Decoding de Bruijn arrays constructed by the FFMS method

W.C. Shiu *

Department of Mathematics
Hong Kong Baptist University
224 Waterloo Road
Kowloon, Hong Kong
email: wcshiu@math.hkbu.edu.hk

ABSTRACT. An $(r,s;m,n)$-de Bruijn array is a periodic $r \times s$ binary array in which each of the different $m \times n$ matrices appears exactly once. C.T. Fan, S.M. Fan, S.L. Ma and M.K. Siu established a method to obtain either an $(r, 2^n, m+1, n)$-array or a $(2r, 2^{n-1}s, m+1, n)$-array from an $(r,s;m,n)$-array. A class of square arrays are constructed by their method. In this paper, decoding algorithms for such arrays are described.

1 Introduction and some notations

A periodic binary sequence of length 2^n in which each of the different n-tuples appears exactly once is called a de Bruijn sequence of span n [4]. It became familiar through the works of de Bruijn and Good in 1946 [1, 5]. A generalization of de Bruijn sequences to two dimensional case was held out. Such generalized objects were called de Bruijn arrays [4, 8] or perfect maps [3, 6, 10, 12]. A number of constructions of de Bruijn arrays were devised [3, 4, 8, 10].

Definition 1.1: An $(r,s;m,n)$-*de Bruijn array* (or for short $(r,s;m,n)$-array) is a periodic $r \times s$ binary matrix (with $m \le r$, $n \le s$, $rs = 2^{mn}$) in which each of the different $m \times n$ matrices appears exactly once.

Remark 1.2: Arrays in which each non-zero $m \times n$ matrices appears exactly once were considered in [3, 6, 9, 11, 12].

*Partially Supported by FRG, Hong Kong Baptist University.

A number of applications exist for de Bruijn arrays, such as 2-dimensional range-finding, scrambling, various kinds of mask configuration, coding, etc., one of which is the position location application [2, 12]. Recently, Mitchell and Paterson devised a construction of a class of de Bruijn arrays and a decoding algorithm [10]. Some questions were asked at the end of their paper. One of those is how to decode the square de Bruijn arrays which were constructed by C.T. Fan, S.M. Fan, S.L. Ma and M.K. Siu [4]. For convenience these constructions will be called the FFMS method.

In this paper a decoding algorithm is presented.

Let $M_{r,s}(\mathbb{Z}_2)$ be the set of $r \times s$ matrices (arrays) over $\mathbb{Z}_2$. When $A = (a_{ij}) \in M_{r,s}(\mathbb{Z}_2)$ we let A_{i*} be the i-th row of A and $A_{\cdot j} = \sum_{i=1}^{r} a_{ij}$ be the j-th column sum of A.

Let $A_{i,j}^{(m,n)}$ be the $m \times n$ sub-matrix (sub-array) of A whose (x, y)-th entry is defined by $a_{i-1+x\,j-1+y}$, $1 \le x \le m$, $1 \le y \le n$, where $i - 1 + x$ is computed modulo r and $1 \le i - 1 + x \le r$, and $j - 1 + y$ is computed modulo s and $1 \le j - 1 + y \le s$. That is, the top left hand corner of $A_{i,j}^{(m,n)}$ is the (i, j)-th entry of A.

2 The FFMS method

Let A be an $(r, s; m, n)$ array.

Type 1: If $(A_{\cdot 1}, \ldots, A_{\cdot s}) = \vec{0} = (0, \ldots, 0)$. Take a de Bruijn sequence span n with n 0's at the beginning. Delete the first zero to obtain a sequence $\alpha = c_1, c_2, \ldots, c_{2^n-1}$.

$B = (b_1 b_2 \ldots b_{2^n-1_s}) \in M_{1,2^n s}(\mathbb{Z}_2)$, where

$$\begin{aligned} &b_1 = b_2 = \cdots = b_s = 0 \text{ and} \\ &b_{s+j+k(2^n-1)} = c_j \ (j = 1, \ldots, 2^n - 1, k = 0, 1, \ldots, s - 1). \end{aligned} \tag{2.1}$$

That is, $B = (\underbrace{0\,0 \ldots 0}_{s \text{ entries}} \underbrace{c_1\, c_2 \ldots c_{2^n-1}}_{1\text{st}} \cdots \underbrace{c_1\, c_2 \ldots c_{2^n-1}}_{s\text{-th}}) \in M_{1,2^n s}(\mathbb{Z}_2)$.

Let $Z = (A \vdots A \vdots \cdots \vdots A)$ (2^n copies of A). Then

$$A_1 = \begin{pmatrix} B \\ B + Z_{1*} \\ B + Z_{1*} + Z_{2*} \\ \vdots \\ B + \sum_{i=1}^{r-1} Z_{i*} \end{pmatrix} \in M_{r,2^n s}(\mathbb{Z}_2).$$

is an $(r, 2^n s; m + 1, n)$ de Bruijn array. For convenience this array will be called an *array of type 1 constructed from A and α.*

Example 2.1: Consider the example 5.4 in [4].

$$A = \begin{pmatrix} 0&0&0&0&1&0&1&0&0&0&1&1&0&1&0&1 \\ 0&0&0&1&1&0&1&1&0&0&1&1&1&1&1&0 \\ 1&1&1&1&0&1&0&1&1&1&0&0&1&0&1&0 \\ 1&1&1&0&0&1&0&0&1&1&0&0&0&0&0&1 \end{pmatrix}$$ is a (4,16; 2,3)-array.

Let $\alpha = c_1, c_2, c_3, c_4, c_5, c_6, c_7 = 0011101$. Then we obtain the $(4, 128; 3, 3)$-array A_1 of type 1 constructed from A and α. Such array is described in Appendix (1).

Type 2: If $(A_{\cdot 1}, \dots, A_{\cdot s}) = \vec{1} = (1, \dots, 1)$. Let $B = (b_1\, b_2 \dots b_{2^{n-1}s}) \in M_{1,2^{n-1}s}(\mathbb{Z}_2)$ be defined as follows.

For $n \geq 3$ we let $b_1 = b_2 = \cdots = b_s = 0$, $b_{s+j+(2^{n-1}-1)k} = c_j$ $(j = 1, \dots, 2^{n-1} - 1; k = 0, 1, \dots, s-1)$, i.e., $B = (\underbrace{0\,0 \dots 0}_{s \text{ entries}} \underbrace{c_1\, c_2 \dots c_{2^{n-1}-1}}_{1\text{st}} \dots \underbrace{c_1\, c_2 \dots c_{2^{n-1}-1}}_{s-th}) \in M_{1,2^{n-1}s}(\mathbb{Z}_2)$, where $c_j = a_0 + a_1 + \cdots + a_{j-1}$ with $\beta = a_0, a_1, \dots, a_{2^{n-1}-1}$ being a de Bruijn sequence of span $n-1$ with $n-1$ 0's at the beginning. For $n = 1$ we let $b_u = 0$ $(u = 1, \dots, s)$ and for $n = 2$ we let $b_1 = b_2 = \cdots = b_s = 0$, $b_{s+2k-1} = 0$, $b_{s+2k} = 1$ $(k = 1, \dots, \frac{s}{2})$.

Let $Z = \begin{pmatrix} \overbrace{A \vdots A \vdots \cdots \vdots A}^{2^{n-1} \text{ copies}} \\ A \vdots A \vdots \cdots \vdots A \end{pmatrix}$ $(2 \times 2^{n-1}$ copies of $A)$. Then

$$A_1 = \begin{pmatrix} B \\ B + Z_{1*} \\ B + Z_{1*} + Z_{2*} \\ \vdots \\ B + \sum_{i=1}^{r-1} Z_{i*} \\ B + \sum_{i=1}^{r} Z_{i*} \\ \vdots \\ B + \sum_{i=1}^{2r-1} Z_{i*} \end{pmatrix} \in M_{2r,2^{n-1}s}(\mathbb{Z}_2)$$

is a $(2r, 2^{n-1}s; m+1, n)$ de Bruijn array. Note that $\sum_{i=1}^{r} Z_{i*} = \sum_{i=1}^{r} A_{i*} = \vec{1}$. For convenience this array will be called an *array of type 2 constructed from A and β*. (For $n = 1$ or 2, this array will be called an *array of type 2 constructed from A*).

Example 2.2: The $(8, 8; 3, 2)$-array of type 2 constructed from a $(4, 4; 2, 2)$-

array

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \text{ is } A_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Note that this is the example 5.6 in [4].

Example 2.3: Let $A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$. Then A is a $(2,4;1,3)$-array. Let $\beta = 0011$, which is a de Bruijn sequence span 2. Then $c_1, c_2, c_3 = 001$. We obtain the following $(4,16;2,3)$-array A_1 of type 2 constructed from A and β.

$$A_1 = \begin{pmatrix} 0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&1 \\ 0&0&0&1&0&0&1&1&0&1&0&1&1&0&0&0 \\ 1&1&1&1&1&1&0&1&1&0&1&1&0&1&1&0 \\ 1&1&1&0&1&1&0&0&1&0&1&0&0&1&1&1 \end{pmatrix}.$$

3 Decoding for arrays of type 1

For any $(m+1) \times n$ matrix N over $\mathbb{Z}_2$, let D: $M_{m+1,n}(\mathbb{Z}_2) \to M_{m,n}(\mathbb{Z}_2)$ be the mapping defined by

$$D(N) = \begin{pmatrix} N_{1*} + N_{2*} \\ N_{2*} + N_{3*} \\ \vdots \\ N_{m*} + N_{m+1*} \end{pmatrix}, N \in M_{m+1,n}(\mathbb{Z}_2). \text{ [4, section 4; 7]}. \quad (3.1)$$

Suppose A is an $(r,s;m,n)$-array with $(A_{\cdot 1}, \ldots, A_{\cdot s}) = \vec{0}$ and A_1 is an $(r, 2^n s; m+1, n)$-array of type 1 constructed from A and some α. The following theorem permits us to find the location in A_1 of a given matrix $M \in M_{m+1,n}(\mathbb{Z}_2)$.

Theorem 1. *Suppose A is an $(r,s;m,n)$-array with $(A_{\cdot 1}, \ldots, A_{\cdot s}) = \vec{0}$, and that $0, c_1, c_2, \ldots, c_{2^n-1}$ is a de Bruijn sequence span n, with n 0's at the beginning. Let A_1 is an $(r, 2^n s; m+1, n)$-array of type 1 constructed from A and $\alpha = c_1, c_2, \ldots, c_{2^n-1}$. Let $M \in M_{m+1,n}(\mathbb{Z}_2)$. If $M = (A_1)_{h,k}^{(m+1,n)}$ then $D(M) = A_{h,j}^{(m,n)}$, for some $1 \le j \le s$. Moreover, if we let*
$\vec{a} = M_{1*} - A_{1,j}^{(1,n)} - A_{2,j}^{(1,n)} - \cdots - A_{h-1,j}^{(1,n)}$ then either $\vec{a} = \vec{0}$ or $\exists!\ g$, $1 \le$

$g \le 2^n - 1$, such that $\vec{a} = (c_g\, c_{g+1} \dots c_{g+n-1})$ (c_1 follows c_{2^n-1}). For the first case $k = j$ and for the last case $\exists!\ x$, $0 \le x \le s-1$, such that $k = s + g + (2^n - 1)\, x \equiv j \pmod{s}$.

Proof: By the construction described in Section 2 we have

$$M = \begin{pmatrix} B_k + (Z_{1,k}^{(r,n)})_{1*} + \cdots + (Z_{1,k}^{(r,n)})_{h-1*} \\ B_k + (Z_{1,k}^{(r,n)})_{1*} + \cdots + (Z_{1,k}^{(r,n)})_{h+m-1*} \end{pmatrix}$$

where $B_k = (b_k\, b_{k+1} \dots b_{k+n-1})$, and $h + m - 1$ is considered modulo r, $1 \le h + m - 1 \le r$. Then $D(M) = \begin{pmatrix} (Z_{1,k}^{(r,n)})_{h*} \\ \vdots \\ (Z_{1,k}^{(r,n)})_{h+m-1*} \end{pmatrix} = A_{h,j}^{(m,n)}$ where $j \equiv k \pmod{s}$, $1 \le j \le s$. Hence

$$(A_1)_{1,k}^{(r,n)} = \begin{pmatrix} \vec{a} \\ \vec{a} + A_{1,j}^{(1,n)} \\ \vec{a} + A_{1,j}^{(1,n)} + A_{2,j}^{(1,n)} \\ \vdots \\ \vec{a} + A_{1,j}^{(1,n)} + A_{2,j}^{(1,n)} + \cdots + A_{r-1,j}^{(1,n)} \end{pmatrix},$$

for some $\vec{a} \in \mathbb{Z}_2^n$ and therefore, $\vec{a} = M_{1*} - A_{1,j}^{(1,n)} - A_{2,j}^{(1,n)} - \cdots - A_{h-1,j}^{(1,n)}$. Since $k \equiv j \pmod{s}$, then $\vec{a} = (b_{ts+j}\, b_{ts+j+1} \dots b_{ts+j+n-1})$ for some t, where b_u's were defined in (2.1)

It is easy to see from the definition of B that $\vec{a} = \vec{0}$ if and only if $1 \le k \le s$. Therefore $k \equiv j \pmod{s}$ yields $k = j$. If $\vec{a} \ne \vec{0}$ there is a unique g, $1 \le g \le 2^n - 1$, such that $\vec{a} = (c_g\, c_{g+1} \dots c_{g+n-1})$ (c_1 follows c_{2^n-1}) where c_t are defined in (2.1). Then

$$k = s + g + (2^n - 1)\, x \equiv j \pmod{s}.$$

This equation has a unique solution modulo s since g.c.d.$(2^n - 1, s) = 1$. □

Decoding algorithm for arrays of type 1:

We shall keep the notations introduced above.

Step 1: Compute $D(M)$.
Step 2: Find the location of $D(M)$ in A. Let us say $D(M) = A_{i,j}^{(m,n)}$.
Step 3: Let $\vec{a} = M_{1*} - A_{1,j}^{(1,n)} - A_{2,j}^{(1,n)} - \cdots - A_{i-1,j}^{(1,n)}$.
Step 4: If $\vec{a} = \vec{0}$ then set $k = j$ and go to step 7. If $\vec{a} \ne \vec{0}$ then find g, $1 \le g \le 2^n - 1$, such that $\vec{a} = (c_g\, c_{g+1} \dots c_{g+n-1})$ where c_t are

defined in (2.1).

Step 5: Solve the congruence equation $g + (2^n - 1)x \equiv j \pmod{s}$, $0 \le x \le s - 1$.

Step 6: $k = s + g + (2^n - 1)x$.

Step 7: The top left hand corner of M is the (i, k)-th entry of A_1.

Example 3.1: Consider the $(4, 128; 3, 3)$-array A_1 constructed in Example 2.1. Now suppose $M = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$. We want to find its location in A_1.

Step 1: $D(M) = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$.

Step 2: $D(M) = A^{(2,3)}_{2,6}$.

Step 3: Since $A^{(4,3)}_{1,6} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$, $\vec{a} = 010 - 010 = 000$.

Step 4: $k = 6$.

Step 7: $M = (A_1)^{(3,3)}_{2,6}$.

Example 3.2: Consider the array in Example 3.1. Suppose $M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

Step 1: $D(M) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$.

Step 2: $D(M) = A^{(2,3)}_{3,11}$.

Step 3: $\vec{a} = 110 - 110 - 111 = 111$.

Step 4: $g = 3$.

Step 5: $3 + 7x \equiv 11 \pmod{16} \Rightarrow 7x \equiv 8 \pmod{16} \Rightarrow x \equiv 8 \pmod{16}$.

Step 6: $k = 16 + 3 + 7 \times 8 = 75$.

Step 7: $M = (A_1)^{(3,3)}_{3,75}$.

4 Decoding for arrays of type 2

Now we are going into decoding a $(2r, 2^{n-1}s; m + 1, n)$-array of type 2 constructed from an $(r, s; m, n)$-array A with $(A_{\cdot 1}, \ldots, A_{\cdot s}) = \vec{1}$.

Theorem 2. *Suppose A is an $(r, s; m, n)$-array, $n \ge 3$, with $(A_{\cdot 1}, \ldots, A_{\cdot s}) = \vec{1}$ and $\beta = a_0, a_1, \ldots, a_{2^{n-1}-1}$ is a de Bruijn sequence span $n - 1$ with $n - 1$ 0's at the beginning. Let A_1 be a $(2r, 2^{n-1}s; m + 1, n)$-array of type 2 constructed from A and β. Let $M \in M_{m+1,n}(\mathbb{Z}_2)$.*

If $M = (A_1)_{h,k}^{(m+1,n)}$ *and* $D(M) = A_{i,j}^{(m,n)}$ *then* $h = i$ *or* $h = r + i$ *and* $j \equiv k \pmod{s}$, $1 \le j \le s$, *where* D *is defined in (3.1). Moreover, if we let* $\vec{a}' = M_{1*} - A_{1,j}^{(1,n)} - A_{2,j}^{(1,n)} - \cdots - A_{h-1,j}^{(1,n)}$ *then we have one of the following cases:*

(1) $\vec{a}' = \vec{0}$;

(2) $\vec{a}' = \vec{1}$;

(3) $\exists!\ g$, $1 \le g \le 2^{n-1} - 1$, *such that* $\vec{a}' = (c_g\, c_{g+1} \ldots c_{g+n-1})$ (c_1 *follows* $c_{2^{n-1}-1}$) *where* c_t *are defined in (2.2);*

(4) $\exists!\ g$, $1 \le g \le 2^{n-1} - 1$, *such that* $\mathbf{a} = \vec{1} + \vec{a}' = (c_g\, c_{g+1} \ldots c_{g+n-1})$ (c_1 *follows* $c_{2^{n-1}-1}$) *where* c_t *are defined in (2.2).*

These cases correspond, respectively, to that (1) $k = j$ *and* $h = i$, *(2)* $k = j$ *and* $h = r + i$, *(3)* $\exists!\ x$, $0 \le x \le s - 1$ *such that* $k = s + g + (2^{n-1} - 1)\, x \equiv j \pmod{s}$ *and* $h = i$, *and (4)* $\exists!\ x$, $0 \le x \le s - 1$, *such that* $k = s + g + (2^{n-1} - 1)\, x \equiv j \pmod{s}$ *and* $h = r + i$.

Proof: By the construction described in Section 2 we have

$$M = \begin{pmatrix} B_k + (Z_{1,k}^{(r,n)})_{1*} + \cdots + (Z_{1,k}^{(r,n)})_{h-1*} \\ \vdots \\ B_k + (Z_{1,k}^{(r,n)})_{1*} + \cdots + (Z_{1,k}^{(r,n)})_{h+m-1*} \end{pmatrix}$$

where $B_k = (b_k\, b_{k+1} \ldots b_{k+n-1})$.

$$\text{Then } D(M) = \begin{pmatrix} (Z_{1,k}^{(r,n)})_{h*} \\ \vdots \\ (Z_{1,k}^{(r,n)})_{h+m-1*} \end{pmatrix} = \begin{cases} A_{h,j}^{(m,n)} & \text{if } h \le r \\ A_{h-r,j}^{(m,n)} & \text{if } h > r \end{cases}$$

where $j \equiv k \pmod{s}$, $1 \le j \le s$. That is, if $D(M) = A_{i,j}^{(m,n)}$ then $h = i$ or $h = r + i$. Hence

$$(A_1)_{1,k}^{(2r,n)} = \begin{pmatrix} \vec{a} \\ \vec{a} + A_{1,j}^{(1,n)} \\ \vec{a} + A_{1,j}^{(1,n)} + A_{2,j}^{(1,n)} \\ \vdots \\ \vec{a} + A_{1,j}^{(1,n)} + A_{2,j}^{(1,n)} + \cdots + A_{r-1,j}^{(1,n)} \\ \vec{a} + A_{1,j}^{(1,n)} + A_{2,j}^{(1,n)} + \cdots + A_{r,j}^{(1,n)} \\ \vdots \\ \vec{a} + A_{1,j}^{(1,n)} + A_{2,j}^{(1,n)} + \cdots + A_{r,j}^{(1,n)} + A_{1,j}^{(1,n)} + A_{2,j}^{(1,n)} + \cdots + A_{r-1,j}^{(1,n)} \end{pmatrix},$$

for some $\vec{a} \in \mathbb{Z}_2^n$, and therefore, $\vec{a}' = M_{1*} - A_{1,j}^{(1,n)} - A_{2,j}^{(1,n)} - \cdots - A_{i-1,j}^{(1,n)}$. Since $k \equiv j \pmod{s}$, then $\vec{a}' = \vec{b}'$ or $\vec{a}' = \vec{b} + \vec{1}$, where $\vec{b} = (b_{ts+j}\, b_{ts+j+1} \ldots b_{ts+j+n-1})$ for some t, and b_u's were defined in (2.2).

We can see in a similar fashion to the one used in the theorem 1 that if $\vec{a}' = \vec{0}$ then $k = j$. If $\vec{a}' \neq \vec{0}$ and there is a g, $1 \leq g \leq 2^{n-1} - 1$, such that $\vec{a}' = (c_g\, c_{g+1} \ldots c_{g+n-1})$ (c_1 follows $c_{2^{n-1}-1}$), then $k = s + g + (2^{n-1} - 1)x \equiv j \pmod{s}$ and $h = i$. If the value of $\vec{a}'$ is not one of the cases discussed above, then we let $\vec{a} = \vec{a}' + \vec{1}$. Now we fall in one of the cases considered above and k can be found. In this case $h = r + i$. □

Decoding algorithm for arrays of type 2 (for n ≥ 3):

We shall keep the notations introduced above.

Step 1: Compute $D(M)$.
Step 2: Find the location of $D(M)$ in A. Let us say $D(M) = A_{i,j}^{(m,n)}$.
Step 3: Let $\vec{a}' = M_{1*} - A_{1,j}^{(1,n)} - A_{2,j}^{(1,n)} - \cdots - A_{i-1,j}^{(1,n)}$.
Step 4: If $\vec{a}' = \vec{0}$ then set $k = j$, $h = i$ and go to step 10.
Step 5: If $\vec{a} \neq \vec{0}$ then solve for g such that $\vec{a}' = (c_g\, c_{g+1} \ldots c_{g+n-1})$, $1 \leq g \leq 2^{n-1} - 1$, where c_t's were defined in (2.2).
Step 6: If we can find a solution in step 5 then set $h = i$ and go to step 8. If we cannot find any solution in step 5 then let $\vec{a} = \vec{a}' + \vec{1}$, $h = r + i$.
Step 7: If $\vec{a} = \vec{0}$ then set $k = j$ and go to step 10. If $\vec{a} \neq \vec{0}$ then find g, $1 \leq g \leq 2^{n-1} - 1$, such that $\vec{a} = (c_g\, c_{g+1} \ldots c_{g+n-1})$ where c_t's were defined in (2.2).
Step 8: Solve the congruence equation $g + (2^{n-1} - 1)\, x \equiv j \pmod{s}$, $0 \leq x \leq s - 1$.
Step 9: $k = s + g + (2^{n-1} - 1)x$.
Step 10: The top left hand corner of M is the (h, k)-th entry of A_1.

For $n = 2$ we also have the following decoding algorithm:

Decoding algorithm for arrays of type 2 (for n = 2):

Keep the notations as above.

Step 1: Compute $D(M)$.
Step 2: Find the location of $D(M)$ in A. Let us say $D(M) = A_{i,j}^{(m,2)}$.
Step 3: Let $\vec{a}' = M_{1*} - A_{1,j}^{(1,2)} - A_{2,j}^{(1,2)} - \cdots - A_{i,j}^{(1,2)}$.
Step 4: If $\vec{a}' = \vec{0}$ then set $k = j$, $h = i$ and go to step 8.
Step 5: If "j is odd and $\vec{a}' = 01$" or "j is even and $\vec{a}' = 10$" then

$k = s + j$ and $h = i$. Go to step 8.

Step 6: If $\vec{a}' = 11$ then set $k = j$, $h = r + i$ and go to step 8.

Step 7: If "j is even and $\vec{a}' = 01$" or "j is odd and $\vec{a}' = 10$" then $k = s + j$ and $h = r + i$. Go to step 8.

Step 8: The top left hand corner of M is the (h, k)-th entry of A_1.

Remark: For $n = 1$, if $\vec{a}' = 0$ then $k = j$, $h = i$; if $\vec{a}' = 1$ then $k = j$, $h = r + i$.

Example 4.1: Consider the $(8, 8; 3, 2)$-array A_1, in Example 2.2. Suppose $M = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$. One can scan A_1, and find that $M = (A_1)_{4,6}^{(3,2)}$. The algorithm suggest the following steps:

Step 1: $D(M) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$.

Step 2: $D(M) = A_{4,2}^{(2,2)}$.

Step 3: Since $A_{1,2}^{(4,2)} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$, $\vec{a}' = 00 - 01 - 00 - 11 = 10$.

Step 5: $h = 4$, $k = 4 + 2 = 6$.

Step 8: $M = (A_1)_{4,6}^{(3,2)}$.

Example 4.2: Consider the $(4, 16; 2, 3)$-array A_1, in Example 2.3. Suppose $M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$. Then

Step 1: $D(M) = (1\ 1\ 0)$.

Step 2: $D(M) = A_{2,2}^{(1,3)}$.

Step 3: $\vec{a}' = 011 - 001 = 010$.

Step 5: $g = 2$.

Step 6: $h = 2$.

Step 8: $2 + 3x \equiv 2 \pmod 4 \Rightarrow x = 0$.

Step 9: $k = 4 + 2 = 6$.

Step 10: $M = (A_1)_{2,6}^{(2,3)}$.

Example 4.3: Consider the $(4, 16; 2, 3)$-array A_1, in Example 2.3 again. Suppose $M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ Then

Step 1: $D(M) = (0\ 1\ 0)$.

Step 2: $D(M) = A_{2,2}^{(1,3)}$.

Step 3: $\vec{a}' = 110$.

Step 6: $\vec{a} = \vec{a}' + \vec{1} = 001$, $h = 2 + 1$.
Step 7: $g = 1$.
Step 8: $1 + 3x \equiv 3 \pmod 4 \Rightarrow x = 2$.
Step 9: $k = 4 + 1 + 6 = 11$.
Step 10: $M = (A_1)^{(2,3)}_{3,11}$.

5 Decoding for square arrays constructed by the FFMS method

Given a square array, Fan et al. [4] established a construction to obtain another square array. This construction can be defined succinctly as follows:

Let A be an $(r, s; m, n)$-array then the transpose of A is an $(s, r; n, m)$-array. We shall denote such array as A^T.

Suppose A is an $(s, s; n, n)$-array. We choose 4 de Bruijn sequences $0, c_1, c_2, \ldots$; $0, c_{11}, c_{21}, \ldots$; $0, c_{12}, c_{22}, \ldots$; and $0, c_{13}, c_{23}, \ldots$ of spans n, n, $n+1$ and $n+1$, respectively, such that each of them has the largest subsequence of zeros. Let $\alpha = c_1, c_2, \ldots$, $\alpha_1 = c_{11}, c_{21}, \ldots$, $\alpha_2 = c_{12}, c_{22}, \ldots$ and $\alpha_3 = c_{13}, c_{23}, \ldots$. We obtain an $(s, 2^n s; n+1, n)$-array A_1 of type 1 from A and α, a $(2^n s, 2^{n+1} s; n+1, n+1)$-array A_2 of type 1 from A_1^T and α_1, a $(2^n s, 2^{n+2} s; n+2, n+1)$-array A_3 of type 1 from A_2 and α_2 and a $(2^{n+2} s, 2^{n+2} s; n+2, n+2)$-array A_4 of type 1 from A_3^T and α_3 [4].

The array A_4 can be decoded by the following procedure:

Suppose $M_4 \in M_{n+2,n+2}(\mathbb{Z}_2)$. Let $M_3 = D(M_4)$, $M_2 = D(M_3^T)$, $M_1 = D(M_2)$ and $M_0 = D(M_1^T)$. Using the algorithm described in section 3 with M_0 and α we can locate M_1^T in A_1 and M_1 in A_1^T. Using the algorithm with M_1 and α_1 we can locate M_2 in A_2. Using the algorithm with M_2 and α_2 we can locate M_3^T in A_3 and M_3 in A_3^T. Using the algorithm with M_3 and α_3 we can locate M_4 in A_4.

Remarks:

1. Before applying the procedure described above we must know A_1, A_2 and A_3.

2. For finding the location of M_4 we have to know all the locations of M_1^T, M_2 and M_3^T. Nevertheless, it is not necessary to scan by a 'brute force' method the matrices A_1, A_2 and A_3. That is, we need not use the 'brute force' method.

The proof of Theorem 6.1 in [4] suggest us to construct a $(256, 256; 4, 4)$-array from the $(4, 16; 3, 2)$-array

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

as follows:

Choose $\alpha = 0011101 = \alpha_1$ and $\alpha_2 = 000100110101111$. We obtain the $(16, 32; 3, 3)$-array A_1 of type 1 from A^T and α, the $(16, 256; 4, 3)$-array A_2 of type 1 from A_1 and α_1 and the $(256, 256; 4, 4)$-array A_3 of type 1 from A_2^T and α_2. Note that the construction of A_3 differs from the general construction of square array described above.

Example 5.1: Suppose $M_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. We want to find its location in A_3. Let $M_2 = D(M_3)$, $M_1 = D(M_2^T)$, $M_0 = D(M_1)$. That is

$M_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, $M_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$, and $M_0 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. Hence

$M_0^T = A_{1,3}^{(3,2)}$ and $M_0 = (A^T)_{3,1}^{(2,3)}$. Now $(A^T)_{1,1}^{(16,3)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ \vdots & \vdots & \vdots \\ 1 & 1 & 0 \end{pmatrix}$. We have $\vec{a}_0 = (M_1)_{1*} - (A^T)_{1,1}^{(1,3)} - (A^T)_{2,1}^{(1,3)} = 010 - 000 - 000$ and $g_0 = 6$. Solving the equation $6 + 7x \equiv 1 \pmod 4$ yields $x = 1$ and $k = 17$. That is, $M_1 = (A_1)_{3,17}^{(3,3)}$. ($A_1$ is described in Appendix (2)).

From $(A_1)_{1,17}^{(16,3} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \end{pmatrix}$ we can determine $\vec{a}_1 = (M_2^T)_{1*} - (A_1)_{1,1}^{(1,3)} - (A_1)_{2,1}^{(1,3)} = 100 - 010 - 010$ and then $g_1 = 7$. Solving the equation $7 + 7x \equiv 17 \pmod{32}$ yields $x = 6$ and $k = 81$. That is, $M_2^T = (A_2)_{3,81}^{(4,3)}$ hence $M_2 = (A_2)_{81,3}^{(3,4)}$ (A_2 is described in Appendix (3)).

From the Appendix we find $(A_2)_{3,1}^{4,256)}$ and hence we know $(A_2^T)_{1,3}^{(256,4)}$. We have $\vec{a}_2 = (M_3)_{1*} - (A_2^T)_{2,3}^{(1,4)} - \cdots - (A_2^T)_{80,3}^{(1,4)} = 1000 - \underbrace{0011}_{\text{1st row}} - \underbrace{0011}_{\text{2nd row}} - \cdots - \underbrace{1010}_{\text{80th row}} = 0010$ and $g_2 = 2$. Solving the equation $2 + 15x \equiv 3 \pmod{16}$ yields $x = 15$ and $k = 243$. That is, $M_3 = (A_3)_{81,243}^{(4,4)}$.

Appendix

(1) A $(4,128;3,3)$-array

Column 1 to 64:

0000000000000000001110100111010011101001110100111010011101001110
0000101000110101001100000100000111100011111001101010110101111011
0001000100001011001010110111111111111000110110001011011001000101
1110010011000001110111101011010100001101000100100100001110001111

Column 65 to 128:

1001110100111010011101001110100111010011101001110100111010011101
1001011100001111011111101101110011011001100100100010010101000
1000110000110001011001011110001011000010101011000101111110010110
0111100111111011100100000010100000110111011001101010101001011100

(2) A $(16,32;3,3)$-array

$$
\begin{pmatrix}
00000011101001110100111010011101\\
00010010101101100101111110001100\\
00000011101001110100111010011101\\
11101101010010011010000001110011\\
00000011101001110100111010011101\\
00010010101101100101111110001100\\
00110000100101000111110110101110\\
10001011001011111100011000010101\\
11111100010110001011000101100010\\
11101101010010011010000001110011\\
10101001000011011110010000110111\\
01000111111000110000101011011001\\
11111100010110001011000101100010\\
11101101010010011010000001110011\\
01100101110000001001010001111011\\
11011110011110101001001101000000
\end{pmatrix}
$$

(3) A (16, 256; 4, 3)-array

Columns 1 through 48:

000000000000000000000000000000000011101001110100
000000111010011101001110100111010011100111010011
000100010001000100010001000100010010101101100101
000100101011011001011111100011000010100011000010
111111111111111111111111111111111100010110001011
111111000101100010110001011000101100011000101100
111011101110111011101110111011101101010010011010
110111100111101010010011010000000111001000001110
010101010101010101010101010101011101111100100001
101010010000110111100100001101111001001101111001
010001000100010001000100010001000111111000110000
111011010100100110100000011100111101011100111101
101010101010101010101010101010101001000011011110
010101101111001000011011110010000110110010000110
101110111011101110111011101110111000000111001111
110111100111101010010011010000000111001000001110

Columns 49 through 96

111010011101001110100111010011101001110100111010
101001110100111010100100111010011101001110100111
111110001100001010110110010111111000110000101011
101101100101111110110101111110001100001010110110
000101100010110001011000101100010110001011000101
010110001011000101011011000101100010110001011000
000001110011110101001001101000000111001111010100
011110101001001101111001001101000000111001111010
101111001000011011110010000110111100100001101111
000011011110010000001110010000110111100100001101
101011011001011111100011000010101101100101111110
010010011010000001001010000001110011110101001001
010000110111100100001101111001000011011110010000
111100100001101111110001101111001000011011110010
010100100110100000011100111101010010011010000001
011110101001001101111001001101000000111001111010

Columns 97 through 144

011101001110100111010011101001110100111010011101
011101110100111010011101001110100100110100111010
011001011111100011000010101101100101111110001100
011001100101111110001100001010110101110000101011
100010110001011000101100010110001011000101100010
100010001011000101100010110001011011001011000101
100110100000011100111101010010011010000001110011
101010101001001101000000111001111001000011100111
001000011011110010000110111100100001101111001000
110111011110010000110111100100001110011110010000
001100001010110110010111111000110000101011011001
100110011010000001110011110101001010001111010100
110111100100001101111001000011011110010000110111
001000100001101111001000011011110001100001101111
110011110101001001101000000111001111010100100110
101010101001001101000000111001111001000011100111

Columns 145 through 192

001110100111010011101001110100111010011101001110
011101001110100111101010011101001110100111010011
001010110110010111111000110000101011011001011111
011001011111100011111011011001011111100011000010
110001011000101100010110001011000101100010110001
100010110001011000010101100010110001011000101100
110101001001101000000111001111010100100110100000
101010010011010000110111101010010011010000001110
011011110010000110111100100001101111001000011011
110111100100001101000000110111100100001101111001
011111100011000010101101100101111110001100001010
100110100000011100000100100110100000011100111101
100100001101111001000011011110010000110111100100
001000011011110010111111001000011011110010000110
100000011100111101010010011010000001110011110101
101010010011010000110111101010010011010000001110

Columns 193 through 240

100111010011101001110100111010011101001110100111
100111101001110100111010011101001101000000000000
100011000010101101100101111110001100001010110110
100011111000110000101011011001011100000100010001
011000101100010110001011000101100010110001011000
011000010110001011000101100010110010111111111111
011100111101010010011010000001110011110101001001
010000110100000011100111101010010000110111011101
110010000110111100100001101111001000011011110010
001101000011011110010000110111100111101010101010
110110010111111000110000101011011001011111100011
011100000111001111010100100110100011111011101110
001101111001000011011110010000110111100100001101
110010111100100001101111001000011000010101010101
001001101000000111001111010100100110100000011100
010000110100000011100111101010010000110111011101

Columns 241 through 256

0100111010011101
0000000000000000
0101111110001100
0001000100010001
1011000101100010
1111111111111111
1010000001110011
1101110111011101
0001101111001000
1010101010101010
0000101011011001
1110111011101110
1110010000110111
0101010101010101
1111010100100110
1101110111011101

References

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

[2] J. Burns and C.J. Mitchell, Coding schemes for two-dimensional position sensing, *Cryptography and Coding III*, edited by M. Ganley, Oxford University Press (to appear).

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

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

[5] I.J. Good, Normally recurring decimals, *J. London Math. Soc.*, **21** (1946), 167–169.

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

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

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

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

[10] C.J. Mitchell and K.G. Paterson, Decoding Perfect Maps, to appear in *Designs, Codes and Cryptography.*

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

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