Coding schemes for two-dimensional position sensing

John Burns and Chris J. Mitchell

Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol BS12 6QZ, England
Computer Science Department, Royal Holloway, University of London, Egham Hill, Egham, Surrey TW20 0EX, England

Abstract

Possible coding and scanning methods for two-dimensional position-sensing are reviewed. Encoding schemes for one type of scanning method, namely 'Window-scanning', are examined in some detail.

1 Introduction

The purpose of this paper is to consider a number of general encoding schemes for a two-dimensional position-sensing scheme. The general idea of the scheme is to write a pattern onto a planar, rectangular surface in such a way that, given any small part of the pattern (of pre-determined nature), the exact location of this part-pattern on the surface can be determined. That is to say, all sub-patterns of a certain type occur at most once in the overall pattern. This then means that a relatively simple scanning device can determine its position by examining any small sub-pattern.

The nature of the scanning device can vary considerably; this means that there is a variety of types of sub-pattern for which surface patterns must be constructed. To make the problem somewhat more difficult, for most scanning methods currently envisaged it is unlikely that the precise *orientation* will be known of the sub-pattern whose location is to be determined.

One obvious application for such position-sensing schemes is in two-dimensional location resolution for *Remote-guided vehicles*. Use of special sequences for one-dimensional position location for such vehicles has previously been discussed by Petriu et al., [1, 2].

This paper is concerned with coding schemes, i.e. with methods for

generating patterns and for subsequently decoding them. A brief survey of the main possibilities for pattern generation and pattern scanning is given, leading to an overview of what general types of coding scheme may be required.

The second part of the paper looks at one type of coding scheme in more detail, namely for 'window-scanning' schemes. The requirement for this type of coding scheme leads naturally to the definition of a number of combinatorial problems. Some of these problems have well-known complete solutions, others have known partial solutions, whilst yet others have hardly been studied at all. The companion problem to code design is that of providing efficient decoding algorithms, i.e. providing a means to translate from a detected surface pattern to the position of that pattern on the coded surface. This problem has historically had little or no attention. Recent work of Lloyd and Burns provides solutions in certain cases, but many versions of the problem lack any efficient techniques.

In summary, the purpose of this paper is to present a practical coding problem for which many of the coding and decoding problems currently lack good solutions. However, it is the authors' belief that this lack of solutions is more due to the lack of attention that the problem has received in the past, rather than to the intrinsic difficulty of the coding and decoding problems.

2 A review of possible scanning and coding methods

2.1 General description of coding schemes

Pattern construction

There appears to be a variety of ways both to construct patterns, i.e. to encode the surface, and to read sub-patterns from the surface. We start by examining general properties of pattern design and scanning.

In all currently envisaged schemes, the surface is partitioned into a large number of *cells*. Each of these cells is assigned an integral value, typically in the range $0, 1, \ldots, c-1$ for some small positive integer c (in which case we say the pattern has c *levels*). The value of each cell is then written onto the surface by some means, e.g. by using c distinct colours or grey scale values. This may either involve 'colouring' the entire surface or simply writing appropriate dots at regular intervals. In the discussion below we refer to different cell colours with the intention of including the case where grey scale values are used.

Types of pattern scanning

Many types of surface scanning scheme can be devised, although the technique we are particular concerned with here involves a scanner capable

of reading the value assigned to a (small) number of cells simultaneously. Of particular interest is the case where a scanner is capable of reading enough cells to determine its position immediately. This type of scanning is considered in more detail in Section 2.3 below.

Cell edge resolution

A major scanning problem we have not yet mentioned relates to the method used to detect the edges of cells. We assume throughout this discussion that the movement of the scanner is irregular, and hence cells will not be traversed at an even rate. Hence in many application domains, timing information cannot be used to recover cell transitions.

First suppose that cells are coloured by the positioning of an appropriately coloured or shaded 'dot' in the centre of each cell. In this case, given that all dots are a different colour to the background, edge detection is not a problem. In some circumstances however, it may be necessary to use cells which are coloured throughout their domain, i.e. the entire patterned surface is coloured. In this case, adjacent cells will probably always need to be coloured in different colours in order that a scanner can detect where one cell ends and another begins.

2.2 Surface partitioning and patterning methods

In our general description of the scheme above, we did not describe how the surface might be partitioned into cells. The most obvious approach is to divide the surface into a regular grid. However, although we do not discuss them here, other approaches may be preferable for practical reasons; for example, it may be desirable to avoid cell boundaries forming continuous lines across the surface.

The simplest regular division is probably what we refer to as the *square grid*, where the rectangular surface is divided into an m by n grid of mn square cells. Each cell is either coloured in its entirety or an appropriately coloured dot (maybe round or maybe asymmetric to give orientation information) is placed in the centre of each cell.

However this is not the only possibility. Two other obvious partitions are the *triangular* and *hexagonal* grids, formed by dividing the plane into congruent regular triangles and hexagons respectively. Both these grids could potentially have advantages in certain situations.

- In the *triangular grid*, each cell has only three neighbours (together with three other cells which it touches in a point). In a rectangular grid each cell has four neighbour cells and four cells which it touches in a point. Given a coding scheme in which each cell transition must involve a colour change and in which each neighbour of a fixed cell must have a different 'colour', a triangular grid offers the potential to

design schemes with fewer colours. This could then reduce the cost and complexity of the scanning device. One potential problem with the triangular grid is the fact that sets of six cells meet in a point, which could cause scanning problems.

- In a *hexagonal grid*, each cell has six neighbours (and no cells which it touches in only a point). In certain scanning environments, the hexagonal grid also offers the possibility of coding schemes with fewer colours.

However, although triangular and hexagonal grids have these possible advantages, encoding schemes are probably more difficult to find in these two cases. Certainly all the existing theory of sequences and arrays with 'window' properties is directed towards rectangular arrays.

It is interesting and informative to consider the set of possible paths between cells in each of the three types of grid described above.

- In a *hexagonal grid* the set of paths between cells forms a triangular grid.

- In a *triangular grid* the set of paths between cells forms a hexagonal grid (given that transitions between cells can only take place between neighbouring cells, i.e. cells sharing an edge).

- In a *square grid* the set of paths between cells again forms a square grid (as above provided that transitions between cells can only take place between neighbouring cells).

2.3 Pattern scanning techniques

We now consider in a little more detail the approach to scanning a pattern which forms the main focus of this paper. That is, we consider the situation that arises when a scanner is used which is capable of 'reading' more than one cell at a time. As previously mentioned, we are particularly interested in the case where the scanner is capable of reading enough cells to determine its position immediately. Clearly, schemes could be devised where a scanner reads a number of cells, but the scanner position cannot be determined until the scanner has examined two or more sets of cells; however, we do not consider such schemes further here.

In the particular case where the scanner can examine a rectangular grid of cells, we refer to the scheme as *window scanning*. More precisely, we define window scanning as being that particular case of the general position detection problem when the following hold.

- The scanner examines a rectangular sub-pattern (or *window*) of fixed size (u by v say).

- The window sub-pattern always uniquely defines the position of the scanner in the pattern.

Now consider how the information provided by a multiple-cell scanner might be used to provide the desired positional information. The first problem for the scanner information processing hardware/software will be deciding what angle the scanned pattern is at (that is, without reference to the actual pattern of colours in the scanned cells).

In the case of a rectangular grid, where we assume that the processing circuitry must resolve the scanned information into a rectangular sub-pattern, three possibilities are as follows.

- The scanner is capable only of the minimum resolution, i.e. there will be four possible orientations for the scanned sub-grid. This would typically be the case where the colours are marked using circular, square or other dots with 90 degree rotational symmetry.
- The scanner is capable of determining the orientation of the scanned sub-grid to within two possibilities (0 or 180 degree rotation). This would typically be the case where the colours are marked using elliptical, rectangular or other dots with 180 degree rotational symmetry.
- The scanner is capable of determining the orientation of the scanned sub-grid completely. This would typically be the case where the colours are marked using asymmetric dots.

In the case of other grids (e.g. triangular or hexagonal grids) similar orientation problems will arise.

Following this first stage of processing, the algorithm will then need to examine the colouring of the cells in the scanned sub-grid. This information must then be sufficient to determine uniquely the position of the scanned sub-grid within the entire coded surface. Hence the degree to which the orientation of the scanned pattern can be resolved without reference to the cell colouring will determine what kind of encoding pattern is required.

2.4 Primary system objectives

As we have seen, many possibilities exist for both the general approach to pattern scanning, and to designing patterns capable of yielding the desired location information. However, there are a number of general criteria applying to all systems which can be used to measure their efficiency. Some of the key criteria are as follows.

- The number of colours used should be minimised, primarily in order to simplify the scanner design.
- The size of the sub-pattern that needs to be examined should be minimised.

- Minimal assumptions should be made regarding the type of path followed by the scanner or the orientation of the window viewed by the scanner.

The first two of these criteria should be read as being relative to the overall pattern size. Simple counting arguments indicate that the larger the pattern size, the larger the number of colours and/or the sub-pattern size will need to be.

3 Coding for window scanning schemes

In the remainder of this paper we consider the coding problem for one particular type of scanning scheme, namely window scanning for square grids. As has already been noted, in such a scheme the patterned surface is divided into an m by n rectangular grid of mn square cells. Each cell is marked in such a way that the scanner can detect which of c levels is assigned to that cell (these levels being numbered $0, 1, \ldots, c-1$). We refer to a scheme as *binary* if $c = 2$, *ternary* if $c = 3$, and, in general, c-ary. We also refer to the m by n rectangular pattern of values from $\{0, 1, \ldots, c-1\}$ used to determine the marking of the grid as a c-ary *array*.

As noted above, one of the main problems to be overcome in designing patterns is the problem of determining both position *and orientation* from the scanned sub-pattern. We consider three versions of the orientation problem, schemes for the solution of which we call: 1-orientable, 2-orientable and 4-orientable arrays.

- *1-orientable arrays* are designed to deal with the simplest case, i.e. where only one possible orientation can occur. In other words, these arrays are suitable only where the orientation of the scanned sub-pattern can be determined completely by means other than that of examining the scanned sub-pattern.

- *2-orientable arrays* are for use where there are 2 possible orientations for the window, namely 'North' or 'South' (i.e. up or down).

- *4-orientable arrays* need to cope with the case where there are four possible orientations for a scanned window, namely 'North, 'South', 'East' or 'West'.

3.1 Preliminary remarks

We next make some general observations applying to all the schemes we consider in the remainder of this paper. We use m and n to denote the

dimensions of the array and u and v to denote the dimensions of the scanned window. For all schemes considered here, we assume that

$$u \leq m$$

and

$$v \leq n,$$

i.e. that the window will fit within the array being scanned. In the 4-orientable case we must cope with the possibility that the scanned window will be 'sideways' with respect to the array. Hence in this case we will also assume that

$$u \leq n$$

and

$$v \leq m,$$

i.e. we assume that

$$\max(u, v) \leq \min(m, n).$$

Each of the array design problems we consider here come in two flavours: periodic and aperiodic. The *aperiodic* case is the one we have so far implicitly considered, i.e. where the array is written onto a planar surface and the scanned sub-array is always completely within the borders of the array.

However, for theoretical and occasionally practical reasons, it is also worth considering the *periodic* case. By theoretical reasons we mean that not only does the theory of such arrays appear to be more tractable than in the aperiodic case, but also existing construction methods are exclusively designed for the periodic case. In the periodic case we consider the array to be wrapped round on itself — in the one-dimensional case this corresponds to writing the sequence on the outside of a cylinder, and in the two-dimensional case to writing the array onto a torus. The window can then be moved anywhere on the array which no longer has any 'edges'. It is also occasionally worth considering the case where one dimension is regarded periodically and the other aperiodically, as would happen if a two-dimensional array were written onto the outside of a cylinder.

Because of the practical importance of the aperiodic case, and because of the theoretical importance of the periodic case, we consider both cases. We show below that any periodic array can be used to construct a slightly larger aperiodic array, and hence the study of the periodic case has practical as well as theoretical interest.

3.2 Formal definitions

Before proceeding we give formal definitions for the objects considered here. All these definitions will apply to c-ary m by n arrays, i.e. arrays

$$A = (a_{ij}, 0 \leq i \leq m-1, 0 \leq j \leq n-1)$$

where each entry a_{ij} in the array satisfies $0 \leq a_{ij} \leq c-1$. Note that it would be quite simple to generalise these definitions to the multi-dimensional case; however, since only the 1- and 2- dimensional cases are of immediate practical application, we consider only those cases here.

If A is an m by n c-ary array, we define its u by v sub-arrays to be the c-ary arrays

$$A_{st} = (a_{ij}^{(st)}, 0 \leq i \leq u-1, 0 \leq j \leq v-1), 0 \leq s \leq m-1, 0 \leq t \leq n-1$$

defined by

$$a_{ij}^{(st)} = a_{i+s,j+t}$$

where $i+s$ is computed modulo m and $j+t$ is computed modulo n.

Observe that in the aperiodic case we are only interested in those sub-arrays A_{st} for which $0 \leq s \leq m-u$ and $0 \leq t \leq n-v$. We call this subset of sub-arrays the *aperiodic sub-arrays*.

1-orientable sequences and arrays

We can now define a *1-orientable aperiodic* (u, v)*-window array* $A = (a_{ij})$ to be a c-ary m by n array ($m \geq u$, $n \geq v$) with the property that all its u by v aperiodic sub-arrays $A_{st}, 0 \leq s \leq m-u, 0 \leq t \leq n-v$ are distinct. I.e. $A_{st} = A_{s't'}$ if and only if $s = s'$ and $t = t'$ (given $0 \leq s, s' \leq m-u$ and $0 \leq t, t' \leq n-v$).

A *1-orientable aperiodic* v*-window sequence* is then simply a 1 by n 1-orientable aperiodic $(1, v)$-window array.

Similarly we define a *1-orientable periodic* (u, v)*-window array* $A = (a_{ij})$ to be a c-ary m by n array ($m \geq u$, $n \geq v$) with the property that all its u by v sub-arrays $A_{st}, 0 \leq s \leq u-1, 0 \leq t \leq v-1$ are distinct. I.e. $A_{st} = A_{s't'}$ if and only if $s = s'$ and $t = t'$ (given $0 \leq s, s' \leq u-1$ and $0 \leq t, t' \leq v-1$).

A *1-orientable periodic* v*-window sequence* is then a 1 by n 1-orientable periodic $(1, v)$-window array.

Observe that what Dénes and Keedwell, [3], call an m by n array having the *u by v window property* is precisely a 1-orientable periodic (u, v)-window array. Similarly what Dénes and Keedwell, [3], call a sequence having the *window property for windows of length v* is precisely a 1-orientable periodic v-window sequence.

2-orientable sequences and arrays

To give the definitions in the 2-orientable case, we first need an additional definition. Given an m by n array $A = (a_{ij}, 0 \leq i \leq m-1, 0 \leq j \leq n-1)$,

we denote the *rotation* of A by 90 degrees by $\mathbf{R}_{90}(A)$ and define it to be the n by m array $\mathbf{R}_{90}(A) = (a_{ij}^{(90)}, 0 \leq i \leq n-1, 0 \leq j \leq m-1)$ where

$$a_{ij}^{(90)} = a_{m-1-j,i}, \;\; 0 \leq i \leq n-1, 0 \leq j \leq m-1.$$

We then write $\mathbf{R}_{180}(A)$ for $\mathbf{R}_{90}(\mathbf{R}_{90}(A))$ and $\mathbf{R}_{270}(A)$ for $\mathbf{R}_{90}(\mathbf{R}_{180}(A))$. Finally observe that $\mathbf{R}_{360}(A) = \mathbf{R}_{90}(\mathbf{R}_{90}(\mathbf{R}_{90}(\mathbf{R}_{90}(A)))) = A = \mathbf{R}_0(A)$.

We can now define a *2-orientable aperiodic* (u, v)*-window array* $A = (a_{ij})$ to be a c-ary m by n array ($m \geq u$, $n \geq v$) with the property that the collection of arrays consisting of the u by v aperiodic sub-arrays of A and the u by v aperiodic sub-arrays of $\mathbf{R}_{180}(A)$ are all distinct.

A *2-orientable aperiodic* v*-window sequence* is then simply a 1 by n 2-orientable aperiodic $(1, v)$-window array.

Similarly we define a *2-orientable periodic* (u, v)*-window array* $A = (a_{ij})$ to be a c-ary m by n array ($m \geq u$, $n \geq v$) with the property that the collection of arrays consisting of the u by v sub-arrays of A and the u by v sub-arrays of $\mathbf{R}_{180}(A)$ are all distinct.

A *2-orientable periodic* v*-window sequence* is then a 1 by n 2-orientable periodic $(1, v)$-window array.

Observe that what Dai, Martin, Robshaw and Wild, [4], call an *orientable sequence of order* n is precisely a (binary) 2-orientable periodic n-window sequence.

4-orientable arrays

In much the same way as before we now define a *4-orientable aperiodic* (u, v)*-window array* $A = (a_{ij})$ to be a c-ary m by n array ($\min(m, n) \geq \max(u, v)$) with the property that the collection of arrays consisting of the u by v aperiodic sub-arrays of A, $\mathbf{R}_{90}(A)$, $\mathbf{R}_{180}(A)$ and $\mathbf{R}_{270}(A)$ are all distinct.

Similarly we define a *4-orientable periodic* (u, v)*-window array* $A = (a_{ij})$ to be a c-ary m by n array ($\min(m, n) \geq \max(u, v)$) with the property that the collection of arrays consisting of the u by v sub-arrays of A, $\mathbf{R}_{90}(A)$, $\mathbf{R}_{180}(A)$ and $\mathbf{R}_{270}(A)$ are all distinct.

Observe that 4-orientable sequences cannot exist, and so we do not consider them further.

Constructing aperiodic window arrays from periodic window arrays

We now describe how any *periodic* s-orientable (u, v)-window sequence (or array) can be transformed into an *aperiodic* s-orientable (u, v)-window sequence (or array) of slightly larger dimensions.

We first consider the sequence case. Suppose $A = (a_i)$ $(0 \leq i \leq n-1)$ is a sequence of length n. Then, given v satisfying $1 \leq v \leq n$, let $\mathbf{E}_v(A) = (b_j)$ $(0 \leq j \leq n + v - 2)$ be the sequence of length $n + v - 1$ defined by

$$b_j = a_{j \bmod n}.$$

Note that $\mathbf{E}_1(A) = A$. Informally $\mathbf{E}_v(A)$ consists of A concatenated with the first $v-1$ terms of itself.

Now suppose $A = (a_{ij})$ $(0 \le i \le m-1,\ 0 \le j \le n-1)$ is an m by n array. Then, given u and v satisfying $1 \le u \le m$ and $1 \le v \le n$, let $\mathbf{E}_{uv}(A) = (b_{ij})$ $(0 \le i \le m+u-2$ and $0 \le j \le n+v-2)$ be the $(m+u-1)$ by $(n+v-1)$ array defined by

$$b_{ij} = a_{i \bmod m, j \bmod n}.$$

As previously observe that $\mathbf{E}_{1,1}(A) = A$.

We can now state the following.

Lemma 1. *If A is a periodic s-orientable v-window sequence of length n ($s \in \{1,2\}$, $n \ge v$), then $\mathbf{E}_v(A)$ is an aperiodic s-orientable v-window sequence of length $n+v-1$.*

Similarly, if A is an m by n periodic s-orientable (u,v)-window array ($s \in \{1,2\}$, $m \ge u$, $n \ge v$), then $\mathbf{E}_{uv}(A)$ is an $(m+u-1)$ by $(n+v-1)$ aperiodic s-orientable (u,v)-window array.

Proof. All four cases of the result (i.e. $s = 1$ and $s = 2$ for sequences and $s = 1$ and $s = 2$ for arrays) are proved in a similar way. To avoid unnecessary duplication we consider one case only, namely $s = 1$ for arrays. In this case it should be clear that the set of u by v sub-arrays of A is identical to the set of u by v *aperiodic* sub-arrays of $\mathbf{E}_{uv}(A)$. The result follows immediately from the definitions. □

The above elementary construction was previously described for the case $s = 1$ by Kanetkar and Wagh, [5]. Finally observe that a similar (although somewhat weaker) result holds for 4-orientable arrays, namely:

Lemma 2. *If A is an m by n periodic 4-orientable (u,v)-window array ($\min(m,n) \ge \max(u,v)$), then $\mathbf{E}_{ww}(A)$ is an $(m+w-1)$ by $(n+w-1)$ aperiodic 4-orientable (u,v)-window array where $w = \min(u,v)$.*

Comments on definitions

In the above definitions we have implicitly ruled out 'self-symmetric' u by v sub-patterns from 2- and 4- orientable arrays. More precisely, it is immediate from the definitions that the following hold.

- A 2-orientable periodic (aperiodic) (u,v)-window array can never contain a u by v (aperiodic) sub-array A_{st} with the property that $\mathbf{R}_{180}(A_{st}) = A_{st}$.

- A 4-orientable periodic (aperiodic) (u, u)-window array can never contain a u by u (aperiodic) sub-array A_{st} with the property that $\mathbf{R}_{90}(A_{st}) = A_{st}$.

However, in practice it would appear that the existence of such symmetric patterns would not cause any problems for a position resolution system (as long as the position only is needed and not the precise angle of orientation of the scanning device). We can therefore conclude that the above definitions are over-restrictive from the view-point of the practical application.

Nevertheless, the above definitions appear to be the most natural from a mathematical respect, and seem likeliest (at least intuitively) to yield practical results in terms of methods of construction. We therefore pursue the above definitions throughout this paper, albeit observing that alternative definitions allowing arrays containing symmetric sub-arrays appear to be a good topic for future work.

4 1-orientable arrays

4.1 A fundamental inequality

We start by giving well-known fundamental bounds for the periodic and aperiodic cases which can be derived by simple counting arguments.

Lemma 3. *The following bound must be satisfied by any 1-orientable aperiodic (u, v)-window array, A:*

$$(m - u + 1)(n - v + 1) \leq c^{uv}.$$

If an array meets the bound then the set of aperiodic u by v sub-arrays of A will contain every possible c-ary u by v array exactly once.

Proof. By definition A has $(m - u + 1)(n - v + 1)$ aperiodic sub-arrays which must all be distinct. However, there are precisely c^{uv} possible u by v c-ary arrays. The bound and the assertion regarding arrays meeting the bound follow. □

Lemma 4. *The following bound must be satisfied by any 1-orientable periodic (u, v)-window array, A:*

$$mn \leq c^{uv}.$$

If an array meets the bound then the set of u by v sub-arrays of A will contain every possible c-ary u by v array exactly once.

Proof. By definition A has mn periodic sub-arrays which must all be distinct. However, there are precisely c^{uv} possible u by v c-ary arrays. The bound and the assertion regarding arrays meeting the bound follow. □

In the spirit of previous authors, e.g. Gordon, [6], we call a 1-orientable aperiodic window array *perfect* if it meets the bound of Lemma 3, and we similarly call a 1-orientable periodic window array *perfect* if it meets the bound of Lemma 4. Reed and Stewart, [7], Gordon, [6], and Etzion, [8], (amongst many others), call perfect binary 1-orientable periodic window arrays *perfect maps*, whereas other authors, (e.g. Iványi, [9], who considers the c-ary case, and Fan, Fan, Ma and Siu, [10]), call such arrays *de Bruijn arrays*. This latter terminology derives from the well-established term *de Bruijn sequence* which has long been used to describe perfect 1-orientable periodic window sequences (see, for example, [11, 12, 13, 14, 15, 16, 17, 18] in the binary case, and [19] in the c-ary case).

In addition, if a 1-orientable aperiodic window array satisfies $(m-u+1)(n-v+1) = c^{uv}-1$, i.e. one less than the maximum of Lemma 3, and the 'missing' u by v array is the all-zero array, then we call it *semi-perfect*. Similarly, a semi-perfect 1-orientable periodic window array is one which satisfies $mn = c^{uv}-1$, i.e. one less than the maximum of Lemma 4, and for which the 'missing' u by v array is the all-zero array. Semi-perfect 1-orientable periodic window arrays are often called *pseudo-random arrays* in the literature, see, for example, [8, 20]. However, this term is somewhat confusing in that it is also often used in a more specialised way to mean an array derived from an m-sequence (following MacWilliams and Sloane, [21]). In fact, pseudo-random arrays in this latter sense are always examples of pseudo-random arrays in the former sense.

Before proceeding we state the following result relating the existence of perfect and semi-perfect periodic and aperiodic sequences and arrays.

Theorem 5.

(i) If A is a perfect periodic 1-orientable v-window sequence, then $\mathbf{E}_v(A)$ is a perfect aperiodic 1-orientable v-window sequence. Similarly, if A is a semi-perfect periodic 1-orientable v-window sequence, then $\mathbf{E}_v(A)$ is a semi-perfect aperiodic 1-orientable v-window sequence.

(ii) If A is a perfect (or semi-perfect) periodic 1-orientable (u,v)-window array, then $\mathbf{E}_{uv}(A)$ is a perfect (or semi-perfect) aperiodic 1-orientable (u,v)-window array.

(iii) There exists a $(c-1)$-to-one correspondence between the set of all perfect periodic 1-orientable v-window sequences and the set of all semi-perfect periodic 1-orientable v-window sequences.

Proof.

(i) If A is a perfect periodic c-ary 1-orientable v-window sequence then, by definition, it has length c^v. By definition $\mathbf{E}_v(A)$ has length c^v+v-1, is an aperiodic 1-orientable v-window sequence by Lemma 1, and hence is perfect

by definition. Similarly, if A is a semi-perfect periodic c-ary 1-orientable v-window sequence then, by definition, it has length $c^v - 1$. By definition $\mathbf{E}_v(A)$ has length $c^v + v - 2$ and does not contain the all-zero v-tuple as an aperiodic sub-sequence, is an aperiodic 1-orientable v-window sequence by Lemma 1, and hence is semi-perfect by definition.

(ii) If A is an m by n perfect periodic c-ary 1-orientable (u, v)-window array then, by definition, $mn = c^{uv}$. By definition $\mathbf{E}_{uv}(A)$ is an $m + u - 1$ by $n + v - 1$ array, is an aperiodic 1-orientable (u, v)-window array by Lemma 1, and hence is perfect by definition. The semi-perfect case follows by an exactly analogous argument.

(iii) A perfect periodic 1-orientable v-window sequence (or, equivalently, a de Bruijn sequence) has length 2^v and (if considered as a 'circular' sequence) will contain every binary v-tuple exactly once, and hence will contain precisely one sequence of v consecutive zeros (which will necessarily be preceded and succeeded by non-zero elements). It will also contain precisely $c - 2$ sequences of $v - 1$ zeros, over and above the two embedded in the sequence of v zeros.
Similarly, a semiperfect 1-orientable periodic v-window sequence will contain precisely $c - 1$ sequences of $v - 1$ zeros (which will necessarily be preceded and succeeded by non-zero elements) and no sequences of v zeros. To obtain a semiperfect sequence from a perfect one it is only necessary to omit one of the zeros from the (unique) sequence of zeros of length v, and to obtain a perfect sequence from a semiperfect one the reverse process can be followed, i.e. add a zero into any one of the $c - 1$ sequences of zeros of length $v - 1$. It should be clear that this defines a $(c - 1)$-to-one correspondence between the two types of sequence. □

4.2 Window sequences

We first consider the case $m = 1$ (and hence $u = 1$), i.e. we look at window sequences.

The binary case

We start by examining the binary case, i.e. $c = 2$. Construction methods for perfect binary 1-orientable periodic v-window sequences (which must satisfy $n = 2^v$) are well-known; as we have already observed, such sequences are normally known as *de Bruijn sequences* (following the work of de Bruijn and Good in the 1940s, [22, 23]). Such sequences exist for every value of $v \geq 1$, and construction techniques for such sequences abound (see, for example, [11, 12, 13, 14, 15, 16, 17, 18, 22, 24, 25]). Indeed, not only do such sequences exist for every v, but the precise number of distinct such sequences is known — it is $2^{2^{v-1}-v}$ (see, for example, [14, 22, 24]).

By Theorem 5(iii), semi-perfect sequences can be constructed using exactly the same techniques. There is a particularly important family of semi-perfect binary 1-orientable periodic window sequences, namely the well known *m-sequences*. These sequences can be generated using linear feedback shift registers equipped with feedback tap positions corresponding to the non-zero coefficients of primitive polynomials over GF(2). A concise review of some of the most significant properties of m-sequences can be found in Macwilliams and Sloane, [21]. Using the 1-1 correspondence of Theorem 5(iii), one can very easily derive a set of de Bruijn sequences from these m-sequences.

We next consider the aperiodic case. By Theorem 5(i), perfect and semiperfect binary aperiodic 1-orientable v-window sequences exist for every v.

It is interesting to observe that the possibility of using 1-orientable window sequences for position detection is well-known. It was discussed in Bondy and Murty's 15-year old book, [26], as well as in more recent papers by Arazi, [27] and Petriu et al., [1, 2, 28, 29, 30, 31].

The c-ary case

Very similar results apply in the c-ary case. Perfect periodic c-ary 1-orientable window sequences (which must satisfy $n = c^v$) can be constructed for every c and v; such sequences are known as c-ary de Bruijn sequences (or simply as de Bruijn sequences). Constructions which work for any c and v were given in 1949 by Good, [23] and Rees, [32]. Since then many other construction techniques have been devised; see, for example, [19, 33, 34, 35, 36].

Theorem 5(iii) means that semiperfect c-ary 1-orientable periodic window sequences (which must satisfy $n = c^v - 1$) can be constructed for every c and v.

When we consider the aperiodic case, as before we need only consider Theorem 5(i). This result implies that perfect and semiperfect c-ary aperiodic 1-orientable v-window sequences exist for every c and v. In summary therefore we have the following result:

Theorem 6. *If $c \geq 2$ and $v \geq 1$ then the following sequences can be constructed:*

(i) A perfect c-ary 1-orientable periodic v-window sequence (having c^v elements),

(ii) A semiperfect c-ary 1-orientable periodic v-window sequence (having $c^v - 1$ elements),

(iii) A perfect c-ary 1-orientable aperiodic v-window sequence (having $c^v + v - 1$ elements),

(iv) A semiperfect c-ary 1-orientable aperiodic v-window sequence (having $c^v + v - 2$ *elements).*

The decoding problem

Given that an abundance of apparently ideal sequences exist for this particular case of the coding problem, one naturally asks whether simple solutions also exist for the corresponding decoding problem. That is, given a particular v-window sequence and given a v-tuple, at what position does that v-tuple lie in the sequence?

This problem has previously been considered by a number of authors, including Arazi, [27] and Petriu, [2, 29, 30]. One simple solution (as mentioned for example in [27]) is to store a complete look-up table giving the conversion from each v-tuple to its position in the sequence. This will result in a look-up table having c^v entries, which would become prohibitively expensive for large v.

In the (binary) m-sequence case, a simple alternative (see, for example, [29]) would be to load the v-tuple into a shift register capable of generating the m-sequence, and clock it until a fixed 'reference' v-tuple is reached. The number of clocks required would indicate the position of the v-tuple in the sequence. This approach would be too computationally expensive for large v.

The two simple solutions outlined above are memory and processor intensive respectively. A compromise between these two approaches has been suggested by Petriu at al., [2, 30], appropriate again to the (binary) m-sequence case. This compromise involves storing a limited look-up table containing 'milestone' values. A v-tuple to be decoded is loaded into a shift register and clocked until it is equal to one of the milestone values for which a position value is stored. This simple solution gives a useful compromise between storage and processing time; however, the product of the storage and processing requirements remains proportional to 2^v, limiting its applicability to relatively small values of v.

For the binary case, Arazi, [27], suggests a radically different approach which has the advantage of being computationally manageable for very large v. However, this solution only applies where it is possible to look at v non-consecutive bits of the sequence, as might be the case if the sequence were to be used to determine the angle of a rotating shaft by examining v out of 2^v bits engraved around its circumference. Arazi constructs sequences for which the decoding problem is simple given that v bits with fixed relative positions can be scanned.

None of the above approaches provide an efficient decoding algorithm for sequences having large v and where a window of v bits is scanned. In the binary case, the best known decoding method appears to be that based on the work of Massey and Liu, [37], who, for the m-sequence case,

showed that the decoding problem can be translated into the well known and computationally tractable problem of extracting discrete logarithms in the field GF(2^k).

Very little is known about the decoding problem in the c-ary case ($c > 2$). Indeed, in general the decoding problem only appears to have been considered for certain special binary 1-orientable periodic window sequences and arrays, although the methods that have been devised also apply to the aperiodic sequences and arrays that can be derived from them.

4.3 Window arrays ($m > 1$)

The binary case

We next consider binary 1-orientable window arrays. As before we divide this discussion into a number of cases.

Perfect periodic arrays

As described above, perfect binary 1-orientable periodic (u, v)-window arrays are usually known as *Perfect Maps*, following the 1962 paper by Reed and Stewart, [7]. In the context of such arrays, the question that naturally arises is as follows. Given a pair u, v $(u, v \geq 1)$ for which possible pairs of positive integers (m, n) satisfying $mn = 2^{uv}$ does there exist an m by n perfect binary periodic 1-orientable (u, v)-window array (or, using the notation of Fan et al., [10], for which m, n does there exist an $(m, n; u, v)$-array)?

This question has only been partially answered. As Fan et al., [10] point out, $mn = 2^{uv}$ is definitely not a sufficient condition for the existence of an m by n perfect binary periodic 1-orientable (u, v)-window array. They cite the case $m = u = 2$, $n = 2^{2s-1}$ and $v = s$, and point out that no such array can exist since the set of 2 by s periodic sub-arrays must contain the all zero sub-array zero times or at least twice. This argument can be generalised to give Lemma 7 below.

Before proceeding, observe that the idea of using perfect maps for 2-dimensional position detection dates back to the 1962 paper of Reed and Stewart, [7]. What does seem to be novel is the idea of using 2- and 4-orientable arrays for position detection where the orientation of the scanned sub-array is unknown.

Lemma 7. *If an m by n binary periodic 1-orientable (u, v)-window array exists which has amongst its u by v periodic sub-arrays the all zero array or the all one array, then*

$$m > u \quad \text{or} \quad u = 1$$

and

$$n > v \quad \text{or} \quad v = 1.$$

The construction results due to Ma, [38], Fan et al, [10] and Etzion, [8], give the following summary of existence results for perfect periodic binary 1-orientable (u, v)-window arrays.

Theorem 8.

(i) Suppose there exists an m by n perfect periodic binary 1-orientable (u, v)-window array A. Then

(a) if every column of A has the property that the sum of its elements is even and $m > u + 1$ then there exists an m by $2^v n$ perfect periodic binary 1-orientable $(u + 1, v)$-window array, and

(b) if every column of A has the property that the sum of its elements is odd then there exists a $2m$ by $2^{v-1}n$ perfect periodic binary 1-orientable $(u + 1, v)$-window array,

(Fan et al., [10] and Ma, [38]).

(ii) For any positive even integer v, there exists a $2^{v^2/2}$ by $2^{v^2/2}$ perfect periodic binary 1-orientable (v, v)-window array (Fan et al., [10]).

(iii) For any pair of positive integers (u, v) there exists a 2^k by 2^{uv-k} perfect periodic binary 1-orientable (u, v)-window array whenever $u < 2^k \leq 2^u$ except when $k = u$ and $v = 2$ (Etzion, [8]).

This is by no means a complete list of existence results for perfect maps; for further information regarding the construction of these arrays the reader is referred to the literature cited in the above theorem and to [20]. It is interesting to observe that Theorem 8(i) implies that considerable progress on the existence question for Perfect Maps may well be possible if construction methods can be devised giving perfect maps with fixed column sums modulo 2.

Semi-perfect periodic arrays

As previously described, semi-perfect (periodic) binary 1-orientable window arrays are often known as *Pseudo-random arrays*. The existence question for such arrays is as follows. Given a pair u, v $(u, v \geq 1)$ for which possible pairs of positive integers (m, n) satisfying $mn = 2^{uv} - 1$ does there exist an m by n semi-perfect binary 1-orientable periodic (u, v)-window array?

This question also remains unanswered in general. The following result summarises some of the main partial answers to the existence question. Note that this result is not an exhaustive list of existence results — further results can be found in two papers by Etzion, [8, 20].

Theorem 9.

(i) Suppose u, v, m and n are positive integers where $m|(2^u - 1)$ and

$$n = \frac{2^{uv} - 1}{m}.$$

If these integers satisfy $m|2^s - 1$ only if $s \geq u$ then there exists an m by n semi-perfect periodic binary 1-orientable (u, v) window array (Etzion, [8], Theorem 5).

(ii) For every pair of positive integers (u, v) there exists an m by n semi-perfect periodic binary 1-orientable (u, v)-window array for some co-prime pair (m, n) satisfying $mn = 2^{uv} - 1$ (Gordon, [6]).

(iii) Suppose u', v, m, n and k are positive integers satisfying:

(a) $m|(2^{u'} - 1)$,

(b) $1 \leq k \leq \phi(m)/u'$,

(c) $n = (2^{ku'v} - 1)/m$, and

(d) $m|2^s - 1$ only if $s \geq u'$.

Then an m by n semi-perfect periodic binary 1-orientable (ku', v)-window array can be constructed (Etzion, [8], Theorem 7).

(iv) For any pair of positive integers (u, v) an m by n semi-perfect binary 1-orientable periodic (u, v)-window array can only exist if $m > u$ or $u = 1$ and $n > v$ or $v = 1$ (from Lemma 7 above).

Nomura et al., [39], first constructed arrays having the parameters of Theorem 9(i) for the case $\gcd(m, n) = 1$. Note also that a simple construction of arrays having the parameters of Theorem 9(i) for the case $m = 2^u - 1$ and $\gcd(m, n) = 1$ can be found in Macwilliams and Sloane, [21]. We conclude this discussion of periodic binary 1-orientable window arrays by mentioning that a number of other 'sub-perfect' periodic arrays have been constructed by Dénes and Keedwell, [3] and Etzion, [8, 20].

Perfect aperiodic arrays

We devote the remainder of this section to a consideration of aperiodic arrays. We start by considering the perfect case. Observe that, by Theorem 5(ii), the existence of an m by n perfect periodic 1-orientable (u, v)-window array implies the existence of an $m+u-1$ by $n+v-1$ perfect aperiodic 1-orientable (u, v)-window array. However, the reverse is not true, i.e. there exist values of m, n, u, v (where $2^{uv} = mn$) for which there does exist an $m+u-1$ by $n+v-1$ perfect aperiodic 1-orientable (u, v)-window array but for which there does not exist an m by n perfect periodic 1-orientable (u, v)-window array. This can be seen from the example of a 3 by 9 perfect aperiodic 1-orientable $(2, 2)$-window array given in Figure 1, since by

Lemma 7, there cannot exist a 2 by 8 perfect periodic 1-orientable $(2, 2)$ window array. Other examples of aperiodic window arrays for which the corresponding periodic window arrays cannot exist are given in Figures 2 and 3.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Figure 1. 3 by 9 perfect aperiodic 1-orientable $(2, 2)$-window array

$$\begin{pmatrix} 0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,0\,0 \\ 0\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0 \\ 1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,1\,1 \end{pmatrix}$$

Figure 2. 3 by 34 perfect aperiodic 1-orientable $(2, 3)$-window array

$$\begin{pmatrix} 0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0 \\ 1\,1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,1\,0\,1 \end{pmatrix}$$

Figure 3. 4 by 33 perfect aperiodic 1-orientable $(3, 2)$-window array

Apart from these examples, further perfect aperiodic arrays not derived from periodic arrays can be obtained from the following two elementary construction methods.

Construction A. Suppose m, n, v are positive integers satisfying $n \geq v$ and $m(n - v + 1) = 2^v$. We construct an m by n binary array. Suppose A is a (binary) de Bruijn sequence of length 2^v (i.e. a perfect binary periodic 1-orientable v-window sequence). Partition the sequence into m segments $S_0, S_1, \ldots, S_{m-1}$ each of length $n - v + 1$ (this is possible since we have assumed that $m(n - v + 1) = 2^v$). To each segment adjoin the next $v - 1$ bits of the sequence A (where necessary working cyclically) to obtain m sub-sequences of A: $T_0, T_1, \ldots, T_{m-1}$ each of length n. Arrange these sub-sequences in any order to form the m rows of an m by n array.

Construction B. Suppose n, u, v are positive integers satisfying $n \geq v$ and $n - v + 1 = 2^{uv}$. We construct a u by n binary array. Let $c = 2^u$, and suppose A is a c-ary de Bruijn sequence of length $c^v = 2^{uv}$ (i.e. a perfect c-ary periodic 1-orientable v-window sequence). Then, by Theorem 5(i), $\mathbf{E}_v(A)$ is a perfect aperiodic 1-orientable v-window sequence, which we write as (a_i), $(0 \leq i \leq 2^{uv} + v - 2)$. Since $c = 2^u$, each element of (a_i) can be written as a u-bit binary tuple, $\mathbf{b}_i$ say (using the binary representation of a_i with, if necessary, leading zeros added). Finally, define a u by $2^{uv} + v - 1$ binary array which has $\mathbf{b}_i$ as column i $(0 \leq i \leq 2^{uv} + v - 2)$.

Theorem 10.

(i) Suppose that m, n, v are positive integers satisfying $n \geq v$ and $m(n-v+1) = 2^v$, and that B is an m by n array constructed using Construction A. Then B is an m by n perfect aperiodic 1-orientable $(1, v)$-window array.

(ii) Suppose that n, u, v are positive integers satisfying $n \geq v$ and $n-v+1 = 2^{uv}$, and that C is a u by n array constructed using Construction B. Then C is a u by n perfect aperiodic 1-orientable (u, v)-window array.

Proof.

(i) Suppose A is the binary de Bruijn sequence of length 2^v used to construct B. Suppose $\mathbf{c}$ is any binary v-tuple. Then, by the de Bruijn property, $\mathbf{c}$ occurs at a unique position in A. The element of A corresponding to the first bit of $\mathbf{c}$ will (trivially) occur in a unique $(n - v + 1)$-bit segment, S_j say. It should be clear that $\mathbf{c}$ will then occur in T_j, and hence in B. Thus every binary v-tuple occurs at least once in B, and hence exactly once in B (since B contains precisely $m(n - v + 1) = 2^v$ aperiodic sub-arrays of size 1 by v).

(ii) Suppose A is the c-ary de Bruijn sequence of length 2^{uv} used to construct C (where $c = 2^u$). It should be clear that there is a 1-1 correspondence between the set of all u by v binary arrays and the set of all c-ary v-tuples. Hence, since every c-ary v-tuple occurs precisely once in A, every u by v binary array will occur uniquely as an aperiodic sub-array of C. □

We conclude this discussion of perfect aperiodic binary window arrays by giving a table of all possible parameter sets for perfect periodic and aperiodic binary 1-orientable (u, v)-window arrays for $uv \leq 6$, $u \leq v$ and, where $u = v$, $m \leq n$. For each parameter set we indicate the status of the existence question for both types of array, where whenever an array is marked as non-existent this follows from Lemma 7. Note that those parameter sets corresponding to de Bruijn sequences (i.e. $m = u = 1$) have been omitted from the table. Note also that an 8 by 8 perfect periodic 1-orientable $(2, 3)$-window array can be constructed using Theorem 8(i)(b) since a 4 by 4 perfect periodic 1-orientable $(2, 2)$-window array exists with all column sums odd (see, for example, Fan et al., [10], Example 5.6).

Table 1. Existence of small perfect binary 1-orientable window arrays

uv	u	v	mn (2^{uv})	m	n	m by n Periodic	$m+u-1$ by $n+v-1$ Aperiodic
2	1	2	4	2	2	Non-existent	(2×3) Exists (10(i))
				4	1	Non-existent	(4×2) Exists (10(i))
3	1	3	8	2	4	Exists (8(iii))	(2×6) Exists (10(i))
				4	2	Non-existent	(4×4) Exists (10(i))
				8	1	Non-existent	(8×3) Exists (10(i))
4	1	4	16	2	8	Exists (8(iii))	(2×11) Exists (10(i))
				4	4	Non-existent	(4×7) Exists (10(i))
				8	2	Non-existent	(8×5) Exists (10(i))
				16	1	Non-existent	(16×4) Exists (10(i))
	2	2	16	1	16	Non-existent	(2×17) Exists (10(ii))
				2	8	Non-existent	(3×9) Exists (Fig. 1)
				4	4	Exists (8(ii))	(5×5) Exists (5(ii))
5	1	5	32	2	16	Exists (8(iii))	(2×20) Exists (10(i))
				4	8	Exists (8(iii))	(4×12) Exists (10(i))
				8	4	Non-existent	(8×8) Exists (10(i))
				16	2	Non-existent	(16×6) Exists (10(i))
				32	1	Non-existent	(32×5) Exists (10(i))
6	1	6	64	2	32	Exists (8(iii))	(2×37) Exists (10(i))
				4	16	Exists (8(iii))	(4×21) Exists (10(i))
				8	8	Exists (8(iii))	(8×13) Exists (10(i))
				16	4	Non-existent	(16×9) Exists (10(i))
				32	2	Non-existent	(8×7) Exists (10(i))
				64	1	Non-existent	(16×6) Exists (10(i))
	2	3	64	1	64	Non-existent	(2×66) Exists (10(ii))
				2	32	Non-existent	(3×34) Exists (Fig. 2)
				4	16	Exists (8(iii))	(5×18) Exists (5(ii))
				8	8	Exists (8(i)(b))	(9×10) Exists (5(ii))
				16	4	Exists (8(iii))	(17×6) Exists (5(ii))
				32	2	Non-existent	(33×4) Exists (Fig. 3)
				64	1	Non-existent	(65×3) Exists (10(ii))

Semi-perfect aperiodic arrays

Semi-perfect aperiodic binary 1-orientable window arrays can again be derived from the corresponding semi-perfect periodic arrays using Theorem 5(ii). However, as in the perfect case, we can construct semi-perfect aperiodic arrays for parameter sets for which periodic arrays cannot exist. The following two construction methods are simple modifications to Constructions A and B described above.

Construction C. Suppose m, n, v are positive integers satisfying $n \geq v$ and $m(n-v+1) = 2^v - 1$. We construct an m by n binary array. Suppose A is a semi-perfect binary periodic 1-orientable v-window sequence

(which must have length $2^v - 1$). Partition the sequence into m segments $S_0, S_1, \ldots, S_{m-1}$ each of length $n - v + 1$ (this is possible since we have assumed that $m(n - v + 1) = 2^v - 1$). To each segment adjoin the next $v - 1$ bits of the sequence A (where necessary working cyclically) to obtain m sub-sequences of A: $T_0, T_1, \ldots, T_{m-1}$ each of length n. Arrange these sub-sequences in any order to form the m rows of an m by n array.

Construction D. Suppose n, u, v are positive integers satisfying $n \geq v$ and $n - v + 1 = 2^{uv} - 1$. We construct a u by n binary array. Let $c = 2^u$, and suppose A is a semi-perfect c-ary periodic 1-orientable v-window sequence (which must have length $c^v - 1 = 2^{uv} - 1$). Then, by Theorem 5(i), $\mathbf{E}_v(A)$ is a semi-perfect aperiodic 1-orientable v-window sequence, which we write as (a_i), $(0 \leq i \leq 2^{uv} + v - 3)$. Since $c = 2^u$, each element of (a_i) can be written as a u-bit binary tuple, $\mathbf{b}_i$ say (using the binary representation of a_i with, if necessary, leading zeros added). Finally, define a u by $2^{uv} + v - 2$ binary array with $\mathbf{b}_i$ as column i $(0 \leq i \leq 2^{uv} + v - 3)$.

Theorem 11.

(i) Suppose that m, n, v are positive integers satisfying $n \geq v$ and $m(n-v+1) = 2^v - 1$, and that B is an m by n array constructed using Construction C. Then B is an m by n semi-perfect aperiodic 1-orientable $(1, v)$-window array.

(ii) Suppose that n, u, v are positive integers satisfying $n \geq v$ and $n - v + 1 = 2^{uv} - 1$, and that C is a u by n array constructed using Construction D. Then C is a u by n semi-perfect aperiodic 1-orientable (u, v)-window array.

Proof.

(i) Suppose A is the binary sequence of length 2^v used to construct B. Suppose $\mathbf{c}$ is any non-zero v-tuple. Then, by the semi-perfect property, $\mathbf{c}$ occurs at a unique position in A. The element of A corresponding to the first bit of $\mathbf{c}$ will (trivially) occur in a unique $(n - v + 1)$-bit segment, S_j say. It should be clear that $\mathbf{c}$ will then occur in T_j, and hence in B. Thus every non-zero binary v-tuple occurs at least once in B, and hence exactly once in B (since B contains precisely $m(n-v+1) = 2^v - 1$ aperiodic sub-arrays of size 1 by v).

(ii) Suppose A is the c-ary sequence of length $2^{uv}-1$ used to construct C (where $c = 2^u$). It should be clear that there is a 1-1 correspondence between the set of all non-zero u by v binary arrays and the set of all non-zero c-ary v-tuples. Hence, since every non-zero c-ary v-tuple occurs precisely once in A, every non-zero u by v binary array will occur uniquely as an aperiodic sub-array of C. □

Other semi-perfect aperiodic arrays have been constructed by a number of authors. Of particular note are the following.

- Nomura et al., [39], consider infinite 2-dimensional arrays in which every $(m+u-1)$ by $(n+v-1)$ sub-array contains every non-zero u by v matrix exactly once; they call such arrays *Maximum Area Matrices*. Clearly these arrays have the property that every $(m+u-1)$ by $(n+v-1)$ sub-array forms a semi-perfect aperiodic window array. Moreover, if the infinite array is also periodic with period m by n, then every m by n sub-array forms a semi-perfect periodic window array. They construct many such arrays.

- Banerji, [40], constructs two infinite families of semi-perfect aperiodic arrays by folding m-sequences. One family has the same parameters as those given by Construction D (see Theorem 11(ii)).

- Kanetkar and Wagh, [5], generalise the method of Banerji, [40], to construct many semi-perfect aperiodic window arrays using 'folded' m-sequences. In particular they construct arrays for every possible parameter set with $uv \leq 15$ and claim, without proof, that their methods can be used to construct arrays for every possible parameter set allowed by the definition.

We conclude this discussion of semi-perfect aperiodic binary window arrays by giving a table of all possible parameter sets for semi-perfect periodic and aperiodic binary 1-orientable (u, v)-window arrays for $uv \leq 6$, $u \leq v$ and, where $u = v$, $m \leq n$. For each parameter set we indicate the status of the existence question for both types of array, where, whenever an array is marked as non-existent, this follows from Theorem 9(iv). Note that those parameter sets corresponding to semi-perfect binary sequences (i.e. $m = u = 1$) have been omitted from the table.

Finally observe that 'sub-perfect' aperiodic window arrays can be constructed from known 'sub-perfect' periodic window arrays using Theorem 5(ii).

The c-ary case

As before we consider the periodic and aperiodic cases separately.

Periodic c-ary arrays

We start by generalising Lemma 7 to give a necessary condition for the existence of a periodic c-ary 1-orientable (u, v)-window array which contains a 'constant' u by v sub-array.

Lemma 12. *If an m by n c-ary periodic 1-orientable (u, v)-window array exists which has amongst its u by v periodic sub-arrays a u by v array all of whose entries are identical, then*

Table 2. Existence of small semi-perfect binary 1-orientable window arrays

uv	u	v	mn	m	n	m by n Periodic	$m+u-1$ by $n+v-1$ Aperiodic
2	1	2	3	3	1	Non-existent	(3×2) Exists (11(i))
3	1	3	7	7	1	Non-existent	(7×3) Exists (11(i))
4	1	4	15	3	5	Exists (9(i))	(3×8) Exists (11(i))
				5	3	Non-existent	(5×6) Exists (11(i))
				15	1	Non-existent	(15×4) Exists (11(i))
	2	2	15	1	15	Non-existent	(2×16) Exists (11(ii))
				3	5	Exists (9(i))	(4×6) Exists (5(ii))
5	1	5	31	31	1	Non-existent	(31×5) Exists (11(i))
6	1	6	63	3	21	Exists (9(i))	(3×26) Exists (11(i))
				7	9	Exists (9(i))	(7×14) Exists (5(ii))
				9	7	Exists (9(iii))	(9×12) Exists (5(ii))
				21	3	Non-existent	(21×8) Exists (11(i))
				63	1	Non-existent	(63×6) Exists (11(i))
	2	3	63	1	63	Non-existent	(2×65) Exists (11(ii))
				3	21	Exists (9(i))	(4×23) Exists (5(ii))
				7	9	?	(8×11) Exists ([5])
				9	7	Exists (9(i))	(10×9) Exists (5(ii))
				21	3	Non-existent	(22×5) Exists ([5])
				63	1	Non-existent	(64×3) Exists (11(ii))

$$m > u \quad \text{or} \quad u = 1$$

and

$$n > v \quad \text{or} \quad v = 1.$$

Proof. Suppose A is an m by n c-ary periodic 1-orientable (u, v)-window array with $m = u > 1$, and suppose there exists a u by v periodic sub-array all of whose entries are identical. Using the notation of Section 3.2, suppose the sub-array is A_{st}, where $0 \leq s \leq m-1$ and $0 \leq t \leq n-1$. Then, since $m = u$, the sub-arrays A_{wt} will contain the same entries as A_{st} for every w $(0 \leq w \leq m-1)$. But since all the entries of A_{st} are equal, this means that A contains m identical periodic sub-arrays, contradicting the definition of periodic window array. Hence $m > u$ or $u = 1$.

A similar argument shows that $n > v$ or $v = 1$ and the lemma follows. □.

There are few *explicit* constructions for c-ary perfect arrays ($c > 2$), although it would appear that many of the constructions for the binary case can be generalised with little effort (see, in particular, Etzion, [8]). Some of the known results on this problem can be summarised as follows.

Theorem 13.

(i) Given any positive integers u, v, c ($c \geq 2$) there exists an m by n perfect periodic c-ary 1-orientable (u, v)-window array for some m, n (Iványi, [9]).

(ii) Given any positive integer u and any odd c, then there exists a c^u by c^u perfect periodic c-ary 1-orientable $(u, 2)$-window array (Etzion, [8]).

(iii) For any pair of positive integers (u, v) an m by n perfect periodic c-ary 1-orientable (u, v)-window array can only exist if

(a) $m > u$ or $u = 1$, and

(b) $n > v$ or $v = 1$

(from Lemma 12 above).

Some literature also exists on the construction of c-ary semi-perfect arrays ($c > 2$). In particular note the papers of Etzion, [8], Nomura et al., [39], and MacWilliams and Sloane, [21]. The following result summarises some of the main known existence results regarding semi-perfect periodic c-ary 1-orientable window arrays.

Theorem 14.

(i) Suppose q is a prime power and u, v and m are positive integers where $m|(q^u - 1)$. Let

$$n = \frac{q^{uv} - 1}{m}.$$

If these integers satisfy

(a) $m|q^k - 1$ only if $k \geq u$, and

(b) $gcd(m, n) = 1$,

then there exists an m by n semi-perfect periodic binary 1-orientable (u, v) window array (Nomura et al., [39]).

(ii) For any pair of positive integers (u, v) an m by n semi-perfect binary 1-orientable periodic (u, v)-window array can only exist if $m > u$ and $n > v$ (from Lemma 12 above).

Note that Dénes and Keedwell, [3] consider 'sub-perfect' c-ary window arrays.

Aperiodic c-ary arrays

As in the binary case, Theorem 5(ii) enables the construction of perfect and semi-perfect aperiodic c-ary window arrays from periodic ones. Constructions A, B, C and D can all be generalised to the c-ary case to give further aperiodic arrays. Finally observe that the constructions of Nomura et al, [39], Banerji, [40] and Kanetkar and Wagh, [5] all generalise to the q-ary case, where q is a prime power.

The decoding problem

The only work that appears to have been done on the decoding problem for arrays (as opposed to sequences) is the recent result of Lloyd and Burns, [41]. Generalising the work of Massey and Liu, [37], Lloyd and Burns show how to reduce the problem of decoding the Pseudorandom Arrays of MacWilliams and Sloane (with the (k_1, k_2)-window property) to the problem of finding the discrete logarithm of an element in $\mathrm{GF}(2^{k_1 k_2})$.

5 2-orientable arrays

5.1 A fundamental inequality

As in the 1-orientable case, we start by deriving simple combinatorial bounds on the sizes of periodic and aperiodic 2-orientable window arrays. However, an important difference here is that, unlike the 1-orientable case where perfect arrays of unbounded size exist, these bounds are not tight in general.

Lemma 15. *The following bound must be satisfied by any m by n aperiodic 2-orientable (u, v)-window array:*

$$(m - u + 1)(n - v + 1) \leq \frac{c^{uv} - c^{\lfloor (uv+1)/2 \rfloor}}{2}.$$

Proof. As noted in Section 3.2, a 2-orientable array can never contain any self-symmetric sub-arrays. Of all the c^{uv} possible u by v c-ary sub-arrays, precisely $c^{\lfloor (uv+1)/2 \rfloor}$ of them are self-symmetric. This is because a u by v array $B = (b_{ij})$ is self-symmetric if and only if $\mathbf{R}_{180}(B) = B$, i.e. if and only if $b_{u+1-i,v+1-j} = b_{ij}$ for every i, j, $(0 \leq i \leq u-1, 0 \leq j \leq v-1)$.

Hence

$$c^{uv} - c^{\lfloor (uv+1)/2 \rfloor}$$

of the u by v arrays are candidates for sub-arrays of a 2-orientable array. Now, by definition, an m by n array has $(m - u + 1)(n - v + 1)$ aperiodic sub-arrays. By definition of 2-orientable, the collection containing the aperiodic sub-arrays of A and $\mathbf{R}_{180}(A)$ must all be distinct, and this collection contains $2(m - u + 1)(n - v + 1)$ arrays. Hence

$$2(m - u + 1)(n - v + 1) \leq c^{uv} - c^{\lfloor (uv+1)/2 \rfloor}.$$

The result now follows. □.

Lemma 16. *The following bound must be satisfied by any m by n periodic 2-orientable (u, v)-window array:*

$$mn \leq \frac{c^{uv} - c^{\lfloor (uv+1)/2 \rfloor}}{2}.$$

Proof. By definition, an m by n array has mn sub-arrays. Following the same argument as in the proof of lemma 15, we have

$$2mn \leq c^{uv} - c^{\lfloor (uv+1)/2 \rfloor}.$$

The result now follows. □.

In the periodic binary sequence case, an improved bound has recently been obtained by Dai et al., [4]. Similar improved bounds can almost certainly be obtained for the c-ary periodic sequence case and the periodic array case (both binary and c-ary). The aperiodic case is somewhat more difficult to handle; however, no doubt some results can also be obtained here.

Before proceeding we give an example of a periodic binary 2-orientable 5-window sequence (of length 6), given in Figure 4, which actually meets the new bound of Dai et al., [4].

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 4. The unique periodic binary 2-orientable 5-window sequence of length 6

5.2 Existence of 2-orientable window sequences

Periodic sequences

We start by considering the binary case. Apart from sequences obtained from computer searches (for small window length v), a recent construction method by Dai et al., [4] yields the best known sequences for almost all values of v; these sequences are asymptotically optimal in length. Very little work appears to have been done on the c-ary case. However, it would seem likely that the construction of Dai et al., [4], can be generalised to give c-ary periodic 2-orientable window sequences.

Aperiodic sequences

Just as in the 1-orientable case, by Lemma 1 aperiodic 2-orientable window sequences can be derived from periodic sequences, and thus the construction of Dai et al., [4] can be applied to give aperiodic binary window sequences. Apart from similar derivations from periodic sequences, no general construction methods for aperiodic 2-orientable window sequences are known. The table given in Figure 3 (derived by computer search) lists the

lengths of the longest known such sequences for $4 \leq v \leq 16$. For v in the range $4 \leq v \leq 7$ the length given is the length of the longest such sequence, since for these values an exhaustive search has been completed.

Table 3. Existence of small binary aperiodic 2-orientable window sequences

window size (v)	sequence length (n)
4	8
5	14
6	26
7	48
8	108
9	210
10	440
11	872
12	1860
13	3710
14	7400
15	15467
16	31766

Even less work appears to have been done on the c-ary case ($c > 2$).

5.3 Existence of 2-orientable window arrays

Very little work has been done on this topic for either the periodic or the aperiodic case. However, examples can be derived from the following simple construction technique.

Construction E. Suppose $A = (a_i)$, $(0 \leq i \leq m-1)$ is a c-ary periodic 1-orientable u-window sequence of length m $(u > 1)$. Suppose $B = (b_j)$, $(0 \leq j \leq n-1)$ is a d-ary periodic 2-orientable v-window sequence of length n $(v > 1)$. Then construct an m by n array $E = (e_{ij})$, $(0 \leq i \leq m-1, 0 \leq j \leq n-1)$, as follows. Let

$$e_{ij} = da_i + b_j$$

for every i, j, $(0 \leq i \leq m-1, 0 \leq j \leq n-1)$.

Theorem 17. *If E is constructed using Construction E, then it is an m by n (cd)-ary periodic 2-orientable (u, v)-window array.*

Proof. First observe that, since $0 \leq a_i \leq c-1$ and $0 \leq b_j \leq d-1$ for every i, j, $(0 \leq i \leq m-1, 0 \leq j \leq n-1)$, we immediately have

$$0 \leq e_{ij} \leq d(c-1) + d - 1 = cd - 1$$

and hence E is a (cd)-ary array.

We need to show that the periodic sub-arrays of E and $\mathbf{R}_{180}(E)$ are all distinct. To do this we need to consider two cases.

(a) *Two sub-arrays of E.* Suppose $X = (e_{s+i,t+j})$ and $Y = (e_{s'+i,t'+j})$ $(0 \leq i \leq u-1, 0 \leq j \leq v-1)$ are u by v sub-arrays of E (where, if necessary, $s+i$ and $s'+i$ are reduced modulo m and $t+j$ and $t'+j$ are reduced modulo n).
If $X = Y$ then

$$e_{s+i,t} = e_{s'+i,t'}$$

for every i, $(0 \leq i \leq u-1)$. Thus, by definition of E,

$$a_{s+i} = a_{s'+i}$$

for every i, $(0 \leq i \leq u-1)$. Hence $s = s'$, since A is 1-orientable. Similarly,

$$e_{s,t+j} = e_{s',t'+j}$$

for every j, $(0 \leq j \leq v-1)$. Thus, by definition of E,

$$b_{t+j} = b_{t'+j}$$

for every j, $(0 \leq j \leq v-1)$. Hence $t = t'$, since B is 2-orientable. Hence X and Y must be the same sub-array of E.

(b) *One sub-array of E and one sub-array of $\mathbf{R}_{180}(E)$.* Suppose $X = (e_{s+i,t+j})$ is a u by v sub-array of E and $Y = (e_{s'+u-1-i,t'+v-1-j})$ is a u by v sub-array of $\mathbf{R}_{180}(E)$, $(0 \leq i \leq u-1, 0 \leq j \leq v-1)$. If $X = Y$ then

$$e_{s+i,t+j} = e_{s'+u-1-i,t'+v-1-j}$$

for every i, j, $(0 \leq i \leq u-1, 0 \leq j \leq v-1)$. Thus, in particular,

$$e_{s,t+j} = e_{s',t'+v-1-j}$$

for every j, $(0 \leq j \leq v-1)$. Thus, by definition of E,

$$b_{t+j} = b_{t'+v-1-j}$$

for every j, $(0 \leq j \leq v-1)$. But this contradicts the assumption that B is a 2-orientable v-window sequence.

The result follows. □

$$\begin{pmatrix} 3 & 3 & 2 & 3 & 2 & 2 \\ 3 & 3 & 2 & 3 & 2 & 2 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 5. 4 by 6 periodic 4-ary 2-orientable (2,5)-window array

As an example of the above construction consider the array derived by setting $A = (1100)$ and $B = (110100)$, where A is a periodic binary 1-orientable 2-window sequence and B is a periodic binary 2-orientable 5-window sequence. The resulting 4 by 6 periodic 4-ary 2-orientable (2,5)-window array is given in Figure 5.

The periodic arrays derived from the above construction method can also be transformed into aperiodic arrays using Lemma 1. Finally observe that, although the above construction is useful in that it does provide examples of 2-orientable arrays, these examples are far from optimal, and there is clearly a need for more research in this area.

6 4-orientable arrays

6.1 A fundamental inequality

As before we start by giving simple combinatorial inequalities governing the sizes of periodic and aperiodic 4-orientable window arrays. As in the 2-orientable case, in general these bounds are not tight.

Lemma 18. *The following bound must be satisfied by any m by n aperiodic 4-orientable (u, v)-window array:*

$$2(mn + uv - 1) - (m + n + 2)(u + v - 2) \leq \frac{c^{uv} - c^{\lfloor (uv+1)/2 \rfloor}}{2}.$$

Proof. As noted in Section 3.2, a 4-orientable array can never contain any self-symmetric sub-arrays, i.e. arrays which map onto themselves under a rotation of 90, 180 or 270 degrees. Clearly, if an array maps onto itself under a rotation of 90 or 270 degrees then it will map onto itself under a rotation of 180 degrees. Hence, of all the c^{uv} possible u by v c-ary sub-arrays, precisely $c^{\lfloor (uv+1)/2 \rfloor}$ of them are self-symmetric (as in the 2-orientable case).

Hence

$$c^{uv} - c^{\lfloor (uv+1)/2 \rfloor}$$

of the u by v arrays are candidates for sub-arrays of a 4-orientable array or one of its rotations. Now, by definition, an m by n array has $(m -$

$u+1)(n-v+1)$ aperiodic sub-arrays. Hence the collection of aperiodic sub-arrays of an m by n array and its rotations by 90, 180 and 270 degrees contains a total of

$$2((m-u+1)(n-v+1)+(m-v+1)(n-u+1))$$

arrays.

Since they must all be distinct we have

$$4(mn+uv-1)-2(m+n+2)(u+v-2) \leq c^{uv}-c^{\lfloor (uv+1)/2 \rfloor}.$$

The result now follows. □.

Lemma 19. *The following bound must be satisfied by any m by n periodic 4-orientable (u,v)-window array:*

$$mn \leq \frac{c^{uv}-c^{\lfloor (uv+1)/2 \rfloor}}{4}.$$

Proof. By definition, an m by n array has mn sub-arrays. Hence the collection of sub-arrays of an m by n array and its rotations by 90, 180 and 270 degrees contains a total of $4mn$ arrays. Following the same argument as in the proof of lemma 18 we have

$$4mn \leq c^{uv}-c^{\lfloor (uv+1)/2 \rfloor},$$

and the result now follows. □.

6.2 Existence of 4-orientable window arrays

We start by repeating the observation (made in Section 3.2 above) that 4-orientable sequences cannot exist. On the other hand, 4-orientable arrays certainly do exist, as shown by the following.

Theorem 20. *Suppose E is constructed from A and B using Construction E, where A is a c-ary periodic 2-orientable u-window sequence of length m $(u>1)$, B is a d-ary periodic 2-orientable v-window sequence of length n $(v>1)$ and $\min(m,n) \geq \max(u,v)$. Then E is an m by n (cd)-ary periodic 4-orientable (u,v)-window array.*

Proof. First observe that, by Theorem 17, E is a 2-orientable (cd)-ary array. In addition, since A is 2-orientable, we may apply the dual of Theorem 17, and hence $\mathbf{R}_{90}(E)$ is also 2-orientable. Hence, to prove the desired result, we need only consider the following case.

Suppose that $X = (e_{s+i,t+j})$ is a u by v sub-array of E and that $Y = (e_{t'+j,s'+u-1-i})$ is a u by v sub-array of $\mathbf{R}_{90}(E)$, $(0 \leq i \leq u-1, 0 \leq j \leq v-1)$. If $X = Y$ then

$$e_{s+i,t+j} = e_{t'+j,s'+u-1-i}$$

for every i, j, $(0 \leq i \leq u-1, 0 \leq j \leq v-1)$. Thus, in particular,

$$e_{s,t+j} = e_{t'+j,s'}$$

for every j, $(0 \leq j \leq v-1)$. Thus, by definition of E,

$$b_{t+j} = b_{s'}$$

for every j, $(0 \leq j \leq v-1)$. Since $v > 1$, by repeating this argument for every possible value of t $(0 \leq t \leq v-1)$ we can show that B must be a constant sequence. But this contradicts the assumption that B is a 2-orientable v-window sequence.

The result now follows. □

As an example of the above construction consider the array derived by setting $A = B = (110100)$, where $A = B$ is a periodic binary 2-orientable 5-window sequence. The resulting 6 by 6 periodic 4-ary 4-orientable (5,5)-window array is given in Figure 6.

$$\begin{pmatrix} 3 & 3 & 2 & 3 & 2 & 2 \\ 3 & 3 & 2 & 3 & 2 & 2 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 3 & 3 & 2 & 3 & 2 & 2 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 6. 6 by 6 periodic 4-ary 4-orientable (5,5)-window array

The periodic arrays derived from the above construction method can also be transformed into aperiodic arrays using Lemma 2. Finally observe that, just as in the 2-orientable case, although the above construction is useful in that it does provide examples of 4-orientable arrays, these examples are far from optimal.

7 Other versions of the problem

We conclude this paper by briefly mentioning one way in which the combinatorial problem considered above could be generalised. We have considered an application of window sequences and arrays and surveyed known

construction and decoding techniques. We have restricted our attention to the one and two dimensional cases, although the definitions could very easily be generalised to the multi-dimensional case.

It is not inconceivable that three and higher dimensional window arrays could find an application, although very little is known about the existence and construction of such arrays. It would appear likely that much of the theory for 2-dimensional arrays would translate directly into corresponding results for the multi-dimensional case, although this remains to be seen. The only published work in this area would appear to be that of Green, [42], who generalises the Pseudorandom Array construction of MacWilliams and Sloane, [21], to the multi-dimensional case, and that of Iványi, [43], who constructs three-dimensional perfect periodic 1-orientable window arrays.

References

1. E.M. Petriu and J.S. Basran. (1989). On the position measurement of automated guided vehicles using pseudorandom encoding. *IEEE Transactions on Instrumentation and Measurement*, **38**, 799–803.

2. E.M. Petriu, J.S. Basran, and F.C.A. Groen. (1990). Automated guided vehicle position recovery. *IEEE Transactions on Instrumentation and Measurement*, **39**, 254–258.

3. J. Dénes and A.D. Keedwell. (1990). A new construction of two-dimensional arrays with the window property. *IEEE Transactions on Information Theory*, **36**, 873–876.

4. Z.D. Dai, K.M. Martin, M.J.B. Robshaw, and P.R. Wild. (1993). Orientable sequences. In M. Ganley, editor, *Cryptography and Coding III*. Oxford University Press.

5. S.V. Kanetkar and M.D. Wagh. (1980). On construction of matrices with distinct submatrices. *SIAM Journal on Algebraic and Discrete Methods*, **1**, 107–113.

6. B. Gordon. (1966). On the existence of perfect maps. *IEEE Transactions on Information Theory*, **IT-12**, 486–487.

7. I.S. Reed and R.M. Stewart. (1962). Note on the existence of perfect maps. *IRE Transactions on Information Theory*, **IT-8**, 10–12.

8. T. Etzion. (1988). Constructions for perfect maps and pseudo-random arrays. *IEEE Transactions on Information Theory*, **34**, 1308–1316.

9. A. Iványi. (1988/89). Construction of infinite de Bruijn arrays. *Discrete Applied Mathematics*, **22**, 289–293.

10. C.T. Fan, S.M. Fan, S.L. Ma, and M.K. Siu. (1985). On de Bruijn arrays. *Ars Combinatoria*, **19A**, 205–213.

11. A.H. Chan and R.A. Games. (1990). On the quadratic spans of de Bruijn sequences. *IEEE Transactions on Information Theory*, **36**, 822–829.

12. A.H. Chan, R.A. Games, and E.L. Key. (1982). On the complexities of de Bruijn sequences. *Journal of Combinatorial Theory, Series A*, **33**, 233–246.

13. T. Etzion. (1986). On the distribution of de Bruijn CR-sequences. *IEEE Transactions on Information Theory*, **IT-32**, 422–423.

14. T. Etzion and A. Lempel. (1984). Algorithms for the generation of full-length shift-register sequences. *IEEE Transactions on Information Theory*, **IT-30**, 480–484.

15. T. Etzion and A. Lempel. (1984). Construction of de Bruijn sequence of minimal complexity. *IEEE Transactions on Information Theory*, **IT-30**, 705–709.

16. T. Etzion and A. Lempel. (1984). On the distribution of de Bruijn sequence of given complexity. *IEEE Transactions on Information Theory*, **IT-30**, 611–614.

17. H. Fredricksen. (1975). A class of nonlinear de Bruijn cycles. *Journal of Combinatorial Theory, Series A*, **19**, 192–199.

18. H. Fredricksen and I. Kessler. (1977). Lexicographic compositions and de Bruijn sequences. *Journal of Combinatorial Theory, Series A*, **22**, 17–30.

19. H. Fredricksen and J. Maiorana. (1978). Necklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Mathematics*, **23**, 207–210.

20. T. Etzion. (1990). On pseudo-random arrays constructed from patterns with distinct differences. In *Sequences*, pages 195–207. Springer-Verlag, New York.

21. F.J. MacWilliams and N.J.A. Sloane. (1976). Pseudo-random sequences and arrays. *Proceedings of the IEEE*, **64**, 1715–1729.

22. N.G. de Bruijn. (1946). A combinatorial problem. *Proceedings Nederlandse Akademie van Wetenschappen*, **49**, 758–764.

23. I.J. Good. (1946). Normally recurring decimals. *Journal of the London Mathematical Society*, **21**, 167–169.

24. H. Fredricksen. (1982). A survey of full length nonlinear shift register cycle algorithms. *SIAM Review*, **24**, 195–221.

25. A. Lempel. (1970). On a homomorphism of the de Bruijn graph and its application to the design of feedback shift registers. *IEEE Transactions on Computers*, **C-19**, 1204–1209.

26. J.A. Bondy and U.S.R. Murty. (1976). *Graph theory with applications.* Elsevier.

27. B. Arazi. (1984). Position recovery using binary sequences. *Electronics Letters*, **20**, 61–62.

28. E.M. Petriu. (1985). Absolute-type pseudorandom shaft encoder with any desired resolution. *Electronics Letters*, **21**, 215–216.

29. E.M. Petriu. (1987). Absolute-type position transducers using a pseudorandom encoding. *IEEE Transactions on Instrumentation and Measurement*, **IM-36**, 950–955.

30. E.M. Petriu. (1988). New pseudorandom/natural code conversion method. *Electronics Letters*, **24**, 1358–1359.

31. E.M. Petriu. (1988). Scanning method for absolute pseudorandom position encoders. *Electronics Letters*, **24**, 1236–1237.

32. D. Rees. (1946). Note on a paper by I.J. Good. *Journal of the London Mathematical Society*, **21**, 169–172.

33. T. Etzion. (1986). An algorithm for constructing m-ary de Bruijn sequences. *Journal of Algorithms*, **7**, 331–340.

34. T. Etzion. (1986). An algorithm for generating shift-register cycles. *Theoretical Computer Science*, **44**, 209–224.

35. A. Ralston. (1981). A new memoryless algorithm for De Bruijn sequences. *Journal of Algorithms*, **2**, 50–62.

36. E. Roth. (1971). Permutations arranged around a circle. *American Mathematical Monthly*, **78**, 990–992.

37. J.L. Massey and R.W. Liu. (1964). Equivalance of nonlinear shift-registers. *IEEE Transactions on Information Theory*, **IT-10**, 378–379.

38. S.L. Ma. (1984). A note on binary arrays with a certain window property. *IEEE Transactions on Information Theory*, **IT-30**, 774–775.

39. T. Nomura, H. Miyakawa, H. Imai, and A. Fukuda. (1972). A theory of two-dimensional linear recurring arrays. *IEEE Transactions on Information Theory*, **IT-18**, 775–785.

40. R.B. Banerji. (1978). The construction of binary matrices with distinct submatrices. *IEEE Transactions on Computers*, **C-27**, 162–164.

41. S.A. Lloyd and J. Burns. (1993). Finding the position of a subarray in a pseudo-random array. In M. Ganley, editor, *Cryptography and Coding III*. Oxford University Press.

42. D.H. Green. (1985). Structural properties of pseudorandom arrays and volumes and their related sequences. *IEE Proceedings, Part E*, **132**, 133–145.

43. A.M. Iványi. (1990). Construction of three-dimensional perfect matrices. *Ars Combinatoria*, **29C**, 33–40.