Costas arrays searching algorithm based on vectors

Dianhong Zhu[a], Jianguo Yao

College of Telecommunications & Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing, 210003, China

Abstract. Costas arrays have been widely applied in wireless communication, radar and cryptography as they have the ideal auto-ambiguity properties. Costas arrays are constructed mostly using the algebraic method, or they can be found by exhaustive search. This paper first introduces the Costas arrays and their properties and then proposes the vector-based algorithm to make depth-first search for Costas arrays since the exhaustive search algorithm based on check matrix is flawed. The experiment results indicate that the algorithm based on vectors has higher search speed on a single CPU when compared with the exhaustive searching algorithm based on check matrix.

1 Introduction

In the 1960's, John P. Costas proposed the concept of frequency hopping signals characterized by ideal auto-ambiguity when he attempted to design sonar signals to improve the performance of sonar systems [1]. A frequency hopping array with an ideal autocorrelation function is also called Costas array. In the 1980's, S.W. Golomb constructed the Costas arrays based on the finite field theory[2]. Another way to obtain the Costas arrays is exhaustive search through the computer program[3].

Let $\boldsymbol{P}$ be an order n permutation matrix. If the maximum value of the side lobe of the auto-correlation function $R(\tau,d)$ is equal to 1, the matrix P is called an order n Costas array[4]. Dots and blanks are usually used to represent 1 and 0 in the permutation matrix, respectively. For example, what is shown in Figure 1 is a Costas array and its auto-correlation is presented in Figure 2. We find that it has ideal auto-correlation properties, the main lobe is high and sharp while the side lobes are low and flat[5]. The Costas arrays are not only applied to radar[6], but also have a wide range of applications in wireless communication and cryptography[7]. Therefore, it is of great significance to conduct research on the construction of Costas arrays and the algorithm for searching Costas arrays.

A Costas arrays searching algorithm based on a check matrix is proposed in [8]. The backtracking method[9] is used to exhaustively enumerate order n permutation matrix $\boldsymbol{P}$, when any column in the check matrix $\boldsymbol{D}$ does not have the same element, then the order n permutation matrix $\boldsymbol{P}$ is a Costas array. The algorithm has the following disadvantages:

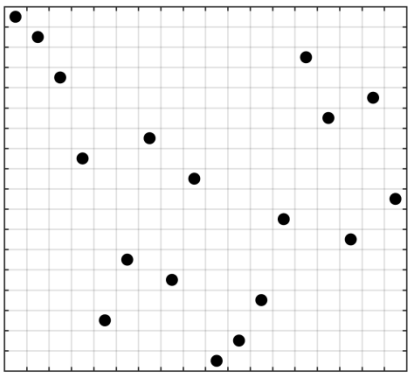

Figure 1. A Costas array of order 18

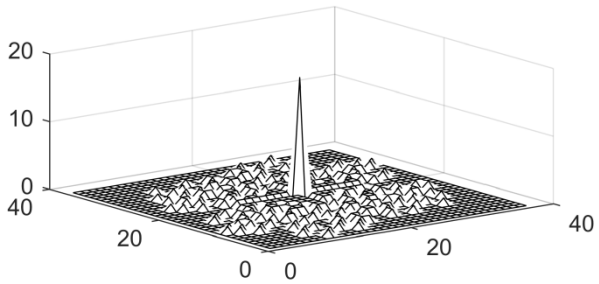

Figure 2. The auto-correlation function of order 18 Costas array

- The backtracking method is used to traverse $n!$ permutation matrices. When the order n is high, the number of permutation matrices is large, and the computation increases dramatically.

- The check matrix is used to determine whether $\boldsymbol{P}$ is a Costas array. The performance is not good if the order is high.

[a] Corresponding author: dianhongzhu@163.com

Given the shortcomings of the searching algorithm based on check matrix and the properties of Costas arrays, a Costas arrays searching algorithm based on vectors is proposed in this paper. Firstly, the determination of permutation matrix and whether it meets the criterion of Costas arrays contributes to the elimination of the shortcomings of the backtracking method that traverses all permutation matrices, hence reducing unnecessary calculations. Secondly, the optimization of Costas arrays criterion reduces the time complexity of the algorithm. Finally, the experiment results indicate that the algorithm proposed in this paper accelerates the searching of the Costas arrays on a single CPU.

2 Permutation matrix

Each permutation of n elements corresponds to a unique permutation matrix. Let f be a permutation of n elements:

$$f:\{1,\cdots,n\}\rightarrow\{1,\cdots,n\}$$

Represented in two-line form:

$$\begin{pmatrix}1 & 2 & \cdots & n\\ f(1) & f(2) & \cdots & f(n)\end{pmatrix}$$

Its corresponding $n \times n$ permutation matrix is P_f, where the entries in row i are all 0 except that a 1 appears in column $f(i)$. That can be written as:

$$\boldsymbol{P_f} = \begin{bmatrix}\boldsymbol{e_{f(1)}}\\ \boldsymbol{e_{f(1)}}\\ \vdots\\ \boldsymbol{e_{f(1)}}\end{bmatrix}$$

where $e_{f(j)}$, a standard basis vector, denotes a row vector of length n with 1 in the column j and 0 in every other position.

From the above definition of the permutation matrix, it can be seen that for the permutation matrix, there are only a single 1 in each row and column.

In the algorithm proposed in this paper, an order n permutation matrix is represented by a one-dimensional array of length n. Each element represents a row, and the value of the element represents the column value of 1 in the row. This is sufficient to fully represent the permutation matrix and avoid any scanning of rows. For example:

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

It can be expressed as

$$4\quad 2\quad 3\quad 1$$

The position of the column where the row before the row i of the permutation matrix has been used is represented by the variable dotUsed. The column value j of the row i to be filled in is represented by the variable dot, where $j = 1 \ll i$. Computes the bitwise and of the variable bitUsed and dot, and the obtained result is judged so that the row i can satisfy the correct position of 1 defined by the permutation matrix. A bitwise or of the variable dotUsed and dot results in the position of the column where the row before row i+1 has been used. The permutation matrix can be obtained by recursion. When searching the permutation matrix, whether it satisfies the Costas arrays criterion is judged, and if it does not satisfy the condition, there is no need to continue downward recursion. Therefore, searching for the Costas arrays does not need to exhaust the $n!$ probability of the order n matrix $\boldsymbol{P}$, which avoids the use of backtracking method to traverse all permutation matrices and reduces unnecessary calculations.

3 Costas arrays judgement criterion

Costas arrays have many properties. For example, when a Costas array is rotated 90°, and flipped horizontally, vertically, and diagonally, it is still a Costas array [10, 11]. This property can be used to optimize the criterion of the Costas arrays. If a permutation matrix is a Costas array and its symmetric array is also a Costas array, there is no need to judge all cases.

Another property of the Costas arrays is mentioned in [12, 13]. Let C={1, 2, …, n} be a set of n elements, where $n\in\mathbb{N}$. f is a permutation on C, and f is Costas permutation if and only if it satisfies the following conditions:

$$\forall i,j,k,\text{where } i,j,i+k\in C:\\ f(i+k)-f(i)=f(j+k)-f(j)\Rightarrow \\ i=j\ or\ k=0 \qquad (1)$$

The Costas array is the permutation matrix that corresponds to the Costas permutation.

Customarily, 1 is represented as dot in the permutation matrix and 0 is represented as blank. From the perspective of vectors, equation (1) can be seen as the vector ($f(j)$-$f(i)$, j - i) between the dots of the column i and the column j of the Costas array, which is not repeated between any other pairs of dots, where $1\leq i,j\leq n$. From a geometrical point of view, equation (1) means that four dots that are not on a straight line cannot form a parallelogram, as the array shown in Figure 3, and four dots form a parallelogram, so this array is not a Costas array; four dots on a straight line cannot form two pairs of equidistant dots, as the array shown in Figure 4, and there are two pairs of equidistant dots, so it is not a Costas array; three dots on a straight line cannot be equidistant, as the array shown in Figure 5, three dots are in a straight line and equidistant, so it is not a Costas array.

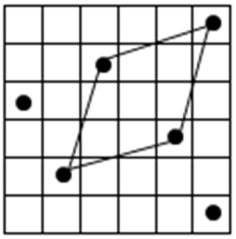

Figure 3. Four dots form a parallelogram

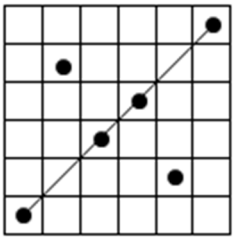

Figure 4. Four dots form two pairs of equidistant dots

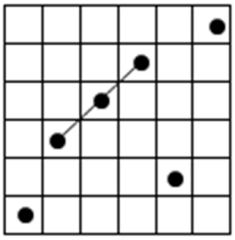

Figure 5. Three dots on a straight line be equidistant

Based on the understanding of the above properties, the vector is introduced into the judgement of the Costas arrays. The vector is used to judge whether a permutation matrix is a Costas array. The vectors for the dot on the row *n* and all previous rows are calculated and the position is recorded in the corresponding position in the array vectorLocation. The purpose of this array is to ensure that any particular vector can only occur once. If any vector is already in the array, indicating that the permutation matrix cannot constitute a Costas array, the dot in current row is invalid and it will be skipped.

4 Algorithm idea

Depth-first search is to select a vertex in a given graph as the starting point of the search, then search for unmarked adjacency points, from which the recursive search is made until all the vertices have been searched.

Based on the idea of depth-first search, each possible position is tried from the first row of the permutation matrix and recursively to the next row to select the point that satisfies the permutation matrix condition. At the same time, it is judged whether or not it meets the criterion of Costas arrays. If not, another dot of this row is selected again and it is judged whether it satisfies the above two requirements. Otherwise, it continues recursively to the next row until all rows have been searched. If all rows are verified, the array is a Costas array. So it returns 1. Finally all the Costas arrays that meet the conditions are added up. The final result of recursion is the total number of Costas arrays.

The specific flow of the algorithm proposed in this paper is as follows:

Input: the order of Costas arrays

Output: the total number of Costas arrays of specific order and time consumed

Step:

```
1.    n=0, count=0;
2.    startTime=clock();
3.    if (n == order)
4.      count = 1;
5.    else
6.      for (i = 0; i < order; i++)
7.        if (row before line n has no dot in this column)
8.          dotLocation[n] =i;
9.          clear = 1;
10.         for (j = 0; j < n; j++)
11.           calculate the vector;
12.           if(the vector exists in the array of
              vectorLocation)
13.             clear=0;
14.           record the vector in the vectorLocation array;
15.         endfor;
16.         if(clear=1)
17.           n+1;
18.           count+= repeat step 3;
19.           remove the position marked in the
              vectorLocation in step 14;
20.     endfor;
21.   endTime=clock();
22. duration=endTime-startTime;
```

5 Experiment and result analysis

Experimental environment: The operating system is Windows 7 64-bits, the processor is Intel(R) Core(TM) i5-4460 3.2GHz 4 cores, the memory is 8.0 GB, the integrated development environment is CLion, and the algorithm is implemented using C language. The experiments with the algorithm presented in [8] and the algorithm proposed in this paper were performed respectively. The results are shown in Table 1 and Table 2, respectively.

Table 1. The result of the algorithm in [8]

Order	Number of Costas arrays	Searching time(s)
1	1	0.000
2	2	0.000
3	4	0.000
4	12	0.000
5	40	0.000
6	116	0.000
7	200	0.000
8	444	0.026
9	760	0.119
10	2160	1.163
11	4368	16.979
12	7852	234.851
13	12828	4122.401

Table 2. The result of the algorithm proposed in this paper

Order	Number of Costas arrays	Searching time(s)
1	1	0.000
2	2	0.000
3	4	0.000
4	12	0.000
5	40	0.000
6	116	0.000
7	200	0.000
8	444	0.014
9	760	0.044
10	2160	0.131
11	4368	0.579
12	7852	3.415
13	12828	19.419
14	17252	111.467

15	19612	654.439
16	21104	3589.334
17	18276	20072.026
18	15096	112225.982
19	10240	614172.357

Comparing Table 1 and Table 2, we can see that the algorithm presented in [8] and the algorithm proposed in this paper can correctly calculate the number of Costas arrays of a given order. However, as can be seen from Figure 6, the searching speed of the algorithm proposed in this paper is significantly increased. Therefore, the experimental dataset verifies the effectiveness of the algorithm based on vectors in increasing the speed of arrays search.

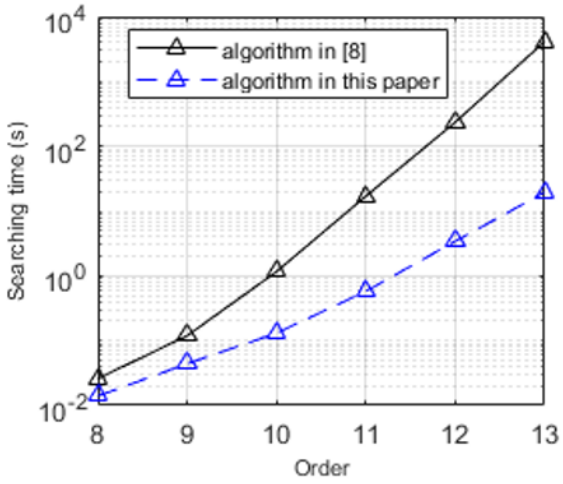

Figure 6. Speed comparison of Costas arrays searching algorithm

In addition, it can be seen from Figure 7 that the number of Costas arrays does not increase with the increase of the order. When the order is 16, it reaches the peak and then gradually decreases. When the order is changed from n to n+1, the calculation is increased by about 6 times. Therefore, this experiment only searches the Costas arrays whose order is not greater than 19, and it can be roughly estimated that when the order is greater than 19, searching the Costas arrays will take several months.

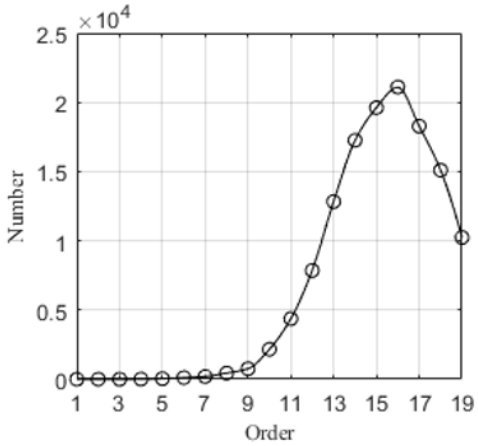

Figure 7. Order of the Costas arrays and corresponding number

6 Conclusions

This paper proposes a vectors-based Costas arrays searching algorithm as the existing Costas arrays searching algorithm has a high time complexity and a low searching speed for higher order. The algorithm introduces vectors and determines whether the array is a Costas array by comparing vectors. Furthermore, it overcomes the drawbacks of the backtracking method that exhausts all the permutation matrices. The results of the experiment that compares the two algorithms show that the algorithm proposed in this paper is effective in increasing the search speed of Costas arrays dramatically.

In addition, we also find that the number of Costas arrays does not increase with the increase of the order, and it reaches the peak when the order is 16. When the order is increased by 1, the computation of Costas arrays search increases by approximately 6 times.

References

1. G. Gong, T. Helleseth and P. V. Kumar, "Solomon W. Golomb—Mathematician, Engineer, and Pioneer," in IEEE Transactions on Information Theory, vol. **64**, no. 4, pp. 2844-2857(2018).
2. S. W. Golomb and H. Taylor, "Constructions and properties of Costas arrays," in Proceedings of the IEEE, vol. **72**, no. 9, pp. 1143-1163(1984).
3. J. K. Beard, J. C. Russo, K. G. Erickson, M. C. Monteleone and M. T. Wright, "Costas array generation and search methodology," in IEEE Transactions on Aerospace and Electronic Systems, vol. **43**, no. 2, pp. 522-538(2007).
4. Weita Chang, "A remark on the definition of Costas arrays," in Proceedings of the IEEE, vol. **75**, no. 4, pp. 522-523(1987).
5. J. C. Guey, "Synchronization Signal Design for OFDM Based On Time-Frequency Hopping Patterns," 2007 IEEE International Conference on Communications, Glasgow, pp. 4329-4334(2007).
6. Jianguo YAO and Qing HUANG, "Applicaton study of Costas arrays in multiuser radar system," in Journal on Communications, vol. **31**, no. 5, pp. 60-72(2010).
7. Jianguo Yao, Yufeng Wang, Wei Heng and Yanling Li, "Constructions of optimal frequency hopping codes based on Welch Costas arrays and their Applications in OFDM system," in Journal of Nanjing University of Posts and Telecommunications(Natural Science), vol. **33**, no. 4, pp. 29-38(2013).
8. Qihang Yao, Yanling Li and Jianguo Yao, "Study on Algorithm of Exhaustive Search for Costas Arrays," 2013 National Radio Application and Management Conference, China, pp. 54-65(2013).
9. J. C. Russo, K. G. Erickson and J. K. Beard, "Costas array search technique that maximizes backtrack and symmetry exploitation," 2010 44th Annual

Conference on Information Sciences and Systems (CISS), Princeton, NJ, pp. 1-8(2010).

10. E. Afacan, "A new search method for costas arrays by using difference triangle analysis," 2017 Progress In Electromagnetics Research Symposium - Spring (PIERS), St. Petersburg, pp. 456-461(2017).
11. K. Drakakis, "A review of Costas arrays," in Journal of Applied Mathematics, pp. 21-32(2006).
12. D. Diaz, F. Richoux, Y. Caniou, P. Codognet and S. Abreu, "Parallel Local Search for the Costas Array Problem," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai , pp. 1793-1802(2012).
13. M. Soltanalian, P. Stoica and J. Li, "Search for Costas arrays via sparse representation," 2014 22nd European Signal Processing Conference (EUSIPCO), Lisbon, pp. 2235-2239(2014).