Costas Array Generation and Search Methodology

JAMES K. BEARD, Life Senior Member, IEEE

JON C. RUSSO
Lockheed Martin Advanced Technology Laboratory

KEITH G. ERICKSON, Member, IEEE
Lockheed Martin Maritime Systems and Sensors

MICHAEL C. MONTELEONE, Member, IEEE
Department of Defense

MICHAEL T. WRIGHT, Member, IEEE
Picatinny Arsenal

Costas arrays are permutation matrices that provide sequencing schemes for frequency hop in FSK waveforms. Such frequency-shift keying (FSK) waveforms can be designed to have nearly ideal ambiguity function properties in both the time and frequency directions: the Costas property permits at most one coincident tone in autocorrelations in both time and frequency. Costas arrays are found by number-theoretic generators and their extensions, and by exhaustive search methods. Two new extensions of number-theoretic methods are introduced here that find two new Costas arrays. All Costas arrays for orders 24, 25, and 26 are disclosed here, including previously unknown examples.

Manuscript received December 21, 2004; revised October 25, 2005; released for publication July 12, 2006.

IEEE Log No. T-AES/43/2/903008.

Refereeing of this contribution was handled by P. K. Willett.

Authors' addresses: J. K. Beard, 122 Himmelein Rd., Medford, NJ 08055-9300, E-mail: (jkbeard@ieee.org); J. C. Russo, Lockheed Martin Advanced Technology Laboratory; K. G. Erickson, Lockheed Martin Maritime Systems and Sensors; M. Monteleone, Dept. of Defense, Defense Communications and Army Transmission Systems; M. Wright, Pictatinny Arsenal.

I. INTRODUCTION

When radar, sonar, or communications systems have requirements for unambiguous time and frequency offset, often the simplest waveforms that will meet these requirements are frequency-jump burst (FJB) types and their derivatives. This arises with sonar that must detect high-speed targets, high-resolution radar for objects at orbital speeds, and many communication systems, and other applications. Comprehensive selection of Costas arrays is important when waveform agility, cross-correlations, or intersymbol interference are a priority.

A. Costas Property and Ambiguity Function

The time versus frequency mapping of an FJB is a permutation. Each of N subwaveforms in an FJB waveform occupies a time slot and a frequency channel. When the time of arrival and frequency are offset by an integral number of time slots and frequency channels, and no more than one of the subwaveforms overlap, this waveform meets the Costas condition. The ideal ambiguity function for an FJB waveform can be achieved when the Costas condition is met. This ideal is a single, central peak and uniform sidelobes in both time and frequency with a uniform maximum height that is inversely proportional to the number of subwaveforms N.

An ambiguity function $\chi(\tau,\phi)$ is a function of the signal $u(t)$, and is defined by [1]

$$\begin{aligned}\chi(\tau,\phi) &= \int u(t)\cdot u^*(t)\cdot \exp(-j\cdot 2\pi\cdot\phi\cdot t)\cdot dt \\ &= \int U^*(f)\cdot U(f+\phi)\cdot \exp(-j\cdot 2\pi\cdot f\cdot\tau)\cdot df\end{aligned} \tag{1}$$

where $U(f)$ is the Fourier transform of the signal. The range-Doppler response of a coherent receiver to a waveform approximates the ambiguity function. When the medium prevents coherent processing across the full FJB frequency range as in most sonar applications, each frequency channel can be processed separately and the frequency channels combined incoherently. The resulting delay-Doppler response has a range resolution determined by the bandwidth of the subwaveform used in each frequency channel. The definitive analysis of this type of processing is given in [2].

When the FJB waveform is processed coherently across the FJB frequency extent, the ambiguity function gives a range resolution determined by the full FJB bandwidth as opposed to that of the subwaveforms; that is, about a factor of N smaller than that achieved with incoherent combination of the frequency channels. This type of processing is used whenever coherency over the entire waveform is available.

1) *Radar, Sonar, Communications and Other Applications*: With emerging technologies for exhaustive search as presented here, sufficient palettes of moderate order Costas arrays are available to support features such as waveform agility. Number-theoretic generators and extensions provide less comprehensive selections but large numbers of Costas arrays of arbitrarily large order. Waveforms incorporating Costas arrays as part of their design are particularly useful in applications where the target Doppler is a significant portion of the radar bandwidth, such as radars designed to track extra-atmospheric objects, particularly in highly eccentric orbits such as those with Molnyia orbits or ballistic objects such as launch vehicles. Other radar types that can use frequency-shift keying (FSK) waveforms based on Costas arrays include CW or quasi-CW bistatic radars and high duty cycle radars with broadband chips that are transmitted in an FSK pattern based on Costas arrays.

Software defined radio (SDR) is an enabling technology for evolving digital wireless communications infrastructure such as cell phones [3]; here, Costas arrays are used to achieve and maintain simultaneous synchronization in time and frequency. An apparently unrelated application is digital watermarking. An additive Costas array at the least significant bit (LSB) level can provide the hook for synchronization and detection of the codes, but more often the use of a maximal length or m-sequence, which is simply the sequence of coefficients of x^{k-1} in $GF(2^k)$ in sequential powers of x, is used in a one-dimensional scheme, which is extended to two dimensions by stacking codes [4, 5]. All of these applications accrue because of the fundamental property that J. P. Costas needed for sonar signals [1, 6, 7]: an ambiguity function with ideal properties. Studies have been made of cross-correlation of FSK waveforms based on Costas arrays with results similar to those of binary phase-shift keying (BPSK) radar and communications waveforms [8]. Also, recent results reported in [9] show that Costas arrays as a component of waveform concepts can lead to very high performance designs.

2) *Radar Waveform Example*: An example for an order 24 Costas array is shown in Fig. 1. The ideal bed-of-nails sidelobe structure is apparent in this three-dimensional plot, but the 28.3 dB sidelobe performance is clearer in the zero Doppler slice shown in Fig. 2.

B. Finding Costas Arrays

Costas arrays are found using exhaustive search and number-theoretic generators. Here we present the results of exhaustive searches for Costas arrays

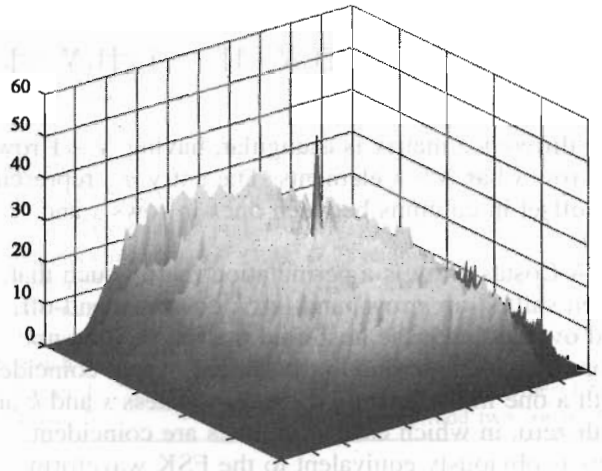

Fig. 1. Ambiguity function for FSK waveform based on order 26 Costas array.

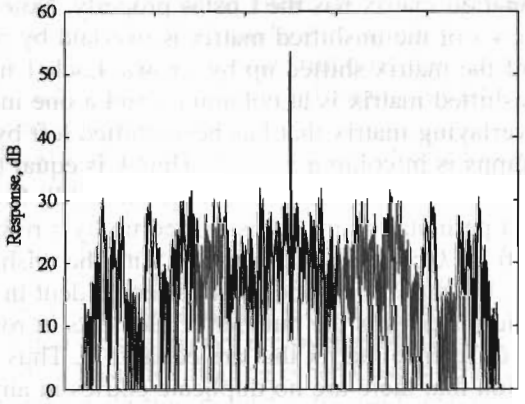

Fig. 2. Zero-Doppler slice of ambiguity function.

of order 24, 25, and 26, and the methodology and implementation of our search method.

We also introduce two new generators of Costas arrays that use a singularity in a modified Lempel or Golomb number-theoretic generator to add a dot and a modification of this generator. These generators may or may not produce a Costas array but have produced two new Costas arrays for orders too large for exhaustive search at the present time, and they found several others that are known only through selective search methods.

C. Basic Properties

1) *Notation and Definitions*: Here we lay the foundations for our presentation of search methodology, existing and new generators, and rate of occurrence of Costas arrays as a function of order.

Column-index notation expresses a permutation matrix as an ordered set of N integers, one for each row of the matrix, each representing the index of the column with the one, as $(c_1, c_2, c_3, \ldots c_N)$.

A difference matrix is defined from the column-index notation of a permutation matrix. Each element

$d_{s,i}$ in row s and column i is

$$d_{s,i} = c_{s+i} - c_i, \qquad s \in [1, N-1], \qquad i \in [1, N-s]. \tag{2}$$

The difference matrix is triangular, having $N-1$ rows, and row s has $N-s$ elements. The entry $d_{s,i}$ represents the offset in columns between ones in rows s and $s+i$.

A Costas array is a permutation matrix such that, when shifted up s rows and left k columns, end-off, and overlaid with the unshifted matrix, there is no more than a single one in the shifted matrix coincident with a one in the unshifted matrix—unless s and k are both zero, in which case all N ones are coincident. This is obviously equivalent to the FSK waveform ambiguity function requirement that we call the Costas condition.

The difference matrix is used to determine if a permutation matrix has the Costas property. Note that row i of the unshifted matrix is overlaid by row $i+s$ of the matrix shifted up by s rows. Each 1 in the unshifted matrix is at column c_i, and a one in the overlaying matrix that has been shifted left by k columns is in column $c_{i+s} - k$. Thus k is equal to the difference matrix entry $d_{s,i}$ as given in (2). So, when a permutation matrix is shifted up by s rows and left by k columns and overlaid with the unshifted matrix, the number of ones that are coincident in the overlaid matrices is the number of elements in row s of the difference matrix that are equal to k. Thus the condition that there are no duplicate entries in any row is equivalent to the Costas condition.

2) *Costas Arrays Come in Sets of Four or Eight*: An important property of Costas arrays is that they come in sets of four or eight. This is apparent when one considers that a set of four or eight Costas arrays can be constructed from a single Costas array.

1) Reversing the order of the rows of a Costas array produces another Costas array.

2) Reversing the order of the columns of a Costas array produces another Costas array.

3) Reversing the orders of both the rows and columns of a Costas array produces another Costas array.

4) Transposing a Costas array provides a basis for another four Costas arrays.

We call any Costas array that is in the same group of four or eight a polymorph of any other Costas array in that group. Any Costas array in a group of only four is called a symmetrical Costas array; two of the set will be symmetrical about the main diagonal (the "attacking bishops" case), and the other two will be symmetrical about the antidiagonal (the "attacking queens" case). If a Costas array is symmetrical about either the main diagonal or antidiagonal, then the transpose will be a duplicate of one of the other cases, and this Costas array will be part of a set of four, not eight. Note that rotating a Costas array is equivalent to transposing it and then reversing the order of either the rows or columns, so rotating a Costas array will also produce a Costas array in the same set of four or eight.

TABLE I
Costas Array and Difference Matrix Example

5	4	6	2	3	1
−1	2	−4	1	−2	
1	−2	−3	−1		
−3	−1	−5			
−2	−3				
−4					

3) *Example*: An example of a Costas array of order 6 is (5,4,6,2,3,1). This Costas array and its difference matrix are shown in Table I.

The matrix expressed using column-index notation as the sequence of positive integers from one to N is the identity matrix, as $(1,2,3,\ldots N)$. We use simultaneous definitions of a Costas array and the difference matrix in fast methods for exhaustive searches; the methods are variations of backtrack programming with preclusion [10].

II. EXISTING NUMBER-THEORETIC GENERATORS

The Welch and Lempel-Golomb generators are given in [11] and were presented with proofs in [12]. Extensions are given with existence proofs in [13] and [14]. They are summarized here to establish notation and to provide a basis for closed form expressions for the difference table entries for these generators.

All the number-theoretic generators are based on Galois finite fields [11]. Galois fields exist for orders q that are powers of a prime p^k; when $k > 1$ the field is a vector extension [15–17].

The usefulness of Galois fields here lies in primitive elements, sometimes called primitive roots. The successive powers of each primitive element represent a permutation of the elements, exclusive of zero; for this reason primitive elements are also sometimes called generators. For the Galois field of order q, $GF(q)$, the number of primitive elements is given by the Euler phi or totient function $\phi(q-1)$.

A. Welch Generator

In the mid 1970s, John P. Costas wrote a letter to Solomon W. Golomb concerning Costas arrays, and Lloyd Welch pointed out that, when q is prime, Costas arrays of order $q-1$ are generated by

$$c_i + 1 = \alpha^{i+r} \bmod q \tag{3}$$

where the offset r is any integer and α is a primitive root in $GF(q)$ [18]. We emphasize that because the column index is on the same level as an element of

the finite field, q must be a prime and the field cannot be a vector extension (i.e., the Welch generator applies for Costas array orders one less than a prime).

An expression for elements in the difference matrix is found using (3),

$$\begin{aligned} d_{s,i} &= c_{i+s} - c_i \\ &= \alpha^{i+s+r} - \alpha^{i+r} \bmod q \\ &= \alpha^{i+r} \cdot (\alpha^s - 1) \bmod q. \end{aligned} \tag{4}$$

B. Lempel-Golomb Generator

Costas arrays of order $q-2$ are generated using two (not necessarily distinct) primitive roots α and β in $GF(q)$ by

$$\alpha^{i+1} + \beta^{c_i+1} = 1 \text{ in } GF(q). \tag{5}$$

When α and β are not distinct, we have the Lempel generator, which generates symmetrical Costas arrays that have polymorphs in groups of four. When α and β are distinct, the resulting Costas arrays are not symmetrical and have polymorphs in groups of eight.

We obtain the elements of the difference matrix by writing (5) for row i, for row $i+s$, and solving both equations for the powers of β,

$$\left.\begin{aligned} \beta^{c_i+1} &= 1 - \alpha^{i+1} \\ \beta^{c_{i+s}+1} &= 1 - \alpha^{i+s+1} \end{aligned}\right\} \text{ in } GF(q) \tag{6}$$

and divide the two equations to obtain β to the power of the entry in the difference table,

$$\left.\begin{aligned} \beta^{d_{s,i}} = \beta^{c_{i+s}-c_i} &= (1-\alpha^{i+s+1}) \cdot (1-\alpha^{i+1})^{-1} \\ &= (1-\alpha^{i+1} + \alpha^{i+1} - \alpha^{i+s+1}) \cdot (1-\alpha^{i+1})^{-1} \\ &= 1 - (1-\alpha^s) \cdot (1-\alpha^{-i-1})^{-1}. \end{aligned}\right\} \text{ in } GF(q) \tag{7}$$

C. Properties of Generators

The use of powers of elements of Galois fields in the Welch and Lempel-Golomb generators means that the indices in the exponents are effective modulo $q-1$. This is true of the row indices of the Welch generator as given by (3) and both the row and column indices of the Lempel-Golomb generator as given by (5).

Equation (3), the Welch generator, produces arrays that are doubly periodic; the row indices have period $q-1$, and the column indices have period q. Equation (3) fails when the left hand side is zero, but still generates a Costas array that is singly periodic. Equation (3) is specifically written with the column index as a dependent variable, and an arbitrary row index offset r, to address this transparently. All known singly periodic Costas arrays are produced by a Welch generator [18].

TABLE II
Number-Theoretic Generators and Extensions From [11]

Generator	Remarks
Welch 1	$c_i + 1 = \alpha^{i+r} \bmod q = p$, base method
Welch 2	(1,1) removed
Welch 3	(1,1) and (2,2) removed
Lempel-Golomb 2	$\alpha^{i+1} + \beta^{c_i+1} = 1$, base method
Lempel-Golomb 3	L-G 2, (1,1) removed (see [13])
Lempel-Golomb 4	(1,1), (2,2) removed; $q = 2^k$
Golomb* 4	(1,1), $(2,q-2)$ removed; $\alpha + \beta = 1$
Golomb* 5	Golomb* 4, $(q-2,2)$ removed
Taylor 4	Lempel 2, (1,2) and (2,1) removed
Welch 0	Add a corner dot to Welch 1
Taylor 1	Lempel-Golomb 2, add a corner dot
Taylor 0	Lempel-Golomb 2, add two corner dots

The Lempel-Golomb generator produces arrays that are doubly periodic, with both row and column indices having period $q-1$. The Lempel-Golomb generator (5) fails when either the row or column index is equal to $q-2 \bmod q-1$. Although (5) produces a doubly periodic structure, the forbidden rows and columns partition off identical Costas arrays of order $q-2$.

Most importantly, note that our closed forms for values in the difference matrix given by (4) and (7) show that the elements of the rows of the difference equations for Costas arrays that are found by the generators are unique modulo $q-1$—which is a stronger condition than the Costas condition. Thus generated Costas arrays are overconstrained and provide a basis for finding other Costas arrays by providing a starting point for focused searches (i.e., nondeterministic try-and-check generators, as are some of the Taylor extensions).

D. Generators and Extensions in [11]

Reference [11] defines methods and extensions from Galois fields in terms of the difference between the order of the Galois field and the order of the Costas array. For example, the Welch generator is referred to as Welch 1. The integer following the name is the difference between the order of the Costas arrays generated and the order of the finite field q; for example, the Taylor 4 method results in Costas arrays of order $q-4$. They are summarized in Table II. The order of the Galois field is referred to as a prime p or a power of a prime q, and the order of the Costas arrays is N.

Proofs of the methods given in the table are provided in [12], [13], and [14].

The advantages of the number-theoretic generators include:

1) Simplicity and speed—the generators are simple and fast, given arithmetic in $GF(q)$.

2) Numbers of Costas arrays generated—the Welch generator guarantees $N = q-1$ Costas arrays for each primitive element that exists.

TABLE III
Primitive Elements in $GF(27)$ for Generator Polynomial (1 2 0 1)

Primitive Element Index	Polynomial
1	(0 1 0)
2	(2 1 0)
3	(2 1 2)
4	(2 2 1)
5	(1 1 0)
6	(2 1 1)
7	(0 0 2)
8	(0 1 2)
9	(2 2 2)
10	(1 0 1)
11	(0 2 2)
12	(1 0 2)

3) Availability of Costas arrays of large order—the generators are implemented with good computational speed for larger orders.

4) The extensions give Costas arrays of order other than one less than a prime or two or three less than a power of a prime.

The disadvantages of the number-theoretic generators include:

1) Not all orders are available—the orders of the Costas arrays generated are strictly limited by values of q for which $GF(q)$ exists, and orders up to five less than q and because some extensions don't work, only orders $q-2$, $q-3$ and $p-1$ are guaranteed, where p is any prime and q is a prime or power of a prime [13, 15].

2) Many of the different combinations of primitive elements in $GF(q)$ produce duplicates or polymorphs of Costas arrays found by other combinations—the number of Costas arrays produced by the generators is thus far less than that which might be expected from the number of primitive roots available for the generators.

Note that Moreno showed that for any $GF(q)$ there always exists a pair of primitive roots α and β such that $\alpha+\beta=1$. Thus for every $GF(q)$, a Costas array is generated that has a 1 at (0,0). This 1 can be removed to form another Costas array of order $q-3$, therefore, whenever a Lempel-Golomb 2 generator exists, a Lempel-Golomb 3 generator also exists [13]. Note in the list of primitive elements for $GF(27)$, given in Table III, that this holds for the pairs {3,4}, {6,9}, and {7,10}.

Note that [11] and [19] report finding no Costas arrays of order 53. The authors have used the methods of [11] and generalizations to find one for some time, (0,41,47,50,25,39,46,23,38,19,36,18,9,31,42,21,37, 45,49,51,52,26,13,33,43,48,24,12,6,3,28,14,7,30, 15,34,17,35,44,22,11,32,16,8,4,2,1,27,40,20,10, 5,29). It is found using the Welch 0 extension using $GF(53)$ with primitive root 2 or 27.

III. TWO NEW GENERATORS OF COSTAS ARRAYS

A. First New Generator Extension

An obvious variation on the Lempel-Golomb generator is

$$\alpha^{i+1}+\beta^{c_i+1}=\gamma \tag{8}$$

where γ is an element in $GF(q)$. Note that making the inhomogeneity different from 1 moves the forbidden row and column. The last line in (7) becomes

$$\beta^{d_{s,i}}=1-(1-\alpha^s)\cdot(1-\gamma\cdot\alpha^{-i-1})^{-1}. \tag{9}$$

We can make the substitution, without loss of generality,

$$\gamma=\alpha^{ga}. \tag{10}$$

Equation (9) is now

$$\beta^{d_{s,i}}=1-(1-\alpha^s)\cdot(1-\alpha^{ga-i-1})^{-1}. \tag{11}$$

The product in (11) can be seen to provide distinct non-zero values for $d_{s,i}$ as i cycles from 0 to $q-3$ for just two values of ga, 0, and $q-1$, both of which give unity as the value of γ. The difficulty lies when

$$\alpha^{i+1}=\gamma=\alpha^{ga} \tag{12}$$

and (8) fails, because the second term on the right hand side of (11) does not exist. We call this a singular case.

We introduce the concept of making the row and column offsets of 1 in (8) arbitrary and allowing an exponent of zero, and define the corresponding column index as the only one that is not represented in the set found for i from 0 to $q-2$ inclusive. This adds a dot at the intersection of the forbidden row and column. For the Lempel-Golomb generator, where γ is 1, this extension is one form of a Taylor extension, adding a corner dot to a Costas array found with the Lempel-Golomb generator [11]. We define a more general extension as follows. We have

$$\alpha^{i+i\text{off}}+\beta^{c_i+c\text{off}}=\gamma=\alpha^{ga}=\beta^{gb} \tag{13}$$

and, dividing through by γ,

$$\beta^{c_i+c\text{off}-gb}+\alpha^{i+i\text{off}-ga}=1. \tag{14}$$

Here, we define the singular cases as

$$c_i\begin{cases}=gb-c\text{off}, & i=ga-i\text{off}\bmod q-1\\ =(\text{from equation}), & i\neq ga-i\text{off}\bmod q-1.\end{cases} \tag{15}$$

Looking at the elements of the difference matrix, we have

$$\beta^{d_{s,i}}=1-(1-\alpha^s)\cdot(1-\alpha^{ga-i-i\text{off}})^{-1}$$
$$\text{when}\quad i\neq ga-i\text{off}\bmod q-1 \tag{16}$$
$$d_{s,ga}=c_{s+ga}+c\text{off}-gb$$

which, when examined for uniqueness in a row of the difference matrix is, in general, inconclusive. Here we have a general extension to the Lempel-Golomb method that always produces a permutation matrix of order $q-1$ but not always a Costas array. Thus, permutation matrices generated using this extension must be tested for the Costas condition.

B. Second New Generator Extension

Another new generator extension is found through a modification of (13),

$$\alpha^{i+i\text{off}} - \beta^{c_i+c\text{off}} = \gamma = \alpha^{ga} = -\beta^{gbn}. \tag{17}$$

Dividing through by γ, as we did before, gives us

$$\alpha^{i+i\text{off}-ga} + \beta^{c_i+c\text{off}-gbn} = 1. \tag{18}$$

The singular cases are defined as

$$c_i \begin{cases} = gbn - c\text{off}, & i = ga - i\text{off} \bmod q - 1 \\ = (\text{from equation}), & i \neq ga - i\text{off} \bmod q - 1. \end{cases} \tag{19}$$

These equations are identical to (13)–(15) through except for the minus sign in (17) and use of gbn in place of gb. These variables map through

$$\beta^{gb} = -\beta^{gbn}. \tag{20}$$

The analogy carries to the explicit form for the elements of the difference matrix. Equation (16) also applies for this generator extension with gbn in place of gb:

$$\begin{aligned} \beta^{d_{s,i}} &= 1 - (1-\alpha^s)\cdot(1-\alpha^{ga-i-i\text{off}})^{-1} \\ i &\neq ga \bmod q - 1 \\ d_{s,ga} &= c_{s+ga} + c\text{off} - gbn \end{aligned} \tag{21}$$

which again, when examined for uniqueness in a row of the difference matrix, is inconclusive. The Costas arrays from this extension will have their columns permuted from the first extension, and the value of zero is allowed for ga, which was taken as the normal Lempel-Golomb generator for the previous form. Again, permutation matrices generated using this extension must be tested for the Costas condition.

C. Examples of New Extensions

We give examples here for several orders, beginning with order 4 and progressing to larger orders. Specific emphasis is on providing simple examples to provide insight, nontrivial examples using off-diagonal singular case positions, and larger orders providing new results. Order 26 is deferred to an in-depth treatment of all cases in subsection D below.

We work here with the column-index formulations. In the column-index formulation, columns are shifted end-around in a permutation matrix by adding the number of columns to each index, modulo the order of the permutation. Rows are shifted end-around by shifting the elements of the representation end-around.

1) *Order 4*: We begin with a simple example of order 4. We show how Costas arrays of order 3, generated by Lempel and Golomb generators over $GF(5)$, produce Costas arrays of order 4 through the extension introduced here.

The primitive roots in $GF(5)$ are 2 and 3. The Costas array (1,0,2) is generated using the Lempel generator for $\alpha = 2$. Using (14) we see that the permutation matrix (0,2,1,3), with its rows and columns shifted together, keeping element (0,0) on the main diagonal, produces the candidates for Costas arrays.

The Costas arrays (0,1,3,2) and (1,0,2,3) are produced by shifts of one and three, respectively. The shifts of zero and two did not produce valid Costas arrays.

The Lempel generator for $\alpha = 3$ produces (0,2,1). Again using (14), we see that the permutation matrix (0,1,3,2), with rows and columns shifted to move the element (0,0) down the main diagonal as before, provides four candidates for the Costas property. The result is the Costas arrays (0,1,3,2) and (1,0,2,3) for shifts of zero and two, while the permutations produced by shifts of one and three are not Costas arrays.

The Golomb generator for $\alpha = 2$ and $\beta = 3$ produces the Costas array (1,2,0). The singularity for powers of α of zero, one, two, and three is at positions (0,0), (1,3), (2,2), and (3,1). All of them produce valid Costas arrays with the rows and columns of (0,2,3,1) shifted to move the element at (0,0) to these positions, producing Costas arrays (0,2,3,1), (0,3,1,2), (1,3,2,0), and (3,0,2,1), respectively.

2) *Order 10*: For order 10, the Golomb generator over $GF(11)$ for $\alpha = 2$ and $\beta = 7$ produces the order 9 Costas array (4,8,5,0,2,3,1,7,6). Valid Costas arrays are produced for shifts of (0,0) and (1,7). They are (1,5,9,6,1,3,4,2,8,7) and (4,7,2,6,3,8,0,1,9,5), respectively. The permutations produced by the other shifts are not Costas arrays.

D. New Costas Arrays of Order 26 and Generators

The authors have implemented all the generators in Table II and the new generators reported here and in [19]. In addition, generalizations not found in the literature are incorporated in the authors' methods, including rotating Costas arrays end-around in both rows and columns and adding or dropping one or more corner dots; we refer to this generalization of the Taylor extensions as the spin generalizations. The dots in Fig. 5 show the total count of unique Costas arrays from this methodology, with our update of the

estimate from [20] as a solid curve (see discussion of [20] in Section IVA). Results for order 26 are given in Table XI and Table XII. The table entries are denoted in discussions to follow as CA25.0–CA25.11 and CA26.0–CA26.7, respectively, denoting the rows in the tables in order.

The entry CA26.0 is found by adding a corner dot to a Costas array that is a polymorph of CA25.0, and CA26.3 can be found by adding a corner to a version of CA25.0 that has been end-around rotated along the main diagonal. This Costas array is found from the generators. CA26.1, CA26.2, CA26.4, and CA16.5 are obtained from deleting a corner dot from one of 196 unique Costas arrays of order 27 produced by the generators.

The notation for polynomials below is of the form $(c0,c1,c2,c3)$ denoting the coefficients of x^0, x^1, x^2, and x^3, respectively. The coefficient $c3$ appears only in the generator polynomial. The primitive elements are numbered in a table and primitive element pairs, each element defined by an index, are denoted by $\{e1,e2\}$.

The Welch 3 generator produces CA26.1 and the rest of its set of eight from $GF(29)$ with primitive roots 2, 9, and 27. Without the spin generalization, only primitive root 2 finds the set.

The Taylor 1 generator produces CA26.0 as mentioned above with $GF(27)$ using the generator polynomial (1 2 0 1). The primitive elements are given below. All of the set of eight polymorphs of CA26.0 are produced by these primitive element pairs: {3,4}, {3,6}, {3,7}, {3,9}, {4,6}, {4,7}, {4,10}, {6,9}, {6,10}, {7,9}, {7,10}, and {9,10}. The spin generalization does not affect the results for Taylor 1.

For order 26, the Lempel-Golomb generators over $GF(27)$ produce several examples, two of which we illustrate. The first few column indices are given in the illustrations here; the full Costas arrays or polymorphs of them are given in Table XII.

The Lempel generator provides the order 25 Costas array, (23,15,19,22...) which is also given in [11]. For a shift of one, the new extension provides a symmetrical Costas array of order 26, (11,1,25,17,21,24...). Other shifts, including zero, do not provide valid Costas arrays. This Costas array was found in [21] through a specialized search for symmetrical Costas arrays.

The Lempel generator for other primitive roots provides the order 25 Costas array (15,24,21,19...). This is a polymorph of the previous example. A shift of 24 provides the order 26 Costas array (23,20,18,5,21...), a polymorph of the Costas array provided by the previous example.

The Golomb generator provides the Costas array of order 25 (16,14,24,6...). A shift of zero provides the Costas array of order 26 (1,17,15,25,7...).

The Golomb generator provides the Costas array of order 25 (24,18,22,20...) and a shift of zero provides the Costas array of order 26 (1,25,19,23,21...). This is a polymorph of the previous example.

TABLE IV
Lempel-Golomb 3 Costas Arrays

Primitive Elements in $GF(29)$	Costas Array of Order 26
(2,2)	CA26.4
(2, 15)	CA26.4
(3, 14)	CA26.2
(3, 27)	CA26.2
(8, 19)	CA26.5
(8, 26)	CA26.5
(10, 14)	CA26.2
(10, 27)	CA26.2
(11, 19)	CA26.5
(11, 26)	CA26.5

TABLE V
New Singularity 1 Costas Arrays

Primitive Element Indices in $GF(27)$	Costas Array of Order 26
{1, 1}	CA26.3
{2, 2}	CA26.3
{4, 6}	CA26.0
{4, 10}	CA26.0
{5, 5}	CA26.3
{6, 10}	CA26.0
{8, 8}	CA26.3
{11, 11}	CA26.3
{12, 12}	CA26.3

TABLE VI
New Subtractive Singularity 1 Costas Arrays

Primitive Element Indices in $GF(27)$	Costas Array of Order 26
{4, 6}	CA26.0
{4, 10}	CA26.0
{6, 10}	CA26.0

The Lempel-Golomb 3 generator produces CA26.2 and CA26.5, and, with the spin generalizations, CA26.4. The base generator of order 27 Costas arrays uses $GF(29)$. The primitive root pairs and the Costas array of order 27 that results are given in Table IV.

The new generator presented here is based on adding a dot at the singularity of the Lempel-Golomb 2 works with $GF(29)$ and produces CA26.0 and CA26.3 as shown in Table V. In this case, the spin generalization does not produce additional Costas arrays.

The new generator presented here based on subtracting powers of primitive elements and adding a dot at the singularity produces CA26.0 using $GF(27)$. Primitive element pairs in $GF(27)$ and the Costas arrays generated are presented in Table VI. The spin generalizations produce no additional Costas arrays here.

The Ricard method, using $GF(27)$ and the Lempel-Golumb 2 as the base generator, produces

TABLE VII
Ricard LG 3 Costas Arrays

Primitive Element Indices in $GF(27)$	Costas Array of Order 26
{1, 1}	CA26.3
{1, 12}	CA26.3
{2, 2}	CA26.3
{2, 11}	CA26.3
{3, 4}	CA26.0
{3, 6}	CA26.0
{3, 7}	CA26.0
{3, 9}	CA26.0
{4, 6}	CA26.0
{4, 7}	CA26.0
{4, 10}	CA26.0
{5, 5}	CA26.3
{5, 8}	CA26.3
{6, 9}	CA26.0
{6, 10}	CA26.0
{7, 9}	CA26.0
{7, 10}	CA26.0
{8, 8}	CA26.3
{9, 10}	CA26.0
{11, 11}	CA26.3
{12, 12}	CA26.3

TABLE VIII
Order 26 Costas Arrays Produced by Generators

Method	26.0	26.1	26.2	26.3	26.4	26.5
Welch 3		Y				
Taylor 1	Y					
LG 3			Y		S	Y
NI1	Y			Y		
NSI1	Y					
RLG1	Y			S		

CA26.0 and, with the spin generalizations, CA26.3. Primitive element pairs in $GF(27)$ and the Costas arrays generated are presented in Table VII.

In summary, the generators produce the rows of Table XII, the comprehensive list of essential Costas arrays of order 26, according to the summary in Table VIII. Those rows produced only when the spin generalization is enabled are denoted by "S" in the table.

The Costas arrays CA26.3 and CA26.4 are symmetrical. The Costas array CA26.4 is found only when the spin generalizations are enabled. However, backtrack programming with preclusion has apparently been implemented for symmetrical Costas arrays and completed up to order 32 [22] so we conclude that CA26.3 and CA26.4 are known as results of that work.

E. Summary

The new method presented here uses a modification of the inhomogeneity of the Lempel-Golomb generator, and the Taylor 1 extension presented in [11] can be considered a special case. It applies whenever $GF(N + 1)$ exists. Unlike previous extensions, it adds dots to the interior of existing matrices produced by the number-theoretic generators.

Reference [19] recently presented a method for adding dots to the forbidden gap that occurs when Costas arrays produced by the Lempel-Golomb generator are viewed as producing doubly periodic quilts of identical Costas arrays separated by blank rows and columns. Four Costas arrays are presented as undiscovered, two of order 29 and one each of order 36 and 42. The one of order 36 presented in [23] can be found using methods given in [11] and generalizations by the authors, but the other three Costas arrays reported in [19] were previously unknown to the authors.

The methods presented here produce Costas arrays for various orders as shown in Table IX.

TABLE IX
Costas Arrays Found by New Methods

	Asymmetrical		Symmetrical	
Order	Sum	Diff	Sum	Diff
3	0	0	1	1
4	1	1	1	1
6	0	1	2	2
7	8	8	4	4
8	4	5	1	0
10	6	9	5	2
12	4	9	3	2
15	1	1	0	0
16	6	6	3	1
18	1	2	1	0
22	5	4	4	0
26	1	1	1	0
28	1	1	0	0
30	0	0	1	1
36	1	1	1	1
40	0	0	1	1
42	0	0	2	1
46	1	1	0	0
52	0	0	1	1

Of the Costas arrays enumerated in the table, two are new, one of order 36, (1,28,32,18,20,26,31,8,0, 29,16,35,15,22,13,11,23,4,30,5,25,14,17,27,21,6, 24,2,10,19,7,3,34,33,12,9), and one of order 42, (0,38,9,3,26,32,37,35,11,2,36,8,20,22,33,19,41, 23,31,15,12,39,13,17,34,27,4,25,40,30,29,18,5, 14,24,7,10,6,1,21,28,16). The new methods find two of order 36, but one of them has been found by the methods of [11] and extensions and is also reported in [19].

The second method introduced here finds three of order 8 that are not found by other generators but does not generate Costas arrays that are unknown or not generated by the first method.

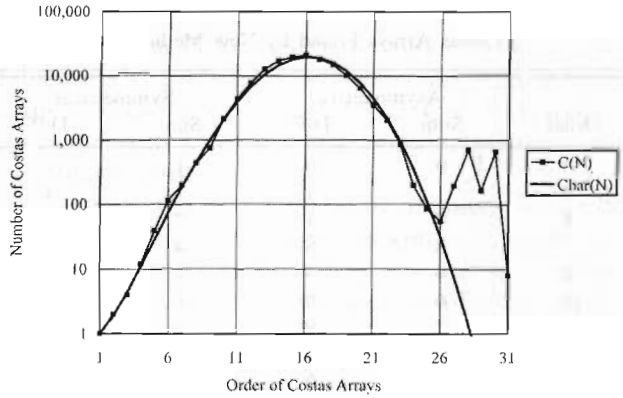

Fig. 3. Numbers of Costas arrays versus order.

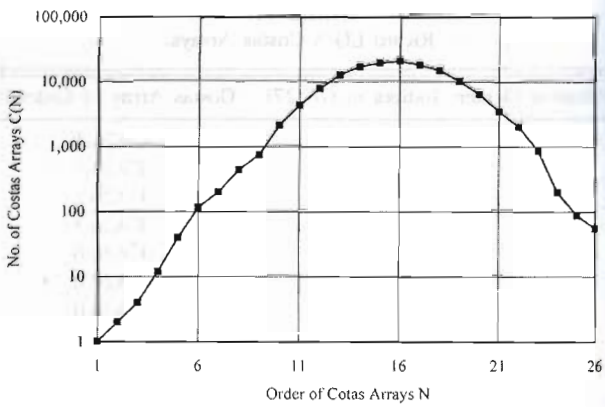

Fig. 4. Number of Costas arrays for order to 26.

IV. FINDING COSTAS ARRAYS THROUGH EXHAUSTIVE SEARCH

A. Numbers of Costas Arrays as Function of Order

A remarkable paper [20] appeared in 1988. Based on a probabilistic analysis of the backtrack programming algorithms that use the difference matrix, they arrived at an equation for the approximate number of Costas arrays versus order,

$$\text{Cbar}(N) = (N!) \cdot \left(1 - \frac{K}{N+1}\right)^{(N+1)\cdot(N-1)\cdot(2N-3)/24}. \tag{22}$$

In (22), the parameter K is a free parameter, and the authors used it to fit to the known numbers of Costas arrays at that time, up to order 17. The value of K that gave the best fit for them was 1.111; we have extended that here to an rms fit to orders to 26, arriving at a value of 1.107814 for K. The result is shown as the bold solid curve in Fig. 3.

The rationales given in the references are very convincing, and the fit to real data is excellent over a wide range of orders. The numbers of Costas arrays generated for orders 27, 28, 29, and 30 are included in the plot. The implications of these results include the following.

1) The backtrack programming methods spend most of their time near an intermediate recursion level, and software monitoring verifies this.

2) The number of Costas arrays not "forced to exist" by number-theoretic considerations continues to decline, dropping below one past order 28.

Work reported on in [20] treated the results of comprehensive searches up to order 17, and did not remark on the generators of [11] in the main body of their paper. However, in their conclusions they refer to [11] and note that algebraic constructions presented there produce large numbers of Costas arrays for larger orders. They also note there that as the order increases, the constructed Costas arrays will dominate those predicted by the probabilistic approach, and noting that the first orders where none are known is 32 and 33, "we challenge the reader to find a Costas array of either of these n."

B. Complexity of Search Methods

The definition of a Costas array as a permutation matrix with special restrictions does not directly lead to a simple method of finding them because the Costas condition is not easily posed in a simple way such as a set of constraint equations. The only known way to obtain all Costas arrays for a given order is an exhaustive search. The number of permutations of order N is $N!$, while the number of Costas arrays of order N increases to a maximum of 21,104 for $N = 16$ after which the number drops rapidly. Fig. 4 shows a curve for orders 1–26. Results presented for the first time here include that there are 200 Costas arrays of order 24, 88 of order 25, and 56 of order 26.

Exhaustive search, such as sequential generation of all $N!$ permutation matrices and examining the difference matrix to determine which have the Costas property, is prohibitively slow for large N. *MacTech*, a journal for serious Apple Macintosh developers and users, had a monthly contest, "Programmers Challenge," until 2002. For December 1999, the contest was generation of all Costas arrays of order 24 [24]. The winner produced a fast method of doing a search, but it was not fast enough to produce a search for order 24. The article was not clear on how high they did go. Until this disclosure, an exhaustive search had not been completed for orders greater than 23 [23, 25].

C. Moore's Law and Orders 32 and 33

We can estimate when we can perform a search for order 32 from Moore's law. Moore's law is interpreted here as meaning that computational power available at a given resource level doubles every one and a half years. Our measurements with several variations of backtrack programming

methods require about five times more computational resources for each succeeding order; this is observed to be independent of the order, the algorithm, the implementation, and the computer, and is consistent with [20] and our understanding of that work as interpreted here. From these estimates, all other things being equal, a new order will be searched each three and a half years. Because we succeeded in the order 26 search in 2004, these casual figures predict that order 32 will be searched by 2025 and order 33 results will follow about three and a half years later. Results will likely be available sooner from sources such as application of casually available massive resources in developing and testing emerging computer technology in algorithms with significant random branching such as backtrack programming, development of a method faster than backtrack programming with preclusion or enhancements on existing methods, application of more resources, or other influences. A search for order 32 with existing methods will require about 15,000 times the resources that the authors used to search order 26.

D. Combinatoric Collaboration

1) *Overview*: Our observation in subsection C is that the exhaustive search problem is, at best, exponential-hard. Exploitation of symmetries, efficiencies of implementation, and other preclusion measures (see subsection E) will scale the curve, but the problem remains exponential-hard with a factor of about five for each increment of order. An enabling technology to find the few remaining unknown Costas arrays in the near term is combinatoric collaboration. Since parallelism seems to be the path that Moore's law is taking at the time of this writing, networked collaboration as we have done with some of our resources is an important step for results in the near term.

Our combinatoric collaboration consisted of three elements:

1) the algorithm, with an initialization shell that allows the algorithm to be started for three, four, or five column indices that are set as inputs;

2) available, idle computational resources including Suns owned by Lockheed Martin Advanced Technology Laboratories on nights and weekends, personal computers belonging to all authors, a Beowulf cluster of castoff machines, and a number of computers left idle on weekends;

3) a bookkeeping and dynamic problem parsing and allocation that supported collaboration of separate facilities.

2) *Collaborative Methods*: The collaboration operated by allocation of different parts of the problem to the various resources. The parts of the problem are denoted by the first three, four, or five column indices that represent the beginning of the block of cases searched by that run. The data is a log file of the cases started or completed. The search algorithm can be stopped at any time on any resource, leaving a truncated log file. These log files were sent to a centralized location and provided to an automated bookkeeping scheme that kept track of the cases run and the Costas arrays found. Blocks of cases to be run are dynamically allocated to resources as the project proceeds. In this way, resources of diverse types are coordinated with little or no duplication.

E. Exhaustive Search with Backtrack Programming Method

The method most often used, including in the Mac Challenge solution, is backtrack programming with preclusion [10, 21, 24–26]. As often applied to exhaustive search of permutation matrices for Costas arrays, this method sequentially builds up the column index representation. The criterion functions are the requirements that no column index may be repeated, and that no row in the difference matrix may contain duplication. The method builds a mask of allowed values for the next row using the portion of the column indices and difference matrix that exist at that point, taking the next available value in the active row, and proceeding to the next row. When an available value is found for the last row, a Costas array has been found. When a row is found that has no allowable values, the previous row is searched for the next available column index. When available indices in the first row are exhausted, the search is complete. The use of this method has the effect of reducing the complexity of an exhaustive search from $N!$-hard to exponential-hard, with time for searching order N proportional to about 5^N.

F. New Exhaustive Search Method

1) *Overview*: We disclose an improved algorithm that is based on computing and storing the necessary elements of the difference table as a stack of bitmasks. The disallowed column index values of the next row are determined by combining the vector difference information of this table with the previously used column data.

The method is recursive. The Costas array is found, beginning with the first row, in column index representation. At a given point in the search:

1) The depth of recursion is equal to the number of rows defined.

2) The search is executed through computation of the portion of the difference matrix that is defined (e.g. the upper triangular part bounded by the column indices found).

The difference matrix is represented as a set of masks, one for each row, with the differences represented as the position of a bit set to 1. This innovation allows efficient implementation in nearly any commonly used architecture.

The method begins with definition of the first row index, or the first few, defined by a trivial generation of a few column indices consistent with the Costas property. At the beginning of a recursion level, the array of masks representing the difference matrix for the previous recursion level is updated by setting the bits appropriate to the newest column index and stored in the stack as local variables for the current recursion level. The method then checks the bits in the row of the difference matrix for an allowable value. If it finds a bit that is not set, it sets the current column index to that bit position and goes to the next level of recursion. If no bits are available, the algorithm backtracks to the previous level. If the recursion level advances to the order of the array, the last column index has been found, and a valid Costas array is defined.

2) *Integration of Search with Collaboration*: The first few levels are used to start a block of indices to search. Three and four levels were used for order 24 and 25, and four and five levels were used for order 26.

3) *Word Length Considerations*: The elements of the difference matrix will vary from $-(N-1)$ to $+(N-1)$, so an offset of at least $(N-1)$ must be used to provide nonnegative bit positions. The length of the mask must be at least $2 \cdot N - 1$ bits, which means that the mask must have more than 32 bits for orders greater than 16. Here we show how to use two 32-bit words to implement a 64-bit mask for the base algorithm.

We begin with a few row indices, with the masks representing the rows of the difference matrix. The remainder of the branch is analyzed recursively according to the method presented above, and we use two 32-bit bitmasks to represent each row of the difference matrix. We use one 32-bit bitmask for positive differences and another to represent negative differences. We use a bitwise logical AND operation to set bits in each bitmask, and we avoid branching by using each difference to set bits in both masks and storing zeros in the lookup table to allow a no-operation bit set when negative differences are to set bits in the positive difference mask, and vice-versa. This method is explained in detail in [26].

G. Exploiting Symmetries

To enable symmetry optimizations beyond the horizontal flip elimination (by only searching the lower half of the possible starting indices), we developed a middle-out approach that allowed two out of the three symmetries to be leveraged easily. The horizontal flip symmetry is eliminated precisely as before, by scanning only half of any row. In the case of $N = 26$ for the middle-out approach, row 12 (zero-based indexing) is scanned from 0 to 12. The vertical flip symmetry may then be eliminated by scanning row 13 from within the margins set by row 12 as specified by the following loop:

```
for  (row[12] = 0; row[12] < 13; row[12] + +)
for  (row[13] = row[12] + 1; row[13] < 26
       − row[12]; row[13] + +)
{...}
```

The penalty associated with keeping track of variables and checking constraints in two directions simultaneously was too costly, so alternative techniques were developed for the top-down approach.

1) *Corner Dot Extension*: If all arrays of order N are known, all $(N+1)$ arrays with zero for a starting index are known simply by appending a dot in the upper left hand corner of all order N arrays, and screening the results for the Costas property. Therefore, all arrays with a corner dot may be omitted from the search if the knowledge of the previous order is complete.

2) *Progressive Redundancy Elimination*: If all Costas arrays with starting index 0 are known, then through rotation symmetry, all the arrays with a dot in any corner (as represented below) have been covered:

```
1***  ***1  ****  ****
****  ****  ****  ****
****  ****  ****  ****
****  ****  ***1  1***
```

So from that point on, no arrays with a dot in any corner need to be searched. This concept may be extended, so when all arrays with a starting index of 1 are checked, by rotation and transpose symmetries, all of the seven polymorphs are fully covered as well:

```
*1**  ****  **1*  ****  ****  ****  ****  ****
****  1***  ****  ***1  ****  ****  ****  ****
****  ****  ****  ****  ***1  ****  ****  1***
****  ****  ****  ****  ****  **1*  *1**  ****
```

Therefore, after exhausting all arrays with starting indices of 0 and 1, there is no need to look for a dot in any corner, or in any outer row or column position that is one step away from a corner, because it would have been covered by the previous search. This concept is extended as the starting index progresses, as deep as the outer control loop but not to the inner recursion, since that must be kept as streamlined as possible. As an example, for a starting index of row[0] = 4, rows 1–3 are scanned from 1 to 24. If this

TABLE X
All Fundamental Costas Arrays of Order 24

0	2	17	18	23	10	3	12	1	15	22	5	21	6	14	11	9	19	13	8	7	20	16	4
0	3	20	7	5	12	18	4	15	14	22	23	11	1	13	17	6	8	21	16	9	19	10	2
0	4	23	15	11	5	7	2	1	10	3	17	20	8	19	9	6	14	21	12	13	18	16	22
0	11	15	21	17	7	8	6	20	12	9	19	14	2	4	23	22	1	18	5	13	16	10	3
1	15	19	12	7	10	20	0	11	17	3	21	23	22	5	6	4	9	18	2	14	8	16	13
2	13	8	23	11	15	5	14	0	7	1	19	12	10	9	6	16	21	3	4	17	20	22	18
3	2	15	12	21	14	10	4	6	23	9	20	7	19	22	13	5	0	16	1	11	17	18	8
3	14	10	20	13	11	6	23	22	19	1	16	2	21	0	8	17	7	12	15	4	5	18	9
3	21	8	10	16	7	2	18	11	19	4	17	0	1	13	20	23	22	14	12	6	15	5	9
4	3	9	19	21	17	15	0	11	14	22	1	23	10	2	20	8	5	12	13	7	16	6	18
4	7	9	20	19	14	8	22	12	17	5	1	21	2	11	23	10	3	16	0	18	15	6	13
4	9	15	2	13	22	21	5	8	1	23	3	18	12	0	17	14	16	20	10	6	7	19	11
4	13	7	8	21	17	5	15	22	1	18	23	0	20	12	11	2	10	3	14	16	19	9	6
4	14	15	6	10	9	1	13	22	19	21	2	23	7	0	20	16	5	3	17	12	18	8	11
4	19	14	16	8	21	7	23	2	11	15	3	13	10	0	22	9	5	12	20	1	6	18	17
5	2	11	14	16	12	13	23	9	7	18	1	17	8	3	22	21	15	0	4	19	6	20	10
5	14	3	21	13	15	0	19	1	9	20	4	16	12	2	18	22	23	6	11	17	10	8	7
5	14	8	18	20	3	4	16	15	12	17	7	23	0	21	10	2	19	1	9	22	6	13	11
5	19	3	15	11	21	13	2	22	20	6	10	7	1	0	23	16	17	4	12	14	9	18	8
6	9	16	14	7	11	0	17	2	21	20	15	23	13	5	1	22	10	12	18	4	19	3	8
6	21	14	9	13	18	19	7	3	5	4	23	20	0	12	15	2	11	22	8	16	10	1	17
6	21	15	16	12	2	13	4	20	17	5	0	10	8	14	23	7	19	18	22	9	1	3	11
7	12	18	13	20	10	9	17	1	3	19	23	6	2	5	15	0	21	14	11	22	4	16	8
7	14	2	11	5	21	18	20	19	10	0	4	23	6	12	22	3	1	15	8	13	16	17	9
8	17	1	5	13	19	20	6	23	0	18	15	10	21	14	4	16	3	2	22	11	7	9	12

technique were not employed, they would be scanned from 0 to 25, resulting in redundant coverage.

3) *Progressive Impossibility Exclusion*: If the search has covered all starting indices up to, but excluding the middle index for odd N, or the middle two indices for even N, there is no need to search further. Consider the following possible configurations of dots in the outer rows and columns of an even order array, that could not have been covered by the preceding search:

```
001000  001000  000100  000100
0****0  0****0  0****0  0****0
0****1  1****0  0****1  1****0
1****0  0****1  1****0  0****1
0****0  0****0  0****0  0****0
000100  000100  001000  001000
```

These satisfy the constraint that there exists exactly one dot in any row or column, but are all fruitless branches because the dots in the perimeter form a parallelogram, which results in a difference table violation corresponding to sidelobes of height 2 in the autocorrelation function.

H. Symmetry-Based Searches

In [21], assumptions of symmetry have been used to provide faster search schemes that allow results such as showing that there are no symmetric Costas arrays of order 24 and finding symmetrical Costas arrays of higher orders. In [27] the number of symmetrical Costas arrays is enumerated to order 32 (27 : 7, 28 : 0, 29 : 5, 30 : 4, 31 : 0, and 32 : 0), a result obtained by symmetry-constrained searches and reported in [22], which also reports searches over antireflective and consecutive symmetries.

V. RESULTS AND CONCLUSIONS

A. Results of Exhaustive Searches for Orders 24, 25, and 26

There are 200 Costas arrays of order 24, 88 of order 25, and 56 of order 26. Table X, Table XI and Table XII list all fundamental Costas arrays of order 24, 25, and 26, respectively. Each entry in the tables is a basis for a set of eight, except those in boldface which are symmetrical and, thus, a basis for a set of four. Preliminary results of this effort were given in [25] and [26]. The last two Costas arrays in Table XII are first presented here.

B. Known Costas Arrays of Orders to 200

Number-theoretic generators provide Costas arrays for a wide variety of orders, and there are some extensions based on augmenting or decrementing rows and columns of existing Costas arrays [11]. We have

TABLE XI
All Fundamental Costas Arrays of Order 25, Symmetrical in Boldface

0	6	2	4	7	20	21	3	8	18	15	14	12	5	23	17	24	10	19	9	13	1	16	11	22
1	8	14	24	19	12	3	11	22	2	18	4	16	15	20	21	23	0	17	9	6	10	13	7	5
1	**9**	**5**	**2**	**14**	**3**	**10**	**16**	**17**	**0**	**20**	**8**	**12**	**7**	**18**	**23**	**13**	**11**	**4**	**22**	**6**	**19**	**21**	**24**	**15**
1	20	5	21	8	10	24	7	17	15	16	13	12	18	0	4	11	6	2	19	22	14	3	23	9
2	**20**	**8**	**14**	**23**	**10**	**13**	**11**	**0**	**4**	**21**	**18**	**12**	**17**	**19**	**1**	**22**	**6**	**7**	**3**	**15**	**5**	**24**	**9**	**16**
3	2	11	17	4	8	10	15	9	5	20	1	12	24	14	22	23	0	18	13	6	16	7	21	19
3	19	11	18	16	7	8	20	9	14	2	4	12	15	24	21	1	23	22	6	0	13	17	10	5
4	5	14	13	19	17	9	24	18	21	11	15	12	0	7	2	10	1	23	16	3	8	20	22	6
5	10	21	17	14	3	20	1	19	22	2	9	13	11	4	23	24	18	8	0	16	15	6	12	7
5	20	13	11	14	3	16	1	10	7	15	19	24	4	23	17	0	6	18	2	22	21	12	8	9
6	10	16	4	3	20	9	23	2	18	0	15	13	14	21	5	17	22	24	7	1	11	19	12	8
6	19	3	2	13	21	7	16	10	24	22	14	1	20	15	0	17	5	9	12	18	23	4	11	8

TABLE XII
All Fundamental Costas Arrays of Order 26, Symmetrical in Boldface

0	17	15	25	7	6	19	10	16	23	11	12	21	13	24	18	20	9	5	2	14	4	8	3	22	1
1	5	13	0	3	9	21	16	6	15	4	11	25	24	22	18	10	23	20	14	2	7	17	8	19	12
1	17	10	21	3	23	18	8	14	11	25	9	13	0	7	12	6	19	20	16	15	24	4	2	5	22
2	**5**	**7**	**20**	**4**	**22**	**15**	**13**	**3**	**8**	**19**	**14**	**18**	**6**	**0**	**9**	**10**	**16**	**23**	**12**	**24**	**21**	**17**	**25**	**1**	**11**
2	**8**	**22**	**23**	**15**	**17**	**11**	**1**	**20**	**9**	**21**	**4**	**0**	**3**	**19**	**6**	**16**	**24**	**5**	**10**	**7**	**14**	**12**	**25**	**18**	**13**
4	2	7	22	11	19	0	13	1	25	9	8	5	23	24	18	10	17	12	15	21	3	14	16	20	6
4	19	18	9	1	14	20	7	16	6	11	0	24	10	22	17	2	23	25	5	3	13	21	15	8	12
5	8	20	16	18	15	4	25	13	19	6	10	2	0	9	24	14	21	3	23	22	7	1	11	12	17

extended upon these. These generated Costas arrays provide an existence proof of Costas arrays for most orders and prove certain early conjectures in [11] such as no upper bound on the number of Costas arrays or the orders for which Costas arrays exist.

Fig. 3 shows our curve modified from that of [20], a smooth approximation of the total number of Costas arrays for orders up to 28. Fig. 5 shows this same curve against a background of points representing the numbers of generated Costas arrays of order up to 200. For orders between 6 and 26, the generators do not find all Costas arrays, so the points fall below the curve. Since the minimum non-zero number of Costas arrays of a given order is four (when only one symmetrical Costas array exists as for orders 55, 67, and others)this left the bottom of the plot free to show the cases where no Costas arrays are known. As the note on the plot states, we use this to show the orders, such as 32, 33, 43, 48, 49, etc. that have no known Costas arrays as having one on the plot; this allows the orders for which no Costas arrays are known to be highlighted there.

With this work, all existing Costas arrays up to order 26 are available, and number-theoretic generators with extensions provide plentiful Costas arrays of larger orders. The authors have implemented all Costas array generators and extensions reported in [11] and added others and can generate Costas arrays of arbitrarily large order. In particular, [11] and [19] report none of order 53, but the author's previous generalizations on the methods of [11] produce one

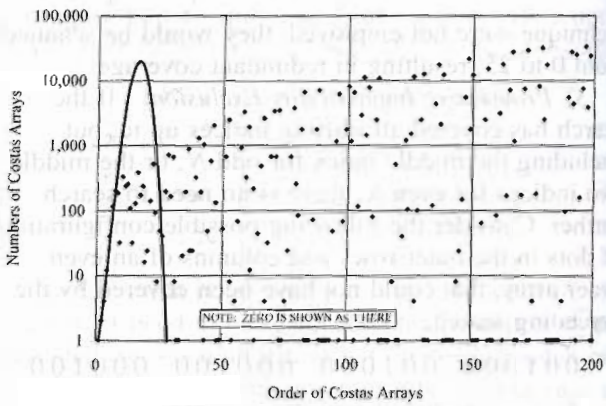

Fig. 5. Costas arrays, all to order 26 (solid curve), generated to order 200 (points).

given here at the end of subsection D. As Fig. 5 shows, the number available generally increases with order, and the authors have generated Costas arrays of very large orders.

Two new extensions presented here are based on variations of the Lempel-Golomb generator. The new extensions produce a number of Costas arrays.

C. Observations and Open Questions

Fig. 3 and Fig. 5 show that the probabilistic estimation of numbers of Costas arrays as a function of order, as predicted by [20], breaks down above

order 26. Number-theoretic constructions from [11] and generalizations by the authors and others produce 196 Costas arrays of order 27, while the probabilistic prediction is 7.8. Above order 28, the probabilistic estimate drops below one, and has an asymptotic form

$$\mathrm{Cbar}(N) \approx \frac{1}{\sqrt{12N}} \cdot \exp\left(-\frac{K \cdot N^2}{12}\right), \quad N \gg 28 \qquad (23)$$

for large N. We see the parabolic shape of this curve in the figures. The maximum peak numbers of generated Costas arrays appears, from Fig. 5, to be bounded by about N^2 for large N. Clearly, there are two influences on the number of Costas arrays as a function of order, and the probabilistic curve dominates for orders below about 25 and the numbers determined by number-theoretic considerations dominate for orders above 26. We see from Table IX above that the methods first presented here find Costas arrays of order 52 and below. Similar observations apply to the spin generalizations, the method presented in [19], and the restricted searches in [21], [22], and other methods that use searches focused on variations of the overconstrained Costas arrays found by the number-theoretic generators.

One possible conclusion is that as N increases, the numbers of Costas arrays that are not found by the number-theoretic generators and their generalizations decreases and the probability of their existence declines. It appears that none are known of orders much larger than 50. These are focused searches in which the probability of a Costas array is much higher than that used in the derivations of the probabilistic curves in [20], but the probability of existence of even these decreases as N increases.

This brings us to the question of the existence of Costas arrays of order 32 or 33, the lowest orders for which no Costas arrays are known at this time. Since the focused searches do find new Costas arrays for higher orders, the possibility of Costas arrays for these orders cannot be ruled out on the basis of our observations.

ACKNOWLEDGMENT

The authors thank Lockheed Martin Advanced Technology Laboratories for allowing use of idle computers and unused machine cycles, and for proofreading assistance. The authors also thank others who donated surplus computers and allowed networked and other machines to be used while idle. The authors owe a debt to Bill Haloupek and Greg Coxson of Lockheed Martin MS2 Moorestown for provoking initial interest in Costas arrays.

REFERENCES

[1] Woodward, P. M.
Probability and Information Theory.
New York: McGraw-Hill/Pergammon, 1953, 120; reprinted by Artech as ISBN 0-8900-6103-3 (1980); attributes the ambiguity function to J. Ville, *Câbles et Transmission*, No. 1, 1948, 61.

[2] Costas, J. P.
A study of a class of detection waveforms having nearly ideal range-Doppler ambiguity properties.
Proceedings of the IEEE, **72**, 8 (Aug. 1984).

[3] Dick, C., and Harris, F.
FPGA DSPs—the platform for next-generation wireless communications.
RF Design, (Oct. 2000), 56–66.

[4] Bocko, M. F.
Data hiding in digital audio files.
IEEE Signal Processing Society, Rochester chapter, Mar. 5, 2003.

[5] Tirkel, A. Z., van Schyndel, R. G., and Osborne, C. F.
A two-dimensional digital watermark.
DICTA'95, University of Queensland, Brisbane, Dec. 5–8, 1995, 378–383.

[6] Costas, J. P.
Project Medior—A medium-oriented approach to sonar signal processing.
HMED Technical Publication R66EMH12, GE Syracuse NY (now Lockheed Martin Marine Systems and Sensors, Syracuse), Jan. 1966.

[7] Costas, J. P.
Medium constraints on sonar design and performance.
In *FASCON Convention Record*, 1975, 68A–68L.

[8] Freedman, A.
Trains of Costas bursts and their ambiguity function.
M.S. thesis, Tel Aviv University, Oct. 1985.

[9] Levanon, N., and Mozenson, E.
Orthogonal train of modified Costas pulses.
In *Proceedings of the IEEE 2004 Radar Conference*, Apr. 26–29, 2004, 255–259; ISBN 0-7803-8234-X.

[10] Golomb, S., and Baumert, L.
Backtrack programming.
Journal of the ACM, (Oct. 1965), 516–524.

[11] Golomb, S. W., and Taylor, H.
Constructions and properties of Costas arrays.
Proceedings of the IEEE, **72**, 9 (Sept. 1984), 1143–2263.

[12] Golomb, S. W.
Algebraic constructions for Costas arrays.
Journal of Combinatorial Theory, Series A, **37** (1984), 13–21.

[13] Moreno, O.
Survey on Costas arrays and their generalizations.
In Jong-Seon No, Hong-Yeop Song, Tor Helleseth, and P. Vijay Kumar (Eds.), *Mathematical Properties of Sequences and Other Combinatorial Structures*, Boston: Kluwer Academic Publishing Co., 2003; ISBN 1-4020-7403-4 (a collection of peer-refereed papers for a three-day conference celebrating the 70th birthday of Solomon W. Golomb—see Preface of this work), 55–64; see especially the top of p. 62 where reference is made to the proof of existence of Costas arrays of order $p^n - 3$ at [O. Moreno and J. Solero, "Computational approach to Conjecture A of Golomb," *Congressus Numerantium*, vol. 70, 1990, 7–16].

[14] Golomb, S., and Taylor, H.
The T-4 and G-4 constructions for Costas arrays.
IEEE Transactions on Information Theory, **38**, 4 (July 1992), 1404–1406.

[15] Stewart, I.
Galois Theory (2nd ed.).
London: Chapman & Hall/CRC, 1989, Theorem 16.4, 157; ISBN 0-412-34550-1.

[16] Garg, H. K.
Digital Signal Processing Algorithms.
Boca Raton, FL: CRC Press, 1998, 138–141; ISBN 0-8493-7178-3.

[17] Sklar, B.
Digital Communications.
Englewood Cliffs, NJ: Prentice-Hall, 1988, 447–450.

[18] Taylor, H.
Singly periodic Costas arrays are equivalent to polygonal path Vatican squares.
In Jong-Seon No, Hong-Yeop Song, Tor Helleseth, and P. Vijay Kumar (Eds.), *Mathematical Properties of Sequences and Other Combinatorial Structures*, Boston: Kluwer Academic Publishing Co., 2003; ISBN 1-4020-7403-4 (a collection of peer-refereed papers for a three-day conference celebrating the 70th birthday of Solomon W. Golomb—see Preface of this work), 45.

[19] Rickard, S.
Searching for Costas arrays using periodicity properties.
Presented at the IMA International Conference on Mathematics in Signal Processing, The Royal Agricultural College, Cirencester, UK, Dec. 2004.

[20] Silverman, J., Vickers, V. E., and Mooney, J. M.
On the number of Costas arrays as a function of array size.
Proceedings of the IEEE, (July 1988), 851–853.

[21] Moreno, O., Rameirez, J., Bollman, D., and Orozco, E.
Faster backtracking algorithms for the generation of symmetry-invariant permutations.
Journal of Applied Mathematics **2**, 6 (2002), 277–287.

[22] Brown, C. P., Cenkl, M., Games, R. A., Rushanan, J. J., and Moreno, O.
New enumeration results for Costas arrays.
In *IEEE Symposium on Information Theory*, (Jan. 1993), 405.

[23] Rickard, S.
The quest for Costas arrays.
Presented at the IEEE AES Regional Meeting, Rowan University, Oct. 8, 2002.

[24] MacTech, Programmer's Challenge.
Available at http://www.mactech.com/progchallenge/, click on "Costas Arrays (December 1999)."

[25] Beard, J. K.
Costas arrays, properties and generators.
Presented at the IEEE AES Regional Meeting, Rowan University, Oct. 8, 2002.

[26] Beard, J. K., Russo, J. C., Erickson, K., Moneleone, M., and Wright, M.
Combinatoric collaboration on Costas arrays and radar applications.
In *Proceedings of the IEEE 2004 Radar Conference*, Apr. 26–29, 2004, 260–265; ISBN 0-7803-8234-X.

[27] Taylor, H.
Costas arrays.
In Charles J. Colbourn and Jeffrey H. Dinitz (Eds.), *The CRC Handbook of Combinatorial Designs*, Boca Raton, FL: CRC Press, 1996, 256–260, see enumeration table on p. 259; ISBN 0-8493-8948-8.

James K. Beard (M'64—LM'04—LSM'05) was born in Austin, TX in 1939. He received a B.S. degree from the University of Texas at Austin in 1962, an M.S. from the University of Pittsburgh, Pittsburgh, PA, in 1963, and the Ph.D. from the University of Texas at Austin in 1968, all in electrical engineering. He studied for his Ph.D. under a GSRF Fellowship and an NSF Fellowship.

Between 1959 and 2004, he worked in Government laboratories, industry, and as an individual consultant. Employers include precursors or current divisions of Northrop Grumman, Raytheon, and Lockheed Martin, most recently Lockheed Martin MS2 in Moorestown, NJ. He is currently an individual consultant based in Medford, NJ near Philadelphia. Current research interests include system engineering solutions to homeland defense issues, estimation and decision theory, radar and communications concept and waveform design, and digital radar concepts. He is an adjunct professor at Temple University in Philadelphia and teaches graduate and undergraduate analog and digital communications.

Dr. Beard is the author of a number of symposia papers and a book, *The FFT in the 21st Century* (Kluwer, 2003).

He is a member of AIAA, AOC, SPIE, and ACM. He was publications chairman for FUSION2005. He is a member of Phi Eta Sigma, Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.

Jon C. Russo was born in Geneva, NY in 1969. He received a B.S. in electrical engineering from Cornell University, Ithaca, NY, in 1992. He received a Masters in Engineering in 1993, in the field of digital signal processing, from the same university.

While at Cornell, he was a teaching assistant for electronic design lab and other classes. He went on to join the research team at Lockheed Martin Advanced Technology Labs, working in signal processing, radar, hardware design, and reconfigurable computing and compiler technologies. Research interests include cognitive architectures, communications processing, and speech and image analysis.

Mr. Russo is a coauthor of [26] and papers in high performance computing and information technologies.

Keith G. Erickson (SM'03—M'04) was born in Cherry Hill, NJ in 1982. He received his B.S. in computer engineering from the New Jersey Institute of Technology, Newark, in 2004.

Mr. Erickson joined Lockheed Martin Maritime Sensors and Systems, Moorestown, NJ in 2004. He is currently assigned to development of emerging technologies in combat weapons systems.

Mr. Erickson is a coauthor of [26]. He was President of the SGA and of the Albert Dorman Honors College at the New Jersey Institute of Technology. He also was President of Team Corona, a joint Burlington, NJ, County College and New Jersey Institute of Technology team that built an electric car.

Michael C. Monteleone III (SM'03—M'04) was born in Toms River, NJ. He received his A.S. in engineering from Ocean County College, NJ, and received his B.S. in computer engineering with a minor in applied mathematics from the New Jersey Institute of Technology, Newark, in 2004 and is pursuing a Masters degree.

Mr. Monteleone is with the Department of Defense under the Department of the Army—Defense Communications and Army Transmission Systems–Wideband Control. His current research interests include broadband communications theory and design.

Mr. Monteleone is a coauthor of [26] and was Vice President and lead design engineer of Team Corona, a joint Burlington, NJ, County College and New Jersey Institute of Technology team that built an electric car.

Michael T. Wright (M'03) was born in Lakewood, NJ, in 1981. He received a B.S. degree from the New Jersey Institute of Technology, Newark, in 2004 in computer engineering.

He is currently a developer at Picatinny Arsenal on the Mortar Fire Control Systems project. His research interests include computer architectures, multiprocessor virtual supercomputers, and combinatoric collaboration.

Mr. Wright is a coauthor of [26].