

# Exhaustive Search for Costas-Type Sequences for Multi-Target Recognition

Oscar Moreno  
University of Puerto Rico at Rio Piedras

Dorothy Bollman  
University of Puerto Rico at Mayaguez

Li Yuchun  
University of Puerto Rico at Rio Piedras

March 7, 2003

## 1 Introduction.

Costas and sonar sequences were respectively introduced in [2] and [10] to deal with the following problem: “An object is moving towards (or away) from an observer, who wants to know effectively the distance to the object and its velocity.” The solution to the problem makes use of the Doppler effect, which states the following: when a signal bounces off a moving target its frequency changes in direct proportion to the velocity of the object relative to the observer. In other words, if the observer sends out a signal towards a moving target, the change between the frequency of the outgoing and that of the returning signal will allow him to determine the velocity of the target, and the time it took to make the round trip will allow him to determine the distance.

In a frequency hopping radar or sonar system, the signal consists of one or more frequencies being chosen from a set  $\{f_1, f_2, \dots, f_m\}$  of available frequencies, for transmission at each of a set  $\{t_1, t_2, \dots, t_n\}$  of consecutive time intervals. For modeling purposes, it is reasonable to consider the situation in which  $m = n$ , and where

$$\{f_1, f_2, \dots, f_n\} = \{t_1, t_2, \dots, t_n\} = \{1, 2, \dots, n\}$$

(we will call this last  $m = n$  case, a Costas type, and the general case sonar type).

Such a Costas signal is conveniently represented by an  $n \times n$  permutation matrix  $A$ , where the  $n$  rows correspond to the  $n$  frequencies, the  $n$  columns correspond to the  $n$  time intervals, and the entry  $a_{ij}$  equals 1 if and only if

frequency  $i$  is transmitted in time interval  $j$ . (Otherwise,  $a_{ij} = 0$ .)

When this signal is reflected from the target and comes back to the observer, it is shifted in both time and frequency, and from the amounts of these shifts, both range and velocity are determined. The observer finds the amounts of these shifts by comparing all shifts (in both time and frequency) of a replica of the transmitted signal with the actual received signal, and finding for which combination of time shift and frequency shift the coincidence is greatest. This may be thought of as counting the number of coincidences between 1's in the matrix  $A = (a_{ij})$  with 1's in a shifted version  $A^*$  of  $A$ , in which all entries have been shifted  $r$  units to the right ( $r$  is negative if there is a shift to the left), and  $s$  units upward ( $s$  is negative if the shift is downward). The number of such coincidences or “hits,”  $C(r, s)$ , is the two-dimensional autocorrelation function between  $A$  and  $A^*$ , and satisfies the following conditions:

$$\begin{aligned} C(0, 0) &= n \\ C(r, s) &= 0 \text{ if } |r| \geq n \text{ or if } |s| > n \\ 0 \leq C(r, s) &< n \text{ except when } r = s = 0 \end{aligned}$$

(This conforms to the assumption that the signal is 0 outside the intervals  $1 \leq f \leq n$  and  $1 \leq t \leq n$ .)

If we have another Costas type signal represented by a matrix  $B = (b_{ij})$ , we can similarly define the two-dimensional cross-correlation function by replacing  $A^*$  by  $B^*$  in the above definition.

In the real world, the returning signal is always noisy. The two-dimensional autocorrelation function,  $C(r, s)$ , is also called the *ambiguity function* in radar and sonar literature and should be thought of as the total “coincidence” between the actual returning noisy signal and the shift of the ideal transmitted signal by  $r$  units in time and  $s$  units in frequency. Among the  $2^{n^2}$  matrices of 0’s and 1’s of order  $n$ , there are  $n!$  permutation matrices, and some of these are not very good as signal patterns for radar and sonar. For example, the  $n \times n$  identity matrix  $I_n$  can be shifted one unit up and one unit left, and will then produce  $n - 1$  coincidences with the original matrix. For large values of  $n$  and a noisy environment, the signal pattern  $I_n$  would most certainly produce spurious targets, shifted an equal number of units in both time and frequency from the real target.

At a minimum, there is a shift of  $A = (a_{ij})$  which will make any of the  $n$  1’s land on any of the  $n - 1$  remaining 1’s, so we know that

$$\begin{aligned} \min C(r, s) &= 1 \\ \max C(r, s) &\geq 1 \\ \text{for all “codes” } (r, s) &\neq (0, 0) \end{aligned}$$

where  $C(r, s)$  is the ideal ambiguity function of the permutation matrix itself. Consequently J.P. Costas defined the ideal  $n \times n$  permutation matrices (which we will call here Costas sequences) as those for which

$$\max_{(r,s) \neq (0,0)} C(r, s) = 1$$

By hand computation, he found examples of such matrices for all  $n \leq 12$ , but was unable to find an example for  $n = 13$ , and was tempted to conclude that these patterns “die out” beyond  $n = 12$ .

In the general sonar case,  $n$  signals are sent out with frequencies ranging from 1 to  $m$ , at times ranging from 1 to  $n$ . Once the whole pattern of signals has returned, the velocity and the distance of the object can be determined as mentioned before. Sonar sequences are those patterns for which exactly one signal is sent out at every time slot, and with the same constraints on the ambiguity function but, in general, with  $n \geq m$ . The goal in constructing sonar sequences is given  $m$  frequencies to construct an  $m \times n$  pattern.

It has been proven in [3] that for  $n > 4$  there are no two different Costas sequences with the same ideal property

in their cross-correlation as that they have in their auto-correlation. Since for the case of multiple targets we need sets of sequences with good auto and cross-correlation properties we must therefore settle for constructing sets of sequences with nearly ideal properties, or in other words cross-correlation 2. In this paper we look at the problem of finding patterns with autocorrelation and cross correlation 1 and 2. Frequency hopping systems are very important for use in wireless communication (see [6], [11]). Therefore our present research has potential applications in this area.

## 2 The algorithms.

We denote the set of Costas-type sequences of size  $n$  with  $i$  hits in their auto-correlation function by  $C_i(n)$ , the set of pairs  $(s, t)$  in  $C_i(n) \times C_i(n)$  with cross-correlation  $j$  by  $C_{i,j}(n)$ , and the largest subset of  $C_i(n)$  such that each pair in  $C_i(n) \times C_i(n)$  has cross-correlation  $j$  by  $C_i^j(n)$ . Our computational tasks involve generating these three sets. In what follows we describe our algorithms for accomplishing this.

### 2.1 Generating $C_i(n)$ .

We use a distributed backtracking algorithm to generate this set. Backtracking is a general procedure for solving a problem by systematically generating all possible solutions. Such a process can be described by a search tree in which each node corresponds to a partial solution. Going down the tree corresponds to progress toward obtaining a complete solution. Going up the tree, that is, “backtracking,” corresponds to returning to a partial solution from which it might be hopeful to proceed forward again.

In the distributed version of backtracking, we use the “manager-worker” technique. The manager dynamically passes each sequence of length  $d$  with no more than  $i$  hits to an idle worker, who in turn continues to search for sequences with  $i$  hits that contain the fixed subsequence of length  $d$ . The worker returns all such sequences with  $i$  hits to the manager and then waits for another subsequence of length  $d$ . The manager continues to compute subsequences of length  $d$  until no more such subsequences exist.

The backtracking used in our algorithm differs from the usual backtracking in two aspects. First, our algorithm

adds terms to both (not just one) sides of a sequence. Ordinary backtracking would not distinguish between unwanted patterns at the beginning and at the end of a sequence. That is, a sequence could be rejected because of an unwanted pattern at the beginning, but ordinary backtracking would generate another sequence with the same pattern, say closer to the end. Adding characters to both sides of the sequence eliminates this redundancy, thus making backtracking faster.

The second source of speedup exploits the fact that  $C_i(n)$  is invariate under the action of the group  $G_4$  of rigid motions of the square generated by reflections about the horizontal and vertical axes. For example, let  $s = (5, 4, 2, 9, 6, 7, 1, 3, 8)$ . Then  $s \in C_i(9)$ . However,  $h(s) = (5, 6, 8, 1, 4, 3, 9, 7, 2)$  (the horizontal reflection of  $s$ ),  $v(s) = (8, 3, 1, 7, 6, 9, 2, 4, 5)$  (the vertical reflection of  $s$ ), and  $h \circ v(s) = (2, 7, 9, 3, 4, 1, 8, 6, 5)$  (the product of horizontal and vertical reflections of  $s$ ) are in  $C_1(9)$ . We have shown [10] that for any  $n$ , there exists a set  $A(n)$  such that  $C_1(n)$  can be expressed as a disjoint union

$$C_1(n) = A(n) \cup h(A(n)) \cup v(A(n)) \cup h \circ v(A(n)).$$

When  $i > 1$  there exist fixed points. For example,  $s = (1, 4, 5, 2, 3, 6) \in C_2(6)$ . In this case  $h(s) = v(s) = (6, 3, 2, 5, 4, 1)$ , which implies that  $s$  is fixed under  $h \circ v$ . In the general case, for any  $i$  and  $n$ , we can determine a set  $A_i(n)$  such that

$$C_i(n) = A_i(n) \cup h(A_i(n)) \cup v(A_i(n)) \cup h \circ v(A_i(n)) \cup F_i(n)$$

where  $F_i(n)$  is the set of fixed points and their horizontal reflections.

Because of the above economies, our backtracking algorithm for generating  $C_i(n)$  runs up to four times faster than the usual backtracking algorithm.

## 2.2 Generating $C_{i,j}(n)$ .

In order to compare all pairs of sequences in  $C_i(n)$ , we need not only more computation nodes, but also more memory in each node. For example, there are 1417736 sequences in  $C_2(10)$  and thus the number of pairs in  $C_2(10)$  to be compared is of order  $10^{12}$ .

For these comparisons, we partition the set of all sequences in  $C_i(n)$  into several smaller sets and load each set into a node. Each node compares pairs of its sequences

and then compares its sequences to those in another node. We use a ring communication structure to carry out these comparisons. For example, for 8 nodes, the comparisons between sequences in distinct nodes can be depicted as in Table 1, where each pair  $(i, j)$  indicates that sequences in node  $i$  are compared to sequences in node  $j$ .

step 1	step 2	step 3	step 4
(0,1)	(0,2)	(0,3)	(0,4)
(1,2)	(1,3)	(1,4)	(1,5)
(2,3)	(2,4)	(2,5)	(2,6)
(3,4)	(3,5)	(3,6)	(3,7)
(4,5)	(4,6)	(4,7)	(4,0)
(5,6)	(5,7)	(5,0)	(5,1)
(6,7)	(6,0)	(6,1)	(6,2)
(7,0)	(7,1)	(7,2)	(7,3)

Table 1: Comparisons of sequences from pairs of nodes.

In the last step, the number of comparisons between nodes need only be half of that in previous steps. This procedure guarantees that after  $\lceil n/2 \rceil$  steps, each pair of sequences will have been compared to each other.

## 2.3 Generating $C_i^j(n)$ .

We can view this problem as the problem of finding a maximum clique in the graph whose nodes are labeled with the members of  $C_i(n)$  and whose edges consist of those pairs in  $C_i(n) \times C_i(n)$  with cross-correlation  $j$ . There are various known algorithms to determine maximal cliques. One especially useful such algorithm that can readily be adopted for our use is the parallel algorithm of Pardalos et al [12]. Unfortunately, the maximal clique problem is NP-hard. It remains to be seen if there is a more efficient method for generating  $C_i^j(n)$ .

## 3 Results.

In this section we discuss some results obtained from our algorithms and some conjectures. We have implemented our algorithms on a variety of platforms, starting with an Intel Paragon and later including a cluster of Sun work stations, an SGI Origin 2000, and a Linux cluster. All three of our algorithms have exponential time complexity, which limits the maximal size problems that we have considered.

First we consider the sequences which have one hit in their cross-correlation function. From [13], we know that every pair of Costas arrays have more than one hit in their cross-correlation function. Therefore we consider only two-hit sequences. In Tables 2 and 3, we list the total number of sequence pairs which satisfy conditions involving cross-correlation 1.

Size	4	5	6	7	8	9	10
Number of Pairs	0	0	4	0	0	0	0

Table 2: One-hit, two-hit pairs with cross-correlation at most one.

Size	4	5	6	7	8	9	10
Number of Pairs	9	12	83	146	113	76	177
Maximum Subset Size	2	2	2	2	2	2	2

Table 3: Two-hit pairs with cross-correlation at most one.

From Table 2, we note that only at the size 6 are there four pairs of sequences which have at one hit in their cross-correlation function. We therefore make

**Conjecture 1** *The cross-correlation of Costas array and a two-hit sequence is two except when size is 6.*

From Table 2 we have following conjecture.

**Conjecture 2** *Suppose we have a set with  $m$  two-hit sequences of length  $n$ , where the cross-correlation is one for any pair of sequences in the set. Then  $m = 2$ .*

Other interesting results obtained from our algorithms are summarized in Tables 4,5, and 6.

Size	No. of Pairs	Max. Subset Size
4	46	4
5	440	8
6	2434	8
7	5084	6
8	14234	6
9	22860	5
10	86528	4
11	155896	5
12	195566	4
13	197238	4
14	129426	4
15	57922	4
16	28688	4

Table 4: One-hit pairs with at most cross-correlation two.

Size	4	5	6	7	8
Number of Pairs	100	1352	20724	162040	5582880

Table 5: One-hit, two-hit pairs with cross-correlation at most two.

Size	4	5	6	7	8
Number of Pairs	39	1179	41244	1369421	47458851

Table 6: Two-hit pairs with cross-correlation at most two.

## Acknowledgement

The research reported here was supported in part by the Office of Naval Research under grant N00014-97-1-0973.

## References

- [1] J. Costas, "A study of a class of detection waveforms having nearly ideal range-dopple ambiguity properties," *Proc. of the IEEE* 72 (1984), 996-1009.
- [2] J. Costas, "Medium Constraints on Sonar Design and Performance" *FASCON Convention Record*, (1975), 68A-68L.
- [3] A. Freedman and N. Levanon, Any two  $N \times N$  Costas signals must have at least one common ambiguity sidelobe if  $N > 3$  - proof. in *Proceedings of the IEEE*, vol. 73, no. 10, (October, 1985), 1530-1532. 1985.
- [4] S. Golomb and H. Taylor, "Two-dimensional synchronization patterns for minimum ambiguity," *IEEE Transactions on Information Theory IT-28* (1982), 600-664.
- [5] S.V. Maric and E.L. Titlebaum, A Class of Frequency Hop Codes with Nearly Ideal Characteristics for Use in Multiple Access Spread Spectrum Communications and Radar and Sonar Systems, in *IEEE Transactions on Communications*, Sept. 1992.
- [6] S. Maric and O. Moreno. "The Role of Frequency Hopping in Today's Digital Communications," accepted for publication in the *Proceedings of the Fifth International Conference on Spread Spectrum Techniques and Applications*.

- [7] O. Moreno and S. Maric, Classes of Costas and Sonar Sequences for Multiple-Target Recognition, in *IEEE ISIT*, 1997.
- [8] O. Moreno, R. Games, and H. Taylor, "Sonar sequences from Costas arrays and the best known sonar sequences with up to 100 symbols," *IEEE Transactions on Information Theory IT-39* (1993), 1985-89.
- [9] O. Moreno, Pei Pei, and J. Ramírez, "A parallel algorithm for the enumeration of Costas sequences," *Proc. of the Seventh SIAM Conf. on Parallel Processing for Scientific Computing* 1995, 255-260.
- [10] O. Moreno, J. Ramirez, D. Bollman, and E. Orozco, "Faster Backtracking Algorithms for the Generation of Symmetry-Invariant Permutations," *Journal of Applied Mathematics* (2002), 277-287.
- [11] O. Moreno and S. Maric, "A New Family of Frequency Hop Codes", *IEEE Transactions on Communications*, August 2000, vol. 48, number 8, pp. 1241-1244.
- [12] P.M. Pardalos, J. Rappe, and M.G.C. Resende, An exact parallel algorithm for the maximum clique problem, High performance algorithms and software in nonlinear optimization, R. De Leone et al (eds.), Kluwer Academic Publishers, pp. 279-300, 1999.
- [13] J. Silverman, V. Vickers, and J. Mooney, "On the number of Costas arrays as a function of array size," *Proceedings of the IEEE 76* (1988), 851-853.