1-1-2013

Fast operations for certain two alphabet circulant matrices

Gene Awyzio
University of Wollongong, gene@uow.edu.au

Jennifer Seberry
University of Wollongong, jennie@uow.edu.au

Fast operations for certain two alphabet circulant matrices

Abstract

In order to efficiently compute some combinatorial designs based upon circulant matrices which have different, defined numbers of 1's and 0's in each row and column we need to find candidate vectors with differing weights and Hamming distances. This paper concentrates on how to efficiently create such circulant matrices. These circulant matrices have applications in signal processing, public key codes and spectography

Keywords

circulant, alphabet, operations, two, fast, certain, matrices

Disciplines

Engineering | Science and Technology Studies

Publication Details

Awyzio, G. & Seberry, J. (2013). Fast operations for certain two alphabet circulant matrices. In H. R. Arabnia, G. A. Gravvanis, G. Jandieri, A. M. G. Solo & F. G. Tinetti (Eds.), Proceedings of the 13th International Conference on Scientific Computing (CSC) (pp. 234-238). United States: CSREA Press.

Fast Operations for Certain Two Alphabet Circulant Matrices

G Awyzio and J Seberry
Centre for Information Security Research,Faculty of Engineering and Information Sciences,
University of Wollongong, NSW, 2522,Australia

Abstract— *In order to efficiently compute some combinatorial designs based upon circulant matrices which have different, defined numbers of* $1s$ *and* $0s$ *in each row and column we need to find candidate vectors with differing weights and Hamming distances. This paper concentrates on how to efficiently create such circulant matrices. These circulant matrices have applications in signal processing, public key codes and spectography.*

Keywords: algorithm, periodic autocorrelation function, cross correlations , Hamming distance, circulant matrices.

1. Introduction

This paper uses the mapping of two-alphabet (for example $\{\pm 1\}$, {x,y}, {0,1}) circulant matrices to binary equivalents.

Interest in binary arrays or matrices with a constant row sums is of continuing study in combinatorics. For example a $BIBD(v, b, r, k, \lambda)$ which can be defined as a $v \times b$ matrix which has constant row sum r and constant column sum k and distinct inner product of rows λ and variations of this design are used in statistical and medical experiments. Recently such circulant matrices have been used to construct asymmetric public key codes. Sequences with elements $0, \pm 1$, very small periodic or non-periodic autocorrelation function and small cross correlation function are also of considerable interest in signal processing. Powers of circulant matrices arise in spectography but in a different form.

2. Definitions and Preliminaries

Since this paper uses the mapping of two-alphabet circulant matrices to binary equivalents the fast construction of binary candidates that match specified parameters is crucial. We discuss how some operations relevant to the multiplication of matrices, an $O(n^3)$ process, can in some cases be converted to a linear process by using architectural level operations.

We first clarify the notation and processes that we will use in our analysis.

2.1 Circulant and Type 1 Matrix Basics

Because it is so important for the rest of our work we now spend a little effort to establish why the properties we will require for binary circulant matrices are so important.

We define the *shift matrix, T* of order n by

$$T = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}. \tag{1}$$

So any circulant matrix, of order n and first row $x_1,\ x_2, \cdots, x_n$, that is,

$$\begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & & & & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix} \tag{2}$$

can be written as the polynomial

$$x_1 T^n + x_2 T + x_3 T^2 \cdots x_n T^{n-1}.$$

We now note that polynomials commute, so any two circulant matrices of the same order n commute.

Mathematically we have that:

Definition 1: A *circulant matrix* $X = (x_{ij})$ of order n is a matrix which satisfies the condition that

$$x_{ij} = x_{1,j-i+1} \tag{3}$$

where $j - i + 1$ is reduced modulo n [1].

Thus any circulant matrix $X = (x_{ij})$ of order n can be defined by

$$x_{ij} = x_{i+1,j+1} = x_{1,j-i+1},$$

that is, the first row is enough to specify the whole matrix. In all cases the sums are reduced modulo n so that n is written n, $n + 1$ is written as 1 and so on.

In all our definitions of circulant matrices we have assumed that the rows and columns have been indexed by the order, that is for order n, the rows are named after the integers 1, 2, $\cdots$, n and similarly for the columns. The internal entries are then defined by the first row using a 1:1 and onto mapping $f : G \to G$. However we could have indexed the rows and columns using the elements of a group G, with elements g_1, g_2, $\cdots$, g_n. Loosely a *type one matrix* will then be defined so the (ij) element depends on a 1:1 and onto mapping of $f(g_j - g_i)$ for type 1 matrices which occur in construction of combinatorial designs. We

use additive notation, but that is not necessary. Seberry-Wallis and Whiteman [2] have shown that circulant and type 1 matrices can be used interchangeably in the enunciations of theorems. This can be used to explore similar theorems in more structured groups.

2.2 Periodic Autocorrelation Function and Cross Correlation Function

These terms, which arise in signal processing, are usually thought of differently by mathematicians.

Definition 2: The $PAF(j)$ or *periodic auto correlation function* of a sequence $\{x_{11},\ x_{12},\ \cdots,\ x_{1n}\}$ (that is the first row of a circulant matrix $X = (x_{ij})$) of order n is given, for $j = 1,\ \cdots, n$, by

$$PAF(j) = \sum_{i=1}^{n} (x_{1i}x_{1,i+j})$$

Definition 3: The $PAF(X)$ or $PAF(j,k)$ or *periodic auto correlation function* of the two rows, j and k, of a circulant matrix $X = (x_{ij})$ of order n is defined as

$$PAF(j,k) = \sum_{i=1}^{n} (x_i x_{k-j+i}).$$

We note that this is exactly the same as the inner product of rows j and k of the matrix $XX^{\top}$.

Definition 4: $CPAF$ or *cross correlation function* of two rows j of a circulant matrix $X = (x_{ij})$ of order n and k of a circulant matrix $Y = (y_{ij})$ of order n is defined as

$$CPAF(j,k) = \sum_{i=1}^{n} (x_i y_{k-j+i}).$$

This is also written as $CPAF(X,Y)$.

In signal processing we would consider matrices with elements ± 1 but clearly each circulant matrix with these two elements uniquely maps to a binary circulant matrix.

3. General Properties of Circulant Matrices

Remark 1: Each row of a circulant ± 1 matrix can be considered as an integer, uniquely, by replacing the elements -1 by zero and converting the sequence to decimal. Thus a circulant matrix of order n can be represented by an integer of size the least integer greater than $ln_2\ n$. This means any sequence we would consider can be represented by one word of storage. For example for the length $n = 7$ the integer $106 = 64 + 32 + 8 + 2$ represents the row

```
1  1 -1  1 -1  1 -1,
        or
1  1  0  1  0  1  0 =106.
```

3.1 Complexity of Squaring a Matrix

We note

Theorem 1: Any circulant matrix, $X = (x_{ij})$,of order n can be defined by its first row. Writing X in terms of the shift matrix (1) we have

$$X^2 = (x_{11}I + x_{12}T + x_{13}T^2 + \cdots + x_{1,n}T^{n-1})^2$$

so squaring can be achieved by $O(\frac{n(n+1}{2})$ operations.

We note X^2 contains at most n distinct values which are the new first row.

Similarly the product of two order n, circulant matrices, takes $O(\frac{n(n+1}{2})$ operations and contain at most n distinct values which are the new first row.

In future work, we will show that, using our construction for candidates, we can make this a linear number of operations.

Example 1: Let X be a 5×5 matrix with first row elements a, b, c, d and e:

$$X = \begin{bmatrix} a & b & c & d & e \\ e & a & b & c & d \\ d & e & a & b & c \\ c & d & e & a & b \\ b & c & d & e & a \end{bmatrix}.$$

As the (j,k) element of X^2 is equal to the $(1, k-j+1)$ element, because the answer is also a circulant matrix: the elements of X^2 are

$$(1,1) = (2,2) = (3,3) = (4,4) = (5,5) = a^2 + 2be + 2cd$$
$$(1,2) = (2,3) = (3,4) = (4,5) = (5,1) = 2ab + 2ce + d^2$$
$$(1,3) = b^2 + 2ac + 2be$$
$$(1,4) = e^2 + 2ad + 2bc$$
$$(1,5) = c^2 + 2ae + 2bd$$

These 5 values are the first row of X^2 and give all the entries of X^2 all the possible PAFs.

We also note that

Theorem 2: The PAF of a sequence of n elements contains only $\frac{n+1}{2}$ distinct entries.

Example 2: Let X be a 5×5 matrix with first row elements a, b, c, d and e:

$$X = \begin{bmatrix} a & b & c & d & e \\ e & a & b & c & d \\ d & e & a & b & c \\ c & d & e & a & b \\ b & c & d & e & a \end{bmatrix}.$$

Then

$$PAF(1,1) = PAF(k,k) = a^2 + b^2 + c^2 + d^2 + e^2,$$
$$PAF(1,2) = \cdots = PAF(n,1) = ae + ba + cb + de + ed,$$
$$PAF(1,3) = \cdots = PAF(n-1,1) = ad + be + ca + db + ec.$$

These 3 values are all the possible PAFs.

4. Architectural Level Operations

Because we depend upon the speed of bit level manipulation we shall now discuss the advantages of architectural level operations in modern computer architectures. Historically architectural level operations afforded speed advantages for all operation in a high level language including additions and subtractions. Modern architectures perform additions almost as fast as bitwise manipulations but still take more cycles to perform a multiplication.

The speed gain is found by being able to manipulate multiple pieces of data with a single instruction. For instance if we wished to find the inner product of two vectors stored in an array we would need to read each entry multiply them and add each result. With two binary vectors the inner product can be found by performing a bitwise XOR and bit count to find the Hamming distance of the two vectors. Knowledge of this result can be used to find the inner product of the two vectors using a single subtraction and a bitwise shift of the Hamming distance. Thus for larger length vectors significant savings in operations can be found using architectural level operations.

5. Integers to Bits

We now move from representing the circulant matrices as vectors of length n and elements from a 2-ary alphabet to first using their binary equivalent and then noting that the binary vector for lengths ≤ 32 can be stored as a single binary word of 32 bits. We now establish the requirements that reflect our specified parameters as above in single words. That is we work to pre-specify the length and weight using architectural level operations.

Example 3: Let X be $\{x, y\}$ be a sequence of length 10:

```
X = {  x  x  x  y  y  y  x  y  x  y}.
```

It can be represented as the binary number

$$1110001010_{base\ 2} = 187_{base\ 10}.$$

In general we are looking for binary numbers of length ℓ and weight h, with pre-specified properties such as the sum of the components, multiplication properties, the PAF and/or inner products. A naive search for candidates with such properties can be achieved in the binary domain by iterating through all the binary numbers and testing the weight of each number. This would require that 2^ℓ numbers are tested for conformity to the required parameters.

In our approach we start by assuming that candidates for a previous weight $h-1$ and length ℓ are already known: then new potential candidates with weight h can be rapidly found using an iterative procedure.

For each candidate of size $h-1$ we need to find the highest set bit (HSB) of the candidate which requires at most $\ell-(h-1)$ tests. Once the position of the HSB has been determined then there are at most $\ell-h$ operations required to set the higher order bits. Thus the expansion from weight $h-1$ to h takes $(2\times\ell-2\times h+1)$.

Algorithm 1 Construction of Binary Candidates

Step 1: Read the first candidate of a given length (ℓ) and weight one less ($h-1$) than the desired weight (h) from file
Step 2: while we have have candidates remaining in this file
Step 3: find the HSB in current candidate
Step 4: set a single bit for each position between this bit and the length of the candidate

Example 4: Suppose we have starting candidate

```
0 0 0 1 1 0
```

of length 6 and weight 2. This can be used directly to find three candidates with a weight of 3 and length 6

```
0 0 1 1 1 0
0 1 0 1 1 0
```

and;

```
1 0 0 1 1 0
```

Which each have one additional bit set above the HSB of the seed candidate. By iterating through the ten candidates of weight 2 we can easily find the ten candidates of length 5 and weight 3 in this manner.

Thus considering the complete candidate set with a weight of 2 we obtain;

```
0 0 0 0 1 1,
0 0 0 1 0 1,
0 0 1 0 0 1,
0 1 0 0 0 1,
1 0 0 0 0 1,
0 0 0 1 1 0,
0 0 1 0 1 0,
0 1 0 0 1 0,
1 0 0 0 1 0,
0 0 1 1 0 0,
0 1 0 1 0 0,
1 0 0 1 0 0,
0 1 1 0 0 0,
1 0 1 0 0 0,
1 1 0 0 0 0
```

Which leads to the generation of 20 candidates of the same length and one extra bit set.

```
0 0 0 1 1 1,
0 0 1 0 1 1,
0 1 0 0 1 1,
1 0 0 0 1 1,
0 0 1 1 0 1,
0 1 0 1 0 1,
1 0 0 1 0 1,
0 1 1 0 0 1,
```

```
1 0 1 0 0 1,
1 1 0 0 0 1,
0 0 1 1 1 0,
0 1 0 1 1 0,
1 0 0 1 1 0,
0 1 1 0 1 0,
1 0 1 0 1 0,
1 1 0 0 1 0,
0 1 1 1 0 0,
1 0 1 1 0 0,
1 1 0 1 0 0,
1 1 1 0 0 0
```

Extending these 20 candidates results in 15 candidates with 4 bits set. These candidates are the inverse of the 15 candidates with 2 bits set. Thus we only need to generate the candidates up to a weight of the integer part of $\frac{\ell+1}{2}$. As the length of the candidate set increase it is seen that the number of candidates will match a row on Pascal's triangle.

Additionally the search space can be reduced further by recognizing that since $3 \equiv 5 - 2$ these candidates could be found directly by inverting all of the candidates of weight 2. Thus we only need to generate the candidates up to a weight of the integer part of $\frac{\ell+1}{2}$.

The number of candidates for each weight (h) of a given length (ℓ) follows the entries for a row in Pascal's triangle meaning that if we wish to find candidates with the integer part of $\frac{\ell+1}{2}$ bits set there would be at worst $\frac{2^\ell}{2}$ operations required to find these candidates. However, in many cases the weight of candidates is either close to 0 or close to ℓ and thus significant savings can be made in not having to test all numbers.

Example 5: If we consider the case where we require candidates of length 13 with 4 bits set then we would need to find all candidates with $1,\ 2,\ 3$ and 4 bits set. The total number of candidates would be $13 + 78 + 186 + 715 = 992$ candidates $\sum_{h=1}^{4} \binom{13}{h}$ that are discovered using this method. This is of order $2(\ell - h + 1)$ which is a significant saving over the $2^\ell = 2^{13}$ numbers that would be tested using the naive approach. For longer length candidates the savings as the required weight deviates from $\frac{\ell}{2}$ becomes even more significant.

5.1 Circulation in the Binary Domain

Construction of a circulant matrix from these binary candidates can be performed using bitwise operations upon the first row to shift the entire row by one bit and move the lowest bit to the highest bit position. Many of the operation usually performed upon these circulant matrices can also be done in the binary domain using architectural level operations.

This requires one operation to perform each of the following four steps for each row, copy $row(i)$ to $row(i+1)$, test and set the variable bit, shift $row(i+1)$ right by one bit and set MSB of $row(i+1)$ to the value of bit. Thus each additional row of the circulant matrix requires four operations to create. Therefore the entire circulant matrix can be computed in $4 \times \ell$ operations. It is noted that any row of the circulant matrix $row(j)$ can be directly computed from the first row by testing the lowest j bits and saving them, shifting the first row right by j bits (j operations at most) and copying the original lowest j bits to the top j bits of the shifted row.

Algorithm 2 Circulation of a Binary Matrix

Input: The first row of the circulant matrix of length and order ℓ in binary form ($row(1)$)

for $i = 1$ **to** ℓ
Copy $row(i)$ to $row(i+1)$
Set variable bit to the LSB of $row(i+1)$
Shift $row(i+1)$ right by one
Set MSB of $row(i+1)$ to bit

5.2 Binary Inner Products

We can take advantage of the fact that when integer matrices under consideration contain ± 1 only to reduce the computation of inner product vectors in the binary domain. When any two values are the same in a first row circulation then they will result in a $+1$ in determining the inner product and when they are different they will result in a -1.

In the integer domain the inner product is determined as

Theorem 3: $IP = a_1^{1^{st}} * a_1^{2^{nd}} + a_2^{1^{st}} * a_2^{2^{nd}} + \cdots + a_\ell^{1^{st}} * a_\ell^{2^{nd}}$

which results in q negative results and $p = (n-q)$ positive results.

Thus the inner product can be determined to be $p - q$

$$IP = (n - q) - q = n - 2q \tag{4}$$

In the binary domain the location of bits that are different (equivalent to a -1 in the integer domain) can be achieved with a bitwise XOR of two rows of the circulant matrix. The Hamming distance of the resultant XOR on the two vectors is equivalent to the number of -1s in the integer domain. The binary inner product can be determined as follows

In the binary domain the number of negative bits (weight) in the inner product of two rows can be determined by performing a bitwise XOR on two vectors (rows of the circulant matrix). The integer inner product can be calculated directly from knowledge of the number of negative bits (q) and the length of the two vectors (ℓ)

$IP = \ell - 2 \times q$

In Example 2 we showed that the circulant inner products obtained from an ℓ matrix has only $\frac{\ell+1}{2}$ distinct values. To find the inner product in the integer domain requires ℓ muliplications and $\ell - 1$ additions for each inner product. Thus in total it would require $\frac{\ell(\ell-1)}{2}$ multiplications and

$\frac{\ell(\ell-2)}{2}$ additions. Now we consider the case where we have moved the first row into a binary word.

Algorithm 3 Binary Inner Product

Input: The first row of the circulant matrix of length and order ℓ in binary form ($row(1)$)

for $i = 2$ **to** $\frac{\ell+1}{2}$
Circulate $row(i)$ to $row(i+1)$
$Hamming \quad Distance =$
$weight\ of\ (XOR(row(1), row(i))$
$IP(i) = \ell - (Hamming\ distance \times 2)$

[Note: multiplication by 2 is a shift left operation at the architectural level]

This means that, ignoring the circulation operations, we have one operation to find the XOR of two rows, one operation to find the weight, and two operations to calculate the inner product of the two rows. This means we have four operations to find the inner product of any two rows of the circulant matrix regardless of the value of ell. To find all the inner products we need to calculate the inner product for $\frac{\ell-1}{2}$ rows. Thus the total number of calculations required to find all inner products of a circulant matrix of order ℓ is $2 \times (\ell - 1)$.

6. Conclusion

We have introduced architectural level operations for various two alphabet circulant matrix operations. We have shown that this approach reduces the complexity in each case we have studied. We intend to further use this approach to search for $BIBD$s, sequences with elements $0, \pm 1$, very small periodic or non-periodic autocorrelation function and small cross correlation function and their applications.

Acknowledgment

The authors would like to thank Bob Brown, Ian Piper, Angela Piper and Graham Williams for their advice and assistance.

References

[1] Jennifer Seberry Wallis, Hadamard matrices, in W. D. Wallis, Anne Penfold Street and Jennifer Seberry Wallis, *Combinatorics: Room Squares, Sum-Free Sets and Hadamard Matrices,* Lecture Notes in Mathematics, Springer Verlag, Berlin, 1972.

[2] Jennifer Wallis and A. L. Whiteman, Some classes of Hadamard matrices with constant diagonal, *Bull. Austral. Math. Soc.*, 7, (1972), 233–249.