hat{x}\to y$.

A *t-cycle* in G_k is a sequence $x^{(1)},x^{(2)},\ldots,x^{(t)}$ of distinct vertices of G_k with the property that $x^{(i)}\to x^{(i+1)}$ for $1\leqslant i<t$ and $x^{(t)}\to x^{(1)}$. Clearly a t-cycle $x^{(1)},x^{(2)},\ldots,x^{(t)}$ may be represented by a binary sequence of period t with generating period $[p_1,\ldots,p_t]$, where p_i is the first component of the ith vertex of the cycle. It is also clear that a span k de Bruijn sequence corresponds to a Hamiltonian cycle in G_k. Such a cycle will be called a de Bruijn cycle. The *weight* of a t-cycle $C=[p_1,\ldots,p_t]$ is defined to be the number of non-zero digits among the p_i's.

If C is a t-cycle in G_k with vertices $x^{(1)},x^{(2)},\ldots,x^{(t)}$, then the *distance* from $x^{(i)}$ to $x^{(j)}$ in C is the integer d such that $d=j-i \bmod t$; equivalently the distance is the number of edges of G_k traversed in going from x to y in C. The *dual* cycle $\bar{C}$ has vertices $\bar{x}^{(1)},\ldots,\bar{x}^{(t)}$ and a cycle is said to be *self-dual* if $\bar{C}$ is identical to the cycle C starting from some other vertex. A cycle is said to be *primitive* if C and $\bar{C}$ are vertex disjoint. Finally, two cycles C_1,C_2 are said to be *adjacent* if they are vertex disjoint and there exists a vertex x in C_1 whose conjugate $\hat{x}$ lies in C_2.

Result 2.1 (Golomb [5]). *Two adjacent cycles C_1,C_2 with x in C_1 and $\hat{x}$ in C_2 are joined into a single cycle when the successors of x and $\hat{x}$ are interchanged.*

Let B^k denote the set of binary k-tuples. Define a mapping $\boldsymbol{D}:B^k\to B^{k-1}$ by $\boldsymbol{x}\boldsymbol{D}=(x_1,\ldots,x_k)\ \boldsymbol{D}=(x_1+x_2,\ldots,x_{k-1}+x_k)$. Note that the action of D on tuples in B_k can be easily extended to the action on cycles in G_k. For any $t\geqslant 2$ let $x_t=(0,1,0,1,\ldots)$ be a vertex of G_t.

Lemma 2.2. *For* $1\leqslant i\leqslant s$, $x_{k+i}\boldsymbol{D}=\bar{x}_{k+i}\boldsymbol{D}=1^{k+i-1}$ *and for* $j\geqslant 2$, $x_{k+i}\boldsymbol{D}^j=\bar{x}_{k+i}\boldsymbol{D}^j=0^{k+i-j}$. *We also have* $\hat{x}_{k+i}\boldsymbol{D}=(0,1,\ldots,1)$ *and* $\bar{x}'_{k+i}\boldsymbol{D}=(1,\ldots,1,0)$ *whilst for* $j\geqslant 2$, $\hat{x}_{k+i}\boldsymbol{D}^j=(1,0,\ldots,0)$ *and* $\bar{x}'_{k+i}\boldsymbol{D}^j=(0,\ldots,0,1)$.

Proof. The Lemma follows directly from the definition of $\boldsymbol{D}$. □

Result 2.3 (Lempel [6]). *Given $\boldsymbol{a}=(a_1,\ldots,a_{k-1})\in B^{k-1}$, the pre-images of $\boldsymbol{a}$ under D are*

$$x=\left(x_1,x_1+a_1,x_1+a_1+a_2,\ldots,x_1+\sum_{i=1}^{k-1}a_i\right),\quad \text{where } x_1=0 \text{ or } 1.$$

Result 2.4 (Lempel [6]). *$\boldsymbol{D}$ is a graph homomorphism (called the Lempel homomorphism) from G_k to G_{k-1}. For all $x,y\in B^k$, $xD=yD \Leftrightarrow x=y$* or *$x=\bar{y}$.*

Result 2.5 (Lempel [6]). *There is a 1–1 correspondence between t-cycles Γ of even weight in G_{k-1} and vertex disjoint (or primitive) pairs of dual cycles $C, \bar{C}$ in G_k under which $\Gamma = C\boldsymbol{D} = \bar{C}\boldsymbol{D}$.*

The following construction is to be found in [6], and more explicitly in [4]. Fix $k \geqslant 2$ and let Γ_k be a Hamiltonian cycle in G_k. Clearly, Γ_k has even weight. By Result 2.5, we may consider the $\boldsymbol{D}$-morphic pre-images C_{k+1} and $\bar{C}_{k+1}$ of Γ_k in G_{k+1}. Each has period 2^k and so between them they include all the vertices of G_{k+1}. We determine C_{k+1} by specifying that x_{k+1} lies on C_{k+1}, so that $\bar{x}_{k+1} = (1,0,1,0,\ldots)$ lies on $\bar{C}_{k+1}$.

Note that $x_{k+1} \to \bar{x}_{k+1}$, but x_{k+1} and $\bar{x}_{k+1}$ lie in disjoint cycles. We must have $x_{k+1} \to \bar{x}'_{k+1}$ in C_{k+1} and $\hat{x}_{k+1} \to \bar{x}_{k+1}$ in $\bar{C}_{k+1}$. By Result 2.1 we obtain a Hamiltonian cycle Γ_{k+1} in G_{k+1} (and a corresponding de Bruijn sequence) by interchanging the successors $\bar{x}'_{k+1}$ and $\bar{x}_{k+1}$ of x_{k+1} and $\hat{x}_{k+1}$, respectively.

This process may be repeated to obtain de Bruijn cycles $\Gamma_{k+2}, \Gamma_{k+3}, \ldots$ in $G_{k+2}, G_{k+3}, \ldots$ on joining successors $\bar{x}'_{k+i}$ and $\bar{x}_{k+i}$ of x_{k+i} and $\hat{x}_{k+i}$.

Example 2.6. Choosing $k=3$ and $\Gamma_3 = [11100010]$, we use the above construction and get

$$\Gamma_4 = [0101000011011110],$$

$$\Gamma_5 = [00110000010010101110011111011010],$$

$$\begin{aligned}\Gamma_6 = [&000100000001110011010001010100100111101111111\\ &-0001100101110101101110].\end{aligned}$$

It is easily shown that distinct de Bruijn cycles in G_k lead to distinct de Bruijn cycles in G_{k+s} with this construction. Thus, using the enumeration of de Bruijn sequences (see, for example [4]), we see that for every $n \geqslant k$, $2^{2^{k-1}-k}$ span n binary de Bruijn sequences can be obtained from the repeated use of Lempel's construction starting with span k de Bruijn sequences.

3. A decoding algorithm $\mathscr{D}$ for Lempel's de Bruijn sequences

Suppose Γ_{k+s} is a de Bruijn cycle in G_{k+s} obtained from a de Bruijn cycle Γ_k in G_k by repeated joining of cycles in $G_{k+1}, G_{k+2}, \ldots, G_{k+s}$ as in Section 2.

Since $\boldsymbol{D}$ is a graph homomorphism and is bijective when restricted to either of the cycles C_{k+i} or $\bar{C}_{k+i}$ $(1 \leqslant i \leqslant s)$ with image Γ_{k+i-1}, distances between vertices on C_{k+i} and $\bar{C}_{k+i}$ and their image vertices in Γ_{k+i-1} are preserved. Moreover, the point in G_{k+i} where the joining process on cycles C_{k+i} and $\bar{C}_{k+i}$ is carried out is completely specified. So if we are given an arbitrary $(k+i)$-tuple z in Γ_{k+i}, we can reduce the problem of finding the distance from x_{k+i} to z in Γ_{k+i} to that of finding the distance from $x_{k+i}\boldsymbol{D}$ to $z\boldsymbol{D}$ in Γ_{k+i-1}, together with making an adjustment which depends on the cycle in which z is located. The decoding algorithm we present below repeatedly applies this reduction for $1 \leqslant i \leqslant s$ and uses additional information on the cycles Γ_{k+i} including a complete look-up table for the cycle Γ_k.

For successful implementation of the algorithm we need three sets of information.

(A) First we need to store 'cycle locating' tables L_{k+i} for each Γ_{k+i}, $1 \leqslant i \leqslant s$. The table entry, $c_{k+i}(\boldsymbol{x})$, $1 \leqslant i \leqslant s$, corresponding to a $(k+i)$-tuple $\boldsymbol{x}$ is given by

$$c_{k+i}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in C_{k+i}, \\ 0 & \text{if } \boldsymbol{x} \in \bar{C}_{k+i}. \end{cases}$$

(B) We also need to store a table R of values r_{k+i}, $0 \leqslant i \leqslant s-1$, that denote the distance from 1^{k+i} to $\boldsymbol{x}_{k+i}$ in Γ_{k+i} for each $0 \leqslant i \leqslant s-1$. Later we will find that the r_{k+i} can be represented in a particularly nice form.

(C) Finally, we need to store a look-up table U of distances of k-tuples from $\boldsymbol{x}_k$ in Γ_k.

Decoding algorithm $\mathscr{D}$. Given a $(k+s)$-tuple z in Γ_{k+s}, we obtain the distance from $\boldsymbol{x}_{k+s}$ to z by the following procedure. Denote by u_{k+i}, $0 \leqslant i \leqslant s$, the distance from $\boldsymbol{x}_{k+i}$ to $z\boldsymbol{D}^{(s-i)}$ in Γ_{k+i}.

(i) Evaluate $z\boldsymbol{D}^{(s-i)}$ for $0 \leqslant i \leqslant s$. Using tables L_{k+i} obtain $c_{k+i}(z\boldsymbol{D}^{(s-i)})$ for $1 \leqslant i \leqslant s$, and using table U obtain u_k for $z\boldsymbol{D}^s$.

(ii) The displacement from $\boldsymbol{x}_{k+s}$ to z in Γ_{k+s}, u_{k+s}, is given by the following iterative calculation with u_k obtained from table U.

$$u_{k+i} = 0 \quad \text{if } z\boldsymbol{D}^{(s-i)} = \boldsymbol{x}_{k+i},$$

otherwise

$$u_{k+i} = ((u_{k+i-1} + r_{k+i-1}) \bmod 2^{k+i-1})$$
$$+ c_{k+i}(z\boldsymbol{D}^{(s-i)})2^{k+i-1} + \overline{c_{k+i}(z\boldsymbol{D}^{(s-i)})}.$$

Lemma 3.1. *Decoding algorithm $\mathscr{D}$ provides a successful procedure for obtaining the distance from $\boldsymbol{x}_{k+s}$ to an arbitrary tuple in Γ_{k+s}.*

Proof. For $0 \leqslant i \leqslant s-1$, let r_{k+i} denote the distance from 1^{k+i} to $\boldsymbol{x}_{k+i}$ in Γ_{k+i} and let t_{k+i} denote the distance from 1^{k+i} to $z\boldsymbol{D}^{(s-i)}$ in Γ_{k+i}.

Then, if u_{k+i} denotes the distance from $\boldsymbol{x}_{k+i}$ to $z\boldsymbol{D}^{(s-i)}$ in Γ_{k+i} for $0 \leqslant i \leqslant s$, we have

$$t_{k+i} \equiv u_{k+i} + r_{k+i} \pmod{2^{k+i}}. \tag{1}$$

To solve the decoding problem we are required to find u_{k+s}.

Since, for $1 \leqslant i \leqslant s$, $\boldsymbol{x}_{k+i}\boldsymbol{D} = \bar{\boldsymbol{x}}_{k+i}\boldsymbol{D} = 1^{k+i-1}$, we have that t_{k+i-1} is the distance from $\boldsymbol{x}_{k+i}$ to $z\boldsymbol{D}^{(s-i)}$ in C_{k+i} when $z\boldsymbol{D}^{(s-i)} \in C_{k+i}$ and that t_{k+i-1} is the distance from $\bar{\boldsymbol{x}}_{k+i}$ to $z\boldsymbol{D}^{(s-i)}$ in $\bar{C}_{k+i}$ when $z\boldsymbol{D}^{(s-i)} \in \bar{C}_{k+i}$.

Hence, from the method of joining the cycles C_{k+i} and $\bar{C}_{k+i}$, we have

$$u_{k+i}=\begin{cases}0 & \text{if } z\boldsymbol{D}^{(s-i)}=\boldsymbol{x}_{k+i},\\ t_{k+i-1}+c_{k+i}(z\boldsymbol{D}^{(s-i)})2^{k+i-1}+\overline{c_{k+i}(z\boldsymbol{D}^{(s-i)})} & \text{otherwise.}\end{cases} \tag{2}$$

Using Eqs. (1) and (2) we can obtain the expression for u_{k+i} in terms of u_{k+i-1} given in algorithm $\mathscr{D}$. The correctness of the algorithm follows immediately. □

Performance of algorithm $\mathscr{D}$. The tables U, R and L_{k+i}, $1 \leqslant i \leqslant s$, may be computed in the process of building up the de Bruijn cycle Γ_{k+s} from G_k. The intermediate and final de Bruijn cycles need not themselves be stored.

Storing U the complete look-up table for Γ_k requires $k.2^k$ bits (since each of the 2^k distances to be stored requires k bits). Since z lies in C_{k+i} if and only if its dual $\bar{z}$ lies in $\bar{C}_{k+i}$ we require $\frac{1}{2}2^{k+i}$ bits to store L_{k+i} for $1 \leqslant i \leqslant s$. Storing table R requires $k+i$ bits for each $0 \leqslant i \leqslant s-1$, a total of $\frac{1}{2}s(2k+s-1)$ bits. This is negligible when compared to the other storage requirements and we ignore it in what follows.

The total storage required by algorithm $\mathscr{D}$ is given in bits by

$$2^{k+s}-2^k+k2^k.$$

Writing $k+s=n$ we find that the storage required $T(k,n)$ to decode a span n de Bruijn sequence for which $\mathscr{D}$ is a valid decoding algorithm is now

$$T(k,n)=2^n+(k-1)2^k.$$

We choose a small value of k, $k \geqslant 3$ to reduce the storage requirements. Then for fixed k, $T(k,n)$ has storage requirements that increase as 2^n with increasing n, whereas the storage for the complete look-up table increases as $n2^n$.

Each of the s iterations in algorithm $\mathscr{D}$ requires one look-up in table L_i and one use of table R, along with two additions, one modulo a power of 2. At the last stage, we need one look-up from table U. These operations are extremely quick to implement. Moreover, as the number of levels s of the scheme increase, the number of very simple operations required to successfully decode in Γ_{k+s} grows linearly with s.

4. The restricted scheme and a decoding algorithm $\mathscr{R}$

Suppose that a de Bruijn sequence Γ_{k+s} is constructed using Lempel's iterated construction. Consider successive $(k+s)$-tuples in Γ_{k+s}. Starting with $\bar{\boldsymbol{x}}_{k+s}$, we see that as we traverse Γ_{k+s},the 2^{k+s-1} $(k+s)$-tuples that follow are located in $\bar{C}_{k+s}$, then the next 2^{k+s-1} are located in C_{k+s}. Thus, if we attempt to obtain the positions of $(k+s)$-tuples $\boldsymbol{x}$ and $\boldsymbol{y}$ in Γ_{k+s} that are close, it is unlikely that the values of $c_{k+s}(\boldsymbol{x})$ and $c_{k+s}(\boldsymbol{y})$ are different between successive decodings. From here on we shall slightly abuse the notation and write $c_{k+s}(\boldsymbol{x})$ as c_{k+s} where the context allows.

From Eq. (2), decoding with the two different values of c_{k+s} yields vertices which are distance $2^{k+s-1}-1$ apart. Thus, provided we know the positions of the last decoding and we restrict how far away the next decoding can be, we can determine which of the two possible values obtained using $c_{k+s}=0$ and $c_{k+s}=1$ is correct. As a result, we need no longer store the cycle look-up table for Γ_{k+s}.

We extend this idea and avoid storing cycle locating tables for Γ_{k+m+1} to Γ_{k+s} for some m, $1 \leqslant m \leqslant s-1$. We make a considerable saving in storage but the cost is that we must restrict the possible displacement between successive position decodings.

Let $\boldsymbol{c}=(c_{k+1},\dots,c_{k+s})$ denote the sequence of cycle locating bits obtained using algorithm $\mathscr{D}$ for a particular vertex of G_{k+s}. We shall call this vector the *cycle vector* of a tuple. Such a vector is used in the iterative calculation of algorithm $\mathscr{D}$ to identify the position of each $(k+s)$-tuple in Γ_{k+s}.

We now present the restricted decoding algorithm $\mathscr{R}$. For successful implementation of the algorithm we need five sets of information.

(A) The cycle vector $\boldsymbol{c}=(c_{k+1},\dots,c_{k+s})$ of some tuple z in Γ_{k+s} and the distance u_{k+s} of that tuple from $\boldsymbol{x}_{k+s}$ in Γ_{k+s}.

(B) Cycle locating tables L_{k+i}, $1 \leqslant i \leqslant m$.

(C) A table U of distances of k-tuples from $\boldsymbol{x}_k$ in Γ_k.

(D) A table R of values r_{k+i}, $0 \leqslant i \leqslant s-1$, that denote the distance from $\boldsymbol{x}_{k+i}$ to 1^{k+i} in Γ_{k+i}.

(E) The tuple z' we wish to locate, which must lie within $dist_{\max}=\frac{1}{2}(2^{k+m-1}-3)$ of the tuple z.

Decoding algorithm $\mathscr{R}$. (i) For each $0 \leqslant i \leqslant m$, evaluate $z'\boldsymbol{D}^{(s-i)}$ and c_i' using table L_{k+i}. Evaluate u_k', the distance from $z'\boldsymbol{D}^s$ to $\boldsymbol{x}_k$, using table U.

(ii) Use u_k' and $(c_{k+1}',\dots,c_{k+m}')$ as the cycle vector in the iterative calculation of algorithm $\mathscr{D}$, to evaluate u_{k+m}', the distance from $\boldsymbol{x}_{k+m}$ to $z'\boldsymbol{D}^{(s-m)}$ in Γ_{k+m}.

(iii) Use u_{k+m}' and $(c_{k+m+1},\dots,c_{k+s})$ as the cycle vector in the iterative calculation of algorithm $\mathscr{D}$ to make the first estimate of u_{k+s}', the distance from $\boldsymbol{x}_{k+s}$ to z' in Γ_{k+s}. If $|u_{k+s}-u_{k+s}'|<dist_{\max}$ then u_{k+s}' is the correct displacement of $\mathbf{z}'$ and the algorithm is finished. Otherwise go to iv).

(iv) Repeat the following steps starting with $j=0$.

(a) $j:=j+1$

(b) Using u_{k+m}' and

$$(c_{k+m+1},\dots,c_{k+m+j-1},\bar{c}_{k+m+j},c_{k+m+j+1},\dots,c_{k+s})$$

as the cycle vector, make the estimate of u_{k+s}' using the iterative calculation in $\mathscr{D}$.

(c) If $|u_{k+s}-u_{k+s}'|<dist_{\max}$ then u_{k+s}' is the correct displacement of z', otherwise return to a).

Performance of algorithm $\mathscr{R}$. The main benefit of using decoding algorithm $\mathscr{R}$ is that we can dispense with cycle locating tables $L_{k+m+1},\dots,L_{k+s}$ for cycles $\Gamma_{k+m+1},\dots,\Gamma_{k+s}$. Here m is fixed with $1 \leqslant m \leqslant s-1$. We need only store cycle locating tables $L_{k+1},\dots,L_{k+m}$ for $\Gamma_{k+1},\dots,\Gamma_{k+m}$ and table U. Once again, there is a small additional amount of storage required for table R and in the execution of the algorithm which we ignore.

The final storage requirement of the restricted algorithm $\mathscr{R}$ in bits is

$$2^{k+m}-2^k+k2^k.$$

Writing $k+s=n$ we obtain

$$T_{\text{rest}}(k,n,m)=2^{n-(s-m)}+(k-1)2^k,$$

which we can compare to the storage requirement for algorithm $\mathscr{D}$

$$T(k,n)=2^n+(k-1)2^k.$$

For a fixed small k, we see that the restricted decoding algorithm requires a factor of 2^{s-m} less storage than our first algorithm. Note, however, that to implement the restricted decoding algorithm we must be willing to limit the distance that we move between successive decodings to less than

$$dist_{\max}=\frac{2^{k+m-1}-3}{2}.$$

We are now in a position to fully compare the storage requirements of each decoding algorithm, $\mathscr{D}$ and $\mathscr{R}$ and the requirements for the complete look-up table.

Table 1 presents the storage required by algorithm $\mathscr{D}$ to decode a span n de Bruijn sequence obtained by the repeated use of Lempel's construction beginning with a span k binary de Bruijn sequence, for a variety of values of k and n. The final row of the table gives the storage requirements of a complete look-up table for a span n binary de Bruijn sequence, while the final column give the number of distinct de Bruijn sequences that can be constructed in this way.

Table 2 presents the storage requirements of algorithm $\mathscr{R}$ to decode any one of the 2^{120} span n de Bruijn sequences that can be obtained by the repeated use of Lempel's construction beginning with a span 8 de Bruijn sequence. The values of $T_{\text{rest}}(k,n,m)$ are compared for various n and m $(k=8)$. The final column gives the maximum distance between successive decodings for algorithm $\mathscr{R}$ to be successful; this is dependent on k and m but independent of n. The final row presents the previous best

Table 1

$k\backslash n$	5	10	15	20	15	30	# generated
3	40	$\sim 2^{10}$	$\sim 2^{15}$	$\sim 2^{20}$	$\sim 2^{25}$	$\sim 2^{30}$	2
5	—	$\sim 2^{10.2}$	$\sim 2^{15}$	$\sim 2^{20}$	$\sim 2^{25}$	$\sim 2^{30}$	2048
8	—	$\sim 2^{11.5}$	$\sim 2^{15}$	$\sim 2^{20}$	$\sim 2^{25}$	$\sim 2^{30}$	2^{120}
12	—	—	$\sim 2^{16.2}$	$\sim 2^{20}$	$\sim 2^{25}$	$\sim 2^{30}$	$2^{2^{11}-12}$
15	—	—	—	$\sim 2^{20.5}$	$\sim 2^{25}$	$\sim 2^{30}$	$2^{2^{14}-15}$
Look-up table	160	$\sim 2^{13}$	$\sim 2^{19}$	$\sim 2^{24}$	$\sim 2^{29.6}$	$\sim 2^{35}$	

Table 2

$m\backslash n$	10	15	20	25	30	$dist_{\max}$
1	$\sim 2^{11.2}$	$\sim 2^{11.2}$	$\sim 2^{11.2}$	$\sim 2^{11.2}$	$\sim 2^{11.2}$	126
5	—	$\sim 2^{13.3}$	$\sim 2^{13.3}$	$\sim 2^{13.3}$	$\sim 2^{13.3}$	$\sim 2^{11}$
8	—	—	$\sim 2^{16}$	$\sim 2^{16}$	$\sim 2^{16}$	$\sim 2^{14}$
10	—	—	$\sim 2^{18}$	$\sim 2^{18}$	$\sim 2^{18}$	$\sim 2^{16}$
12	—	—	—	$\sim 2^{20}$	$\sim 2^{20}$	$\sim 2^{18}$
Previous best storage	$\sim 2^{11.5}$	$\sim 2^{15}$	$\sim 2^{20}$	$\sim 2^{25}$	$\sim 2^{30}$	

storage required to decode a span n de Bruijn sequence with $k=8$ obtained from Table 1.

For each decoding using algorithm $\mathscr{R}$ we make one calculation of u'_{k+m} using the bits $c'_{k+1},\ldots,c'_{k+m}$, perform one look up in the table U for Γ_k and, for some of the $s-m+1$ candidate cycle vectors, we make one partial application of the iterative calculation of decoding algorithm $\mathscr{D}$. Clearly, algorithm $\mathscr{R}$ will be slower than algorithm $\mathscr{D}$ but as soon as a candidate cycle vector is shown to give a position consistent with the restriction, the algorithm terminates. In fact it is possible to show that the first candidate cycle vector tested in step (iv) of $\mathscr{R}$ is twice as likely to lead to the correct position for z' than the second, which is twice as likely as the third and so on. It is then simple to show that we expect to have to make the full decoding calculation of algorithm $\mathscr{D}$ less than three times in order to find the correct position of the tuple z'.

Successful implementation of decoding algorithm $\mathscr{R}$ relies on knowledge of the full decoding of some nearby tuple. The question arises as to how a practical system based on algorithm $\mathscr{R}$ can be initialised. Two suggestions are as follows. First, it might be possible to set reference starting tuples in the sequence for which the full decoding is known. At the start of a position sensing session, a device is set off from a physical starting position which corresponds to one of these starting tuples. The second suggestion is merely to store the full decoding of the final position at the end of the session. When the system is re-started, the device will begin from a position for which the full decoding is stored. Depending on the implementation, there may well be other techniques which can be used to surmount the problem of starting the system.

5. Validity of decoding algorithm $\mathscr{R}$

This section is rather technical and proves the validity of algorithm $\mathscr{R}$; it might be omitted at a first reading.

Suppose two vertices $\boldsymbol{a}$ and $\boldsymbol{b}$ in Γ_{k+s} have cycle vectors that differ in at least one position. Then there is some i, $1\leqslant i\leqslant s$, such that $\boldsymbol{a}\boldsymbol{D}^{(s-i)}$ and $\boldsymbol{b}\boldsymbol{D}^{(s-i)}$ lie on different cycles in Γ_{k+i}. If the pair of vertices $\boldsymbol{a}$ and $\boldsymbol{b}$ are adjacent in Γ_{k+s} then we say that they induce a *cycle change in* Γ_{k+i}. The following Lemma is a direct consequence of Lempel's construction.

Lemma 5.1. *Suppose two consecutive vertices $\boldsymbol{a}, \boldsymbol{b} \in \Gamma_{k+s}$ induce a cycle change in Γ_{k+i}. Then the corresponding vertices in Γ_{k+i}, $\boldsymbol{a}\boldsymbol{D}^{(s-i)}$ and $\boldsymbol{b}\boldsymbol{D}^{(s-i)}$, which lie on different cycles, are either $\boldsymbol{x}_{k+i}, \bar{\boldsymbol{x}}_{k+i}$ in going from C_{k+i} to $\bar{C}_{k+i}$ or $\hat{\boldsymbol{x}}_{k+i}, \bar{\boldsymbol{x}}'_{k+i}$ in going from $\bar{C}_{k+i}$ to C_{k+i}.*

Suppose the two vertices $\boldsymbol{a}$ and $\boldsymbol{b}$ in the above lemma have cycle vectors $c_1 = (c^1_{k+1}, \ldots, c^1_{k+s})$ *and* $\boldsymbol{c}_2 = (c^2_{k+1}, \ldots, c^2_{k+s})$, respectively. It is straightforward to establish that for $1 \leqslant j \leqslant s, j \neq i$, $c^1_{k+j} = c^2_{k+j}$ and $c^1_{k+i} \neq c^2_{k+i}$; that is, the cycle vectors of $\boldsymbol{a}$ and $\boldsymbol{b}$ differ in exactly one position.

We now introduce some convenient notation. Define z_{k+i} as the distance from 1^{k+i} to 0^{k+i} in Γ_{k+i}. Let r_{k+i} be, as before, the distance from 1^{k+i} to x_{k+i} in Γ_{k+i}.

Lemma 5.2. *For* $1 \leqslant i \leqslant s$,

$$z_{k+i} = \begin{cases} 2^{k+i-1}+1 & \text{if } 1^{k+i} \in C_{k+i}, \\ 2^{k+i-1}-1 & \text{otherwise} \end{cases}$$

and

$$r_{k+i} = \begin{cases} 2^{k+i-1} - z_{k+i-1} & \text{if } 1^{k+i} \in C_{k+i}, \\ 2^{k+i} - z_{k+i-1} - 1 & \text{otherwise.} \end{cases}$$

Proof. Since 1^{k+i} and 0^{k+i} have the same image under $\boldsymbol{D}$, namely 0^{k+i-1}, we have that 1^{k+i} and 0^{k+i} lie the same distance apart from $\boldsymbol{x}_{k+i}, \bar{\boldsymbol{x}}_{k+i}$ in C_{k+i} and $\bar{C}_{k+i}$, respectively. Because of the method of joining C_{k+i} and $\bar{C}_{k+i}$ to form Γ_{k+i}, we obtain the desired result for z_{k+i}.

To obtain the result for r_{k+i}, we attempt to apply one step of the algorithm $\mathscr{D}$ for decoding in Γ_{k+i}. We have that $1^{k+i}\boldsymbol{D} = 0^{k+i-1}$ and $\boldsymbol{x}_{k+i}\boldsymbol{D} = 1^{k+i-1}$ and the distance from 0^{k+i-1} to 1^{k+i-1} in Γ_{k+i-1} is $2^{k+i-1} - z_{k+i-1}$. When $1^{k+i} \in C_{k+i}$, each of the vertices passed through in moving from 1^{k+i} to x_{k+i} in Γ_{k+i} lie in C_{k+i} and the result follows. When $1^{k+i} \in \bar{C}_{k+i}$, $2^{k+i-1} - z_{k+i-1}$ is the distance from 1^{k+i} to $\bar{x}_{k+i}$ in $\bar{C}_{k+i}$. Then the distance from 1^{k+i} to the predecessor of $\bar{x}_{k+i}$ is $2^{k+i-1} - z_{k+i-1} - 1$ and so the distance from 1^{k+i} to x_{k+i} in Γ_{k+i} is $2^{k+i-1} - z_{k+i-1} - 1 + 2^{k+i-1}$ which gives the stated value. $\square$

If two pairs of consecutive vertices $\boldsymbol{a}_1, \boldsymbol{b}_1$ and $\boldsymbol{a}_2, \boldsymbol{b}_2$ in Γ_{k+s} induce cycle changes, then the distance from the cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ to the cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is defined to be the distance in Γ_{k+s} from $\boldsymbol{b}_1$ to $\boldsymbol{b}_2$. Given some m, $1 \leqslant m \leqslant s-1$, we can calculate the minimum distance in Γ_{k+s} from one cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ induced in Γ_{k+i} to another cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ induced in $\Gamma_{k+j}, m+1 \leqslant i \leqslant j \leqslant s$, by making the following observations.

If two cycle changes have this minimum distance then we may assume that the distance from $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ to $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is less than or equal to the distance from $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ to $(\boldsymbol{a}_1, \boldsymbol{b}_1)$. We may also assume that no other pair of consecutive vertices encountered in going from $\boldsymbol{a}_1$ to $\boldsymbol{a}_2$ in Γ_{k+s} induce a cycle change in some Γ_{k+i} with $m+1 \leqslant i \leqslant s$.

Otherwise, we would obtain a smaller distance from one cycle change to another, contradicting the original assumption that the distance from $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ to $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ was minimal.

With these assumptions in mind, we consider the three cases where our pairs of cycle changes leading to a minimum distance are induced in Γ_{k+i} and Γ_{k+j} with $i=j, i=j-1$ and $i \leqslant j-2$. The bounds we obtain are slightly different in each case.

Lemma 5.3. *Consider the situation where a pair of cycle changes* $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ *and* $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ *are induced in* Γ_{k+i}, $m+1 \leqslant i \leqslant s$, *and the distance from the first change to the second change is minimal. Then the distance in* Γ_{k+s} *from* $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ *to* $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ *is at least* 2^{k+i}.

Proof. With the assumption that no other cycle changes are induced in going from the first cycle change to the second, it is easily seen that in Γ_{k+i} we simply have 2^{k+i} consecutive vertices of either C_{k+i} or $\bar{C}_{k+i}$ in going from $\boldsymbol{b}_1\boldsymbol{D}^{s-i}$ to $\boldsymbol{b}_2\boldsymbol{D}^{s-i}$. □

Lemma 5.4. *Consider the situation where cycle changes* $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ *and* $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ *are induced in* Γ_{k+i} *and* Γ_{k+i+1}, $m+1 \leqslant i \leqslant s-1$ *and where the distance from the first change to the second change is the minimal distance between any pair of cycle changes. Then this distance is at least* $2^{k+i-2}-2$.

Proof. There are eight cases to consider, generated by the following three independent choices. Firstly, either the change in Γ_{k+i+1} is from C_{k+i+1} to $\bar{C}_{k+i+1}$ or it is from $\bar{C}_{k+i+1}$ to C_{k+i+1}. Secondly, the change in Γ_{k+i} is from C_{k+i} to $\bar{C}_{k+i}$ or it is from $\bar{C}_{k+i}$ to C_{k+i}. Finally, the cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ is induced in Γ_{k+i+1} and the cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is induced in Γ_{k+i} or vice-versa.

The arguments are similar in each case so, as an example, we consider in detail the situation where the change in Γ_{k+i+1} is from C_{k+i+1} to $\bar{C}_{k+i+1}$, the change in Γ_{k+i} is from C_{k+i} to $\bar{C}_{k+i}$, the cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ is induced in Γ_{k+i+1} and the cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is induced in Γ_{k+i}.

At the cycle change in Γ_{k+i+1} we have vertex $\boldsymbol{x}_{k+i+1}$ follow by $\bar{\boldsymbol{x}}_{k+i+1}$, by Lemma 5.1, each being mapped to 1^{k+i} under $\boldsymbol{D}$. With the assumption that no other cycle changes are encountered in going from $\boldsymbol{a}_1$ to $\boldsymbol{a}_2$, we see that there must be an uninterrupted sequence of consecutive vertices of Γ_{k+i} from 1^{k+i} to the cycle change of interest in Γ_{k+i}. Since the cycle change induced in Γ_{k+i} is also of type C_{k+i} to $\bar{C}_{k+i}$, 1^{k+i} lies in C_{k+i} and the distance between our cycle changes is the distance from 1^{k+i} to $\bar{\boldsymbol{x}}_{k+i}$ in Γ_{k+i}, that is $r_{k+i}+1$. Using Lemma 5.2, we can bound this distance below by 2^{k+i-2}.

In the other cases, where the cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ is induced in Γ_{k+i+1} and the cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is induced in Γ_{k+i}, we need to consider distances between pairs of vertices 1^{k+i} and $\bar{\boldsymbol{x}}'_{k+i}, \bar{\boldsymbol{x}}'_{k+i+1}\boldsymbol{D}$ and $\bar{\boldsymbol{x}}_{k+i}$, and $\bar{\boldsymbol{x}}'_{k+i+1}\boldsymbol{D}$ and $\bar{\boldsymbol{x}}'_{k+i}$. Using the known adjacencies between such vertices, 0^{k+i} and 1^{k+i}, and the values of r_{k+i} and z_{k+i} from Lemma 5.2, the stated bound may be verified. The corresponding arguments are similar when the cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ is induced in Γ_{k+i} and the cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is induced in Γ_{k+i+1}. We omit the details. □

Lemma 5.5. *Consider the situation where cycle changes $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ and $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ are induced in Γ_{k+i} and Γ_{k+j} with $m+1 \leqslant i \leqslant j-2$ and $j \leqslant s$ and where the distance from the first change to the second change is the minimal distance between any pair of cycle changes. Then this distance is at least $2^{k+i-2}-3$.*

Proof. Again, there are eight cases to consider, arising from the same three independent choices. We consider in detail the case when the change in Γ_{k+j} is from C_{k+j} to $\bar{C}_{k+j}$, the change in Γ_{k+i} is from C_{k+i} to $\bar{C}_{k+i}$, the cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ is induced in Γ_{k+j} and the cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is induced in Γ_{k+i}.

At the cycle change of interest in Γ_{k+j} we have vertex $\boldsymbol{x}_{k+j}$ followed by $\bar{\boldsymbol{x}}_{k+j}$, each having image 0^{k+i} under $\boldsymbol{D}^{(j-i)}$. With the assumption that there are no intermediate cycle changes between our pair, we see that there must be an uninterrupted sequence of consecutive vertices of Γ_{k+i} in going from 0^{k+i} to the cycle change of interest in Γ_{k+i}. Since the cycle change induced in Γ_{k+i} is also of type C_{k+i} to $\bar{C}_{k+i}$, 0^{k+i} lies in C_{k+i} and the distance between our cycle changes is the distance from 0^{k+i} to $\bar{\boldsymbol{x}}_{k+i}$ in Γ_{k+i}. Using Lemma 5.2, we can bound this distance below by $2^{k+i-2}-2$.

In the other cases where the cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ is induced in Γ_{k+j} and the cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is induced in Γ_{k+i}, we need to consider distances between the pairs of vertices 0^{k+i} and $\hat{\boldsymbol{x}}_{k+i}, \bar{\boldsymbol{x}}'_{k+i+1}\boldsymbol{D}^{(j-i)}$ and $\hat{\boldsymbol{x}}_{k+i}$, and $\bar{\boldsymbol{x}}'_{k+i+1}\boldsymbol{D}^{(j-i)}$ and $\bar{\boldsymbol{x}}'_{k+i}$. Using the known adjacencies between such vertices, 0^{k+i} and 1^{k+1}, and the values of r_{k+i} and z_{k+i} from Lemma 5.2, the stated bound may again be verified.The corresponding arguments are similar when the cycle change $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ is induced in Γ_{k+i} and the cycle change $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ is induced in Γ_{k+j}. Again we omit the details of these cases. □

Theorem 5.6. *Suppose $1 \leqslant m \leqslant s-1$. Then the distance from a cycle change induced in Γ_{k+i} to a cycle change induced in Γ_{k+j} with $m+1 \leqslant i, j \leqslant s$ is at least $2^{k+m-1}-3$.*

Proof. Follows directly from previous Lemmas. □

Lemma 5.7. *Let z be a vertex of G_{k+s} having cycle vector $\boldsymbol{c} = (c_{k+1}, \ldots, c_{k+s})$ and suppose z' is the vertex of G_{k+s} satisfying $z\boldsymbol{D}^{(s-k)} = z'\boldsymbol{D}^{(s-k)}$ with cycle vector $\boldsymbol{c}'$ obtained by changing the ith component, $1 \leqslant i \leqslant s$, of $\boldsymbol{c}$. Then the distance from vertex z to z' in Γ_{k+s} is at least $2^{k+i-1}-1$.*

Proof. As in Section 4, let u_{k+j} denote the distance from $\boldsymbol{x}_{k+j}$ to $z\boldsymbol{D}^{(s-j)}$ and let u'_{k+j} denote the distance from $\boldsymbol{x}_{k+j}$ to $z'\boldsymbol{D}^{(s-j)}$ in Γ_{k+j}. Then since $z\boldsymbol{D}^{(s-k)} = z'\boldsymbol{D}^{(s-k)}$ and both z and z' have cycle vectors agreeing in the first $i-1$ positions, we have $u_{k+i-1} = u'_{k+i-1}$. Thus, on using the iterated expression for u_{k+s} of algorithm $\mathscr{D}$, we have

$$\begin{aligned} u_{k+s} = (\cdots (u_{k+i-1} &+ R(k+i-1) \pmod{2^{k+i-1}}) \\ &+ c_{k+i} 2^{k+i-1} + \bar{c}_{k+i} + R(k+i) \pmod{2^{k+i}}) \\ &\vdots \\ &+ c_{k+s-1} 2^{k+s-2} + \bar{c}_{k+s-1} + R(k+s-1) \pmod{2^{k+s-1}}) \\ &+ c_{k+s} 2^{k+s-1} + \bar{c}_{k+s} \end{aligned}$$

and

$$\begin{aligned} u'_{k+s} = (\cdots (u_{k+i-1} &+ R(k+i-1) \pmod{2^{k+i-1}}) \\ &+ \bar{c}_{k+i} 2^{k+i-1} + c_{k+i} + R(k+i) \pmod{2^{k+i}}) \\ &\vdots \\ &+ c_{k+s-1} 2^{k+s-2} + \bar{c}_{k+s-1} + R(k+s-1) \pmod{2^{k+s-1}}) \\ &+ c_{k+s} 2^{k+s-1} + \bar{c}_{k+s}. \end{aligned}$$

The expressions for u_{k+s} and u'_{k+s} differ by at least $2^{k+i-1}-1$ and the result follows. □

Theorem 5.8. *Given an arbitrary $(k+s)$-tuple z' lying within $dist_{\max} = \frac{1}{2}(2^{k+m-1}-3)$ of a $(k+s)$-tuple z for which the position and full cycle vector are known, algorithm $\mathscr{R}$ correctly produces the distance u'_{k+s} from x_{k+s} to z' in Γ_{k+s} and the full cycle vector of z'.*

Proof. Let $(k+s)$-tuple z' with cycle vector $c' = (c'_k, \ldots, c'_{k+s})$ lie within $dist_{\max}$ of tuple z whose position u_{k+s} as full cycle vector $c = (c_{k+1}, \ldots, c_{k+s})$ are known.

Using algorithm $\mathscr{R}$, evaluate $z'\boldsymbol{D}^{(s-i)}$ for $0 \leqslant i \leqslant m$ and to obtain the values of $c'_{k+1}, \ldots, c'_{k+m}$ using the cycle-locating tables $L_{k+1}, \ldots, L_{k+m}$ (as in decoding scheme $\mathscr{D}$). Find u'_k, the distance from $z'\boldsymbol{D}^{(s-k)}$ to x_k in Γ_k, using the complete look-up table U. Next guess values for the bits $c'_{k+m+1}, \ldots, c'_{k+s}$ of the cycle vector of z', based on the corresponding values in the cycle vector of z.

With the given restriction on distance, we show that algorithm $\mathscr{R}$ considers each guess in a limited set of permitted guesses. Among these only one leads to a decoding within $dist_{\max}$ of z.

Suppose the vector of cycle locating bits $(c'_{k+m+1}, \ldots, c'_{k+s})$ differs in two or more positions from the vector $(c_{k+m+1}, \ldots, c_{k+s})$. Then in going from vertex z to the new vertex z' in Γ_{k+s} we must traverse at least two cycle changes. But the distance moved between z and z' is less than $dist_{\max}$, and so by Theorem 5.6 we have a contradiction. Consequently there can only be $s-m+1$ possible choices for the values of the vector

$(c'_{k+m+1}, \ldots, c'_{k+s})$, namely

$$\begin{aligned} c^0 &= (c_{k+m+1}, \ldots, c_{k+s}), \\ c^1 &= (\bar{c}_{k+m+1}, c_{k+m+2}, \ldots, c_{k+s}), \\ &\vdots \\ c^{s-m} &= (c_{k+m+1}, \ldots, c_{k+s-1}, \bar{c}_{k+s}). \end{aligned}$$

Each choice of c^w, $0 \leqslant w \leqslant s-m$, leads to a unique distance from x_{k+s} in Γ_{k+s} using algorithm $\mathscr{D}$. But two distinct choices c^u, c^v with $u > v \geqslant 1$, cannot both identify a tuple in Γ_{k+s} that lies within $dist_{\max}$ of z. By Lemma 5.7 we know that the distance between positions obtained by decoding using c^0 and c^i, $1 \leqslant i \leqslant s-m$, is at least $2^{k+m}-1$ and this is larger than $2dist_{\max}$.

Thus we conclude that only one of the possible positions obtained using $c^0, \ldots, c^{s-m}$ will lie within $dist_{\max}$ of z. Algorithm $\mathscr{R}$ considers each of these possibilities in turn and terminates as soon as it finds the c^i which leads to a value for u'_{k+s} within $dist_{\max}$ of z. The algorithm will correctly obtain the cycle vector of z' and the distance u'_{k+s} of z' from x_{k+s} in Γ_{k+s}. □

6. Conclusions

The success of any position sensing application based on the window property of a sequence depends on finding an efficient procedure for decoding the sequence. Present solutions to this problem involve a look-up table which is very fast but has large memory requirements and a milestone or register implementation which may be very slow.

We have presented a decoding algorithm for a class of de Bruijn sequences that exploits the speed of look-ups but saves on memory by using the structure in the construction of these particular de Bruijn sequences to our advantage. Being restricted to a class of de Bruijn sequences need not be a limitation; in addition these sequences are easily generated.

In a particular implementation it may well be the case that the device, whose position we wish to determine, has some maximum speed and there is a maximum time delay between successive decodings. In such a case the displacement between decodings is restricted and this can be used to give a further marked improvement in memory requirements at the expense of a small additional amount of processing time.

Acknowledgements

Many thanks to Fred Piper, Peter Wild and Chris Mitchell, all at Royal Holloway, University of London, for some most helpful discussions. We would also like to thank the referees for many useful comments.

References

[1] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications (Elsevier, Amsterdam, 1976).

[2] J. Burns and C.J. Mitchell, Coding schemes for two-dimensional position sensing, in: M. Ganley, ed., Cryptography and Coding III (Oxford University Press, 1993).

[3] N.G. de Bruijn, A combinatorial problem, Proc. Kon, Nederl. Akad, Wetensch. 49 (1946) 758–764.

[4] H. Fredricksen, A survey of full length shift register cycle algorithms, SIAM Rev. 24 (1982) 195–221.

[5] S.W. Golomb, Shift Register Sequences (Holden-Day, San Francisco, 1967).

[6] A. Lempel, On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers, IEEE Trans. Comput. C-19 (1970) 1204–1209.

[7] E.M. Petriu, Absolute-type position transducers using a pseudorandom encoding, IEEE Trans. on Instrumentation and Measurement IM-36 (1987) 950–955.

[8] E.M. Petriu, J.S. Basran and F.C.A. Groen, Automated guided vehicle position recovery, IEEE Trans. on Instrumentation and Measurement IM-39 (1990) 254–258.