Outer-Product Matrix Representation of Optical Orthogonal Codes

Hossein Charmchi and Jawad A. Salehi, *Member, IEEE*

***Abstract*—In this letter, we introduce a whole new approach in defining and representing optical orthogonal codes (OOCs), namely, outer-product matrix representation. Instead of applying commonly used approaches based on inner product to construct OOC codes, we use the newly defined approach to obtain a more efficient algorithm in constructing and generating OOC codes. The outer-product matrix approach can obtain a family of OOC codes with a cardinality closer to the Johnson upper bound, when compared with the previously defined accelerated greedy algorithm using the inner-product approach. We believe the new look introduced in this letter on OOCs could help to devise new approaches in designing and generating OOC codes, using the rich literature in matrix algebra.**

***Index Terms*—Design techniques, optical code-division multiple access (CDMA), optical networks, optical orthogonal codes (OOCs).**

I. Introduction

EVER since the introduction of important optical orthogonal codes (OOCs) and their use in optical code-division multiple-access (CDMA) systems [1]–[7], a trend has been set to explore efficient algorithms in constructing and generating the above codes by mathematicians and sequence designers [8]–[12]. Up until now, most approaches were based on the initial inner-product definition of OOCs. Using the above definition, most algorithmic designs would follow the mathematical languages of combinatorics, graph theory, Steiner numbers, etc.

However, in this letter, we present a new approach based on outer-product matrices of OOCs. This approach, we believe, will give, for the first time, the mathematical language based on matrix algebra, with its tremendous body of literature, to possibly develop a more efficient algorithm in constructing and generating OOC codes.

A major weakness with most combinatorics or projective geometry-based algorithms is that they can only support certain code lengths, code weights, and auto- and cross-correlation integer values that have special properties. In other words, most algorithms developed so far can not be easily extended to generalize for all integers of choice on code length, weight, and auto- and cross-correlation constraints. We believe by shifting the fundamental definitions and properties of OOC from the inner-product to outer-product matrix may help to remove the above bottleneck in designing and generating OOCs.

Paper approved by W. C. Kwong, the Editor for Optical Communications of the IEEE Communications Society. Manuscript received April 3, 2005; revised November 14, 2005. This work was supported in part by the Hi-Tech Industries Center of Iran, Tehran, Iran.

The authors are with the Optical Networks Research Lab, Electrical Engineering Department, Sharif University of Technology, Tehran, Iran (e-mail: charmchi@ee.sharif.edu; jasalehi@sharif.edu).

Digital Object Identifier 10.1109/TCOMM.2006.876839

In Section II of this paper, we first present the basic definition of OOCs based on outer-product matrix representation, then discuss the fundamental properties of OOCs based on the above matrices. In Section III, we present the algebraic matrix properties of representing OOCs. In Section IV, we present the first algorithm in constructing and generating OOC codes based on their matrix representation, which proves to have larger cardinality when compared with its inner-product-based counterpart, namely, the accelerated greedy algorithm [3].

II. Outer-Product Matrix Representation

Up until now [1]–[12], OOCs have been defined based on the inner product of two code vectors. According to [3], an $(n, w, \lambda_a, \lambda_c)$ OOC C is a collection of binary n-tuples, each of Hamming weight w. If we denote t to be the tth cyclic shift of code vector $\mathbf{x}$, i.e., $\mathbf{x}(t)$, then the two needed fundamental properties can be written as follows.

1) Auto-correlation: For any $\mathbf{x} = (x_0 x_1 \cdots x_{n-1}) \in C$ and any integer $0 < \tau < n$

$$\mathbf{x}(t) \cdot \mathbf{x}(t+\tau) = \mathbf{x}(t)\mathbf{x}^T(t+\tau) = \sum_{t=0}^{n-1} \mathbf{x}_t \mathbf{x}_{t \oplus \tau} \leq \lambda_a. \quad (1)$$

2) Cross-correlation: For any $\mathbf{x} = (x_0 x_1 \cdots x_{n-1}) \in C$ and $\mathbf{y} = (y_0 y_1 \cdots y_{n-1}) \in C$, such that $\mathbf{x} \neq \mathbf{y}$ and any integer τ

$$\mathbf{x}(t) \cdot \mathbf{y}(t+\tau) = \mathbf{x}(t)\mathbf{y}^T(t+\tau) = \sum_{t=0}^{n-1} \mathbf{x}_t \mathbf{y}_{t \oplus \tau} \leq \lambda_c. \quad (2)$$

Since OOCs were defined in terms of periodic correlation, the addition in the subscripts above, denoted "$\oplus$," is modulo-n, "$\cdot$" denotes the inner product of two code vectors, and superscript "T" implies transpose. In general, to investigate the above two properties using the inner-product approach, the inner products should be calculated for various τ.

However, the definition of OOCs can also be presented based on the outer product of two code vectors, which results in matrix representation and crystallizes many approaches based on matrix algebra to find new design techniques for OOCs.

The first matrix we consider is obtained by the outer product of a vector with itself. For this purpose, we consider each codeword $\mathbf{x}$ as a vector of length n (or a $1 \times n$ matrix), for which the related $n \times n$ matrix will be

$$\mathbf{D}_{\mathbf{xx}} = [d_{ij}]_{n \times n} = \mathbf{x} \times \mathbf{x} = \mathbf{x}^T \mathbf{x} \quad (3)$$

where "×" means the outer product of two vectors, and

$$d_{ij} = x_i x_j = \begin{cases} 1, & x_i = x_j = 1 \\ 0, & \text{otherwise.} \end{cases}$$

Example 1: Let $n = 5$, $w = 2$, $\lambda_a = \lambda_c = 1$, and $\mathbf{x} = (10100)$; then

$$\mathbf{D_{xx}} = \mathbf{x}^T\mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} (1\ 0\ 1\ 0\ 0) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The second matrix considered is the outer product of two different code vectors. For any two code vectors $\mathbf{x} = (x_0 x_1 \cdots x_{n-1})$ and $\mathbf{y} = (y_0 y_1 \cdots y_{n-1})$, $x_i, y_i \in \{0, 1\}$, we associate an $n \times n$ matrix

$$\mathbf{D_{xy}} = [d_{ij}]_{n\times n} = \mathbf{x} \times \mathbf{y} = \mathbf{x}^T\mathbf{y} \tag{4}$$

where

$$d_{ij} = x_i y_j = \begin{cases} 1, & x_i = y_j = 1 \\ 0, & \text{otherwise.} \end{cases}$$

Example 2: Assume a $(5, 2, 1, 1)$ OOC with two code vectors $\mathbf{x} = (10100)$ and $\mathbf{y} = (11000)$; then

$$\mathbf{D_{xy}} = \mathbf{x}^T\mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} (1\ 1\ 0\ 0\ 0) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

III. PROPERTIES OF OUTER-PRODUCT MATRIX REPRESENTATION

With the above definitions the two matrices $\mathbf{D_{xx}}$ and $\mathbf{D_{xy}}$ will have the following three properties:

1) the matrices $\mathbf{D_{xx}}$ and $\mathbf{D_{xy}}$ are (0, 1) matrices;
2) the matrix $\mathbf{D_{xx}}$ is symmetric;
3) in the matrices $\mathbf{D_{xx}}$ and $\mathbf{D_{xy}}$, the nonzero columns (rows) are equal to each other.

In the following, we show how the two needed fundamental properties of OOC codes can be captured and represented at once for various τ using outer-product matrices.

Definition: Assume a matrix $\mathbf{A} = [a_{ij}]_{n\times n}$. In this matrix, the elements a_{ij}, where $i = j$, are located on the main diagonal, and the elements which have equal $(j - i)_{\text{mod}n}$ are defined as the "$(j - i)_{\text{mod}n}$th diagonal." In Fig. 1, five elements $(n = 5)$ located on the second diagonal of a 5×5 array are shown.

Lemma 1: The value of the auto-correlation function of code vector $\mathbf{x}$ for the τth cyclic shift is equal to the summation of the elements of the τth diagonal in outer-product matrix $\mathbf{D_{xx}}$.

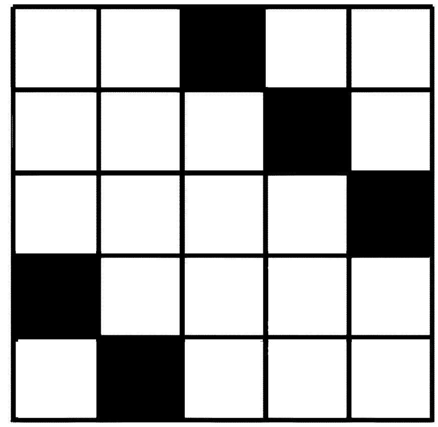

Fig. 1. Example of a 5×5 matrix with its second diagonal shown by black squares.

Proof: If d_{ij} is located on the τth diagonal, then $(j - i)_{\text{mod}n} = \tau$, or $j = i \oplus \tau$. Therefore

$$\mathbf{S_{xx}}(\tau) = \sum_{d_{ij} \in \tau\text{th diagonal of } \mathbf{D_{xx}}} d_{ij} = \sum_{i=0}^{n-1} x_i x_j = \sum_{i=0}^{n-1} x_i x_{i\oplus\tau}$$

which is equal to the summation in (1).

Lemma 2: The amount of cross-correlation function of two code vectors $\mathbf{x}$ and $\mathbf{y}$ for the τth cyclic shift is equal to the summation of the elements of the τth diagonal in outer-product matrix $\mathbf{D_{xy}}$.

Proof: Similar to *Lemma 1*

$$\mathbf{S_{xy}}(\tau) = \sum_{d_{ij} \in \tau\text{th diagonal of } \mathbf{D_{xy}}} d_{ij} = \sum_{i=0}^{n-1} x_i y_j = \sum_{i=0}^{n-1} x_i y_{i\oplus\tau}$$

which is equal to the summation in (2).

Example 3: Consider the two code vectors of *Example 2*. Fig. 2 shows the auto- and cross-correlation functions of $\mathbf{x}$ and $\mathbf{y}$. We have also three matrices

$$\mathbf{D_{xx}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{D_{yy}} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{D_{xy}} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

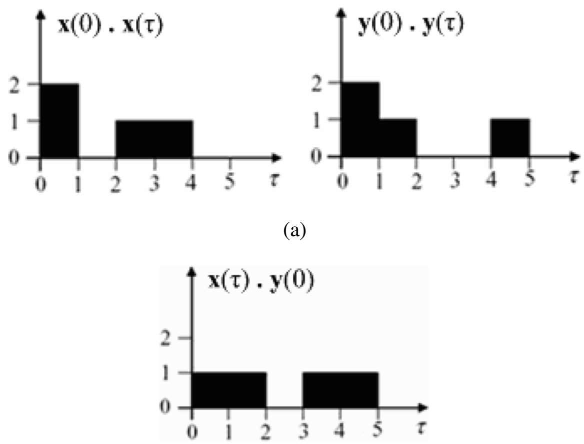

Fig. 2. (a) Auto-correlation functions of *Example 3*. (b) Cross-correlation function of *Example 3*.

Summations of elements in various diagonals of matrices are

$$\begin{aligned}\{\mathbf{S_{xx}}(0), \mathbf{S_{xx}}(1), \ldots, \mathbf{S_{xx}}(4)\} &= \{2,0,1,1,0\}\\ \{\mathbf{S_{yy}}(0), \mathbf{S_{yy}}(1), \ldots, \mathbf{S_{yy}}(4)\} &= \{2,1,0,0,1\}\\ \{\mathbf{S_{xy}}(0), \mathbf{S_{xy}}(1), \ldots, \mathbf{S_{xy}}(4)\} &= \{1,1,0,1,1\}.\end{aligned}$$

We see that the elements of these sets are equal to the correlation functions shown in Fig. 2.

We represent two other useful matrices $\mathbf{C_{xx}}$ and $\mathbf{C_{xy}}$. For a single code vector $\mathbf{x}$, we consider $\mathbf{C_{xx}}$ as

$$\mathbf{C_{xx}} = [c_{ij}]_{n\times n} = \sum_{t=0}^{n-1} \mathbf{x}(t) \times \mathbf{x}(t) = \sum_{t=0}^{n-1} \mathbf{x}^T(t)\mathbf{x}(t) \tag{5}$$

and for two code vectors $\mathbf{x}$ and $\mathbf{y}$, we consider $\mathbf{C_{xy}}$ as

$$\mathbf{C_{xy}} = [c_{ij}]_{n\times n} = \sum_{t=0}^{n-1} \mathbf{x}(t) \times \mathbf{y}(t) = \sum_{t=0}^{n-1} \mathbf{x}^T(t)\mathbf{y}(t). \tag{6}$$

Lemma 3: The elements of $\mathbf{C_{xx}}$ located on τth diagonal are equal to the auto-correlation function of vector $\mathbf{x}$ for the τth cyclic shift.

Proof: If c_{ij} is on the τth diagonal of $\mathbf{C_{xx}}$ as we saw before, then $j = i \oplus \tau$. On the other hand, from (5), we have

$$c_{ij} = \sum_{t=0}^{n-1} x_{i\oplus t} x_{j\oplus t}$$

therefore

$$c_{ij} = \sum_{t=0}^{n-1} x_{i\oplus t} x_{i\oplus \tau\oplus t} = \sum_{k=0}^{n-1} x_k x_{k\oplus\tau}. \tag{7}$$

Lemma 4: The elements of $\mathbf{C_{xy}}$ located on the τth diagonal are equal to the cross-correlation function of vectors $\mathbf{x}$ and $\mathbf{y}$ for the τth cyclic shift.

Proof: Similar to *Lemma 3*, from (6), we can write

$$c_{ij} = \sum_{t=0}^{n-1} x_{i\oplus t} y_{j\oplus t} = \sum_{t=0}^{n-1} x_{i\oplus t} y_{i\oplus\tau\oplus t} = \sum_{k=1}^{n} x_k y_{k\oplus\tau}. \tag{8}$$

Example 4: Consider the two code vectors of *Example 2*. Then

$$\mathbf{C_{xx}} = \begin{pmatrix} 2 & 0 & 1 & 1 & 0\\ 0 & 2 & 0 & 1 & 1\\ 1 & 0 & 2 & 0 & 1\\ 1 & 1 & 0 & 2 & 0\\ 0 & 1 & 1 & 0 & 2 \end{pmatrix}$$
$$\mathbf{C_{yy}} = \begin{pmatrix} 2 & 1 & 0 & 0 & 1\\ 1 & 2 & 1 & 0 & 0\\ 0 & 1 & 2 & 1 & 0\\ 0 & 0 & 1 & 2 & 1\\ 1 & 0 & 0 & 1 & 2 \end{pmatrix}$$
$$\mathbf{C_{xy}} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1\\ 1 & 1 & 1 & 0 & 1\\ 1 & 1 & 1 & 1 & 0\\ 0 & 1 & 1 & 1 & 1\\ 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Using *Lemmas 3* and *4*, we can find a way to verify which code vectors satisfy the correlation properties.

Lemma 5: Any code vector $\mathbf{x}$ satisfies the auto-correlation constraint if and only if (iff) all the elements of $\mathbf{C_{xx}}$ not located on the main diagonal are less than or equal to λ_a.

Lemma 6: Any two code vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy the cross-correlation constraint iff all the elements of $\mathbf{C_{xy}}$ are less than or equal to λ_c.

Now we can present the following theorem for the special case of $\lambda_c = 1$.

Theorem 1: In an $(n, w, \lambda_a, 1)$ OOC, with m code vectors $\mathbf{x}_i$ $(i = 1, \ldots, m)$, if the τth diagonal of $\mathbf{C}^i_{\mathbf{xx}}$ has a nonzero value, then this diagonal in all other matrices $\mathbf{C}^j_{\mathbf{xx}}$ $(i \neq j)$ is zero.

Proof: According to *Lemma 3*, if the τth diagonal of $\mathbf{C}^j_{\mathbf{xx}}$ is nonzero, $\mathbf{x}_j$ has at least two chip pulses with relative distance τ. Since $\lambda_c = 1$, no other code vector $\mathbf{x}_i$ $(i \neq j)$ can have two or more chip pulses with relative distance τ. So the τth diagonal of all other matrices should be zero (see *Example 4*).

As the result of *Theorem 1*, we conclude with the two following lemmas.

Lemma 7: In a family of OOCs with $\lambda_c = 1$, if $i \neq j$, then $\mathbf{C}^i_{\mathbf{xx}} \neq \mathbf{C}^j_{\mathbf{xx}}$.

Lemma 8: In a family of OOCs with $\lambda_c = 1$, matrix $\mathbf{D}^i_{\mathbf{xx}}$ has at most λ_a "1" elements in its τth off-principal diagonal. On the other hand, if the τth off-principal diagonal of matrix $\mathbf{D}^j_{\mathbf{xx}}$ has some "1" elements, then this diagonal in all other matrices is all zero.

Now by relating a matrix $\mathbf{D}^i_{\mathbf{xx}}$ to any codeword $\mathbf{x}_i$, we can introduce the new definition of OOCs with $\lambda_c = 1$ based on outer-product matrices. Note that because of outer-product representation, the first row of $\mathbf{D}^i_{\mathbf{xx}}$ is exactly equal to $\mathbf{x}_i$. So there is no problem for returning to 1-D representation from the 2-D outer-product matrix representation.

Definition: An $(n, w, \lambda_a, 1)$ OOC is a family of (0,1) $n \times n$ matrices $\mathbf{D}^i_{\mathbf{xx}}$ with the following properties.

1) The matrices should be symmetric and have "w" ones in their nonzero rows. Also, in each matrix, the nonzero rows should be equal.
2) At most λ_a ones should be located on each off-principal diagonal of each matrix (auto-correlation constraint).
3) If the τth off-principal diagonal of a matrix is nonzero, this diagonal in all other matrices should be zero (cross-correlation constraint).

With the above definition, we can search for matrices with the three properties in the (0,1) matrix domain.

An important issue about OOCs is the upper bound on the number of codewords for given code parameters. In the following, we once again prove the traditional upper bound [4] for the case $\lambda_c = 1$ using previous lemmas.

Theorem 2: The number of codewords of an $(n, w, \lambda_a, 1)$ OOC satisfies the inequality

$$m \leq \left\lfloor \frac{(n-1)\lambda_a}{w(w-1)} \right\rfloor . \tag{9}$$

Proof: According to property 1) of the definition, in each matrix $\mathbf{D}^i_{\mathbf{xx}}$ there are w^2 ones, w of them located on the main diagonal. From property 2), off-principal diagonals have at most λ_a "1" elements. And from property 3), the same off-principal diagonals of two matrices can not be nonzero. Each matrix has $n-1$ off-principal diagonals. So if we divide $n-1$ by $(w^2 - w)/\lambda_a$, the above inequality is obtained.

Lemma 9: $\mathbf{C}_{\mathbf{xx}}$ and $\mathbf{C}_{\mathbf{xy}}$ are circulant matrices.

Proof: In *Lemmas 3* and *4*, we saw that the elements on the τth diagonal of matrices $\mathbf{C}_{\mathbf{xx}}$ and $\mathbf{C}_{\mathbf{xy}}$ are equal to the correlation functions for the τth cyclic shift. So all the elements located on the same diagonal are equal, and this means that each row is a circular shift of the previous one [13].

IV. Matrix Search Algorithm

In this section, we present a way of searching for $\lambda_a = \lambda_c = 1$ OOCs using the properties discussed in Section III. For this purpose, we consider an $n \times n$ matrix $\mathbf{D}$ which, in fact, equals the summation of all code matrices $\mathbf{D}^i_{\mathbf{xx}}$ $(1 \leq i \leq m)$ of the desired OOC family (with m codewords $\mathbf{x}_i$, $1 \leq i \leq m$). The importance of using matrix $\mathbf{D}$ is due to property 3) of matrices $\mathbf{D}^i_{\mathbf{xx}}$, which exerts the condition that if the τth off-principal diagonal of a matrix is nonzero, this diagonal in all other matrices should be zero (cross-correlation). In a sense, matrix $\mathbf{D}$ captures in a parallel and simultaneous manner the conditions for all $\mathbf{D}^i_{\mathbf{xx}}$'s, thereby indicating its importance and usefulness in constructing a family of OOCs.

In each step of the algorithm, the position of one chip pulse of a codeword $\mathbf{x}_i$ is determined in the first row of $\mathbf{D}$ (as mentioned before, the first row of $\mathbf{D}^i_{\mathbf{xx}}$ is equal to $\mathbf{x}_i$).

We have two considerations:

1) for each code vector $\mathbf{x} = (x_0 x_1 \ldots x_{n-1})$, $x_i \in \{0, 1\}$, we assume $x_0 = 1$;
2) for the jth code in matrix $\mathbf{D}$, we denote its corresponding "1" positions by letter j, in order to distinguish the difference between the "1"s of different code matrices.

Due to the assumption of $\lambda_a = \lambda_c = 1$, the first consideration in the above causes that no "1" elements of two different code matrices, except for the first element x_0, be located in the same position of $\mathbf{D}$. Furthermore, the main diagonal of $\mathbf{D}$ is common among all matrices $\mathbf{D}^i_{\mathbf{xx}}$. Therefore, it does not help us any further in constructing different code matrices. Hence, we fill the main diagonal with * (see *Example 6*). On the other hand, the second consideration in the above helps us to separate matrices of different codes at the end of our algorithm.

In each step, when we insert one "1" element in $\mathbf{D}$, because of correlation constraints, we should not insert any "1" elements in certain positions of $\mathbf{D}$, which we refer to as forbidden positions. These forbidden positions are determined by the rules discussed in Section III, especially in the matrix-based definition of OOCs. If a position is forbidden because of the auto-correlation constraint, we fill it with $\circ$; but if it is because of the cross-correlation constraint, we fill it with $\times$. When the construction of the jth code is complete, we remove all $\circ$ marks before we start the construction of the next code matrix.

In each step, we insert the new chip pulse of the codeword currently being constructed in the first empty position of the first row of $\mathbf{D}$, after the last inserted element of the current codeword. But if there is no empty position in the first row, while the construction of the desired number of codes is not complete, we turn back one step, advance the last inserted element by one empty position, and fill its previous position with $\circ$. This, in fact, shows that with an element in this position, we can not obtain a complete codeword. However, if there is no empty position to move the last inserted element back into, then we go back two steps. It means that we remove the last inserted element, advance the element before the last inserted element by one empty position, and fill its previous position with $\circ$. But if again there is no empty position to move to, then we go back three steps, and so on.

Example 6: Consider constructing a $(13, 3, 1, 1)$ OOC with $m = 2$ codewords using the newly described algorithm above. We begin with the matrix

$$\mathbf{D} = \begin{pmatrix} * & & & & & & & & & & & & \\ & * & & & & & & & & & & & \\ & & * & & & & & & & & & & \\ & & & * & & & & & & & & & \\ & & & & * & & & & & & & & \\ & & & & & * & & & & & & & \\ & & & & & & * & & & & & & \\ & & & & & & & * & & & & & \\ & & & & & & & & * & & & & \\ & & & & & & & & & * & & & \\ & & & & & & & & & & * & & \\ & & & & & & & & & & & * & \\ & & & & & & & & & & & & * \end{pmatrix}$$

and after the first step, we obtain

$$\mathbf{D}=\begin{pmatrix}
\ast & 1 & & & & & & & & & & & \times \\
1 & \ast & \times & & & & & & & & & & \\
 & \times & \ast & \times & & & & & & & & & \\
 & & \times & \ast & \times & & & & & & & & \\
 & & & \times & \ast & \times & & & & & & & \\
 & & & & \times & \ast & \times & & & & & & \\
 & & & & & \times & \ast & \times & & & & & \\
 & & & & & & \times & \ast & \times & & & & \\
 & & & & & & & \times & \ast & \times & & & \\
 & & & & & & & & \times & \ast & \times & & \\
 & & & & & & & & & \times & \ast & \times & \\
 & & & & & & & & & & \times & \ast & \times \\
\times & & & & & & & & & & & \times & \ast
\end{pmatrix}$$

and then

$$\mathbf{D}=\begin{pmatrix}
\ast & 1 & \times & 1 & & & & & & & \times & \times & \times \\
1 & \ast & \times & 1 & \times & & & & & & & \times & \times \\
\times & \times & \ast & \times & \times & \times & & & & & & & \times \\
1 & 1 & \times & \ast & \times & \times & \times & & & & & & \\
 & \times & \times & \times & \ast & \times & \times & \times & & & & & \\
 & & \times & \times & \times & \ast & \times & \times & \times & & & & \\
 & & & \times & \times & \times & \ast & \times & \times & \times & & & \\
 & & & & \times & \times & \times & \ast & \times & \times & \times & & \\
 & & & & & \times & \times & \times & \ast & \times & \times & \times & \\
 & & & & & & \times & \times & \times & \ast & \times & \times & \times \\
\times & & & & & & & \times & \times & \times & \ast & \times & \times \\
\times & \times & & & & & & & \times & \times & \times & \ast & \times \\
\times & \times & \times & & & & & & & \times & \times & \times & \ast
\end{pmatrix}$$

which contains the complete code matrix for the first codeword. But if we continue the instructions to construct the matrix for the second codeword, we see that it is impossible to obtain any other codewords. For example, as it is clear in the following matrix, after inserting the first element of the second codeword, the first row of $\mathbf{D}$ gets filled with forbidden positions, i.e.,

$$\mathbf{D}=\begin{pmatrix}
\ast & 1 & \times & 1 & 2 & \circ & \circ & \circ & \circ & \times & \times & \times & \times \\
1 & \ast & \times & 1 & \times & \times & & & & & \times & \times & \times \\
\times & \times & \ast & \times & \times & \times & \times & & & & & \times & \times \\
1 & 1 & \times & \ast & \times & \times & \times & \times & & & & & \times \\
2 & \times & \times & \times & \ast & \times & \times & \times & \times & & & & \\
 & \times & \times & \times & \times & \ast & \times & \times & \times & \times & & & \\
 & & \times & \times & \times & \times & \ast & \times & \times & \times & \times & & \\
 & & & \times & \times & \times & \times & \ast & \times & \times & \times & \times & \\
 & & & & \times & \times & \times & \times & \ast & \times & \times & \times & \times \\
\times & & & & & \times & \times & \times & \times & \ast & \times & \times & \times \\
\times & \times & & & & & \times & \times & \times & \times & \ast & \times & \times \\
\times & \times & \times & & & & & \times & \times & \times & \times & \ast & \times \\
\times & \times & \times & \times & & & & & \times & \times & \times & \times & \ast
\end{pmatrix}$$

and advancing this element is not useful either. This can be examined easily by inserting it in other positions of the first row. So we have to turn back another step and advance the last element of the previous codeword. Therefore, we obtain the following matrix for the first code:

$$\mathbf{D}=\begin{pmatrix}
\ast & 1 & & \times & 1 & & & & & \times & \times & & \times \\
1 & \ast & \times & & 1 & \times & & & & & \times & \times & \\
 & \times & \ast & \times & & \times & \times & & & & & \times & \times \\
\times & & \times & \ast & \times & & \times & \times & & & & & \times \\
1 & 1 & & \times & \ast & \times & & \times & \times & & & & \\
 & \times & \times & & \times & \ast & \times & & \times & \times & & & \\
 & & \times & \times & & \times & \ast & \times & & \times & \times & & \\
 & & & \times & \times & & \times & \ast & \times & & \times & \times & \\
 & & & & \times & \times & & \times & \ast & \times & & \times & \times \\
\times & & & & & \times & \times & & \times & \ast & \times & & \times \\
\times & \times & & & & & \times & \times & & \times & \ast & \times & \\
 & \times & \times & & & & & \times & \times & & \times & \ast & \times \\
\times & & \times & \times & & & & & \times & \times & & \times & \ast
\end{pmatrix}$$

and if we continue by constructing the second one, we reach the following matrix:

$$\mathbf{D}=\begin{pmatrix}
\ast & 1 & 2 & \times & 1 & \circ & \circ & & & \times & \times & \times & \times \\
1 & \ast & \times & \times & 1 & \times & & & & & \times & \times & \times \\
2 & \times & \ast & \times & \times & \times & \times & & & & & \times & \times \\
\times & \times & \times & \ast & \times & \times & \times & \times & & & & & \times \\
1 & 1 & \times & \times & \ast & \times & \times & \times & \times & & & & \\
 & \times & \times & \times & \times & \ast & \times & \times & \times & \times & & & \\
 & & \times & \times & \times & \times & \ast & \times & \times & \times & \times & & \\
 & & & \times & \times & \times & \times & \ast & \times & \times & \times & \times & \\
 & & & & \times & \times & \times & \times & \ast & \times & \times & \times & \times \\
\times & & & & & \times & \times & \times & \times & \ast & \times & \times & \times \\
\times & \times & & & & & \times & \times & \times & \times & \ast & \times & \times \\
\times & \times & \times & & & & & \times & \times & \times & \times & \ast & \times \\
\times & \times & \times & \times & & & & & \times & \times & \times & \times & \ast
\end{pmatrix}$$

which has enough empty positions to continue constructing the desired number of codewords. Finally, we obtain the matrix

$$\mathbf{D}=\begin{pmatrix}
\ast & 1 & 2 & \times & 1 & \times & \times & 2 & \times & \times & \times & \times & \times \\
1 & \ast & \times & \times & 1 & \times & \times & \times & \times & \times & \times & \times & \times \\
2 & \times & \ast & \times & \times & \times & \times & 2 & \times & \times & \times & \times & \times \\
\times & \times & \times & \ast & \times & \times & \times & \times & \times & \times & \times & \times & \times \\
1 & 1 & \times & \times & \ast & \times & \times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \ast & \times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \ast & \times & \times & \times & \times & \times & \times \\
2 & \times & 2 & \times & \times & \times & \times & \ast & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \ast & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & \ast & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \ast & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \ast & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \ast
\end{pmatrix}.$$

Therefore, the desired code matrices are

$$\mathbf{D}_{\mathbf{xx}}^{1}=\begin{pmatrix}1&1&0&0&1&0&0&0&0&0&0&0&0\\1&1&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\1&1&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\end{pmatrix}$$

$$\mathbf{D}_{\mathbf{xx}}^{2}=\begin{pmatrix}1&0&1&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\1&0&1&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\1&0&1&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0\end{pmatrix}.$$

This means that the two code vectors are $\mathbf{x}_1 = (1100100000000)$ and $\mathbf{x}_2 = (1010000100000)$. One can easily observe that these codewords meet the auto- and cross-correlation constraints.

For faster convergence, it is also useful to select the place of the new elements in the first row randomly, and not to select just the first empty position after the last inserted element. If we decide to select it randomly, then we should note that the positions exactly between the two previously inserted "1" elements of one code, in the first row of $\mathbf{D}$, is also forbidden, because $\lambda_a = \lambda_c = 1$. Therefore, if the number of positions between "1" elements of one code is odd, then we fill the position which is exactly in the middle with $\circ$.

The pseudocode for the complete proposed algorithm is given in the Appendix. The parameters of some codes constructed in this way are listed in Table I, and the number of codewords of each family is compared with the number of codewords obtained by the accelerated greedy algorithm [3], which randomly searches for codewords in the domain of (0, 1) vectors of length n and Hamming weight w. We see that for larger size codes, with matrix-based search, we can find more codewords relative to the accelerated greedy algorithm. Fig. 3 also compares the number of obtained codewords in terms of code length for various code weights.

If we compare the two algorithms from the viewpoint of computational complexity, it is evident that matrix search needs more computations in each step of algorithm; however, the time needed in both algorithms is almost the same. This is because the greedy algorithm is a full random search among all possible vectors, but the new algorithm has a strategy to eliminate many possible matrices from one step to the next, which reduces the time needed to reach the desired number of codewords.

TABLE I
COMPARISON OF TWO ALGORITHMS

Code weight	Code length	Johnson upper bound	Outer-product matrix based algorithm	Inner-product based accelerated greedy algorithm
3	61	10	10	10
3	301	50	50	50
3	365	60	60	60
3	601	100	99	98
3	901	150	148	147
4	121	10	10	10
4	247	20	20	19
4	481	40	38	37
4	841	70	67	66
4	5000	416	403	388
5	91	4	4	4
5	400	19	18	17
5	800	39	36	35
5	5000	249	236	220
5	10000	499	475	440
6	300	9	8	8
6	1000	33	29	27
6	10000	333	305	270
7	1000	23	19	18
7	5000	119	103	90
7	10000	238	206	170
8	1000	17	13	12
8	5000	89	73	60
9	5000	69	52	40
9	10000	138	110	90
10	91	1	1	1
10	10000	111	80	70

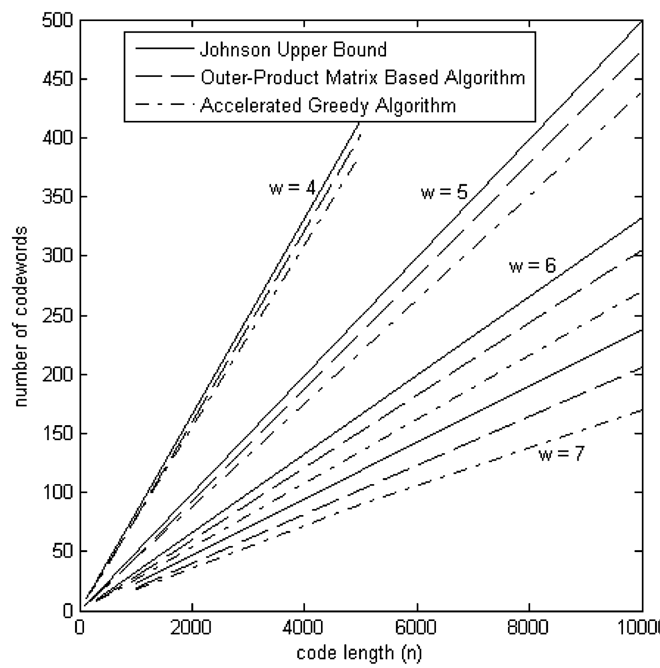

Fig. 3. Comparison of two algorithms.

V. CONCLUSION

In this letter, we introduced a novel and new representation of OOCs, namely, outer-product matrix representation, and investigated some of their properties and discussed a new definition of OOCs based on these properties. Furthermore, we presented a new search algorithm based on outer-product matrices, and we showed that it is more efficient when compared with the accelerated greedy algorithm in terms of number of obtained codewords, especially when the cardinality of the OOC is large.

APPENDIX

In this appendix, we give pseudocode that just clarifies all the steps of outer-product matrix-based search algorithm.

```
// md: total number of desired codewords
// wd: code weight
// n: code length
// Forbidden positions because of cross-correlation
   constraints are filled with md + 1
// Forbidden positions because of auto-correlation
   constraints are filled with md + 2
reset all variables with zero
for i = 0 to n, D(i,i) = md + 1
if mod(n,2) = 0 then for i = 0 to n − 1,
D(mod(n/2 + i, n), i) = md + 1
D1(s,:,:) = D
while(m! = md)
    m = m + 1
    w = 1
    replace md + 2 with 0 in D
    while(w! = wd)
        D1(s,:,:) = D
        s = s + 1
        cnt=number of empty positions in D(0,:)
        if cnt < wd − w then
            while(cnt < wd − w)
                s = s − 1
                if w > 1 then w = w − 1 else
                (w = wd − 1 and m = m − 1)
                D1(s − 1, 0, x(m,w)) = md + 2
                cnt=number of empty positions in
                D1(s − 1, 0, x(m,w)))
            D = D1(s − 1,:,:)
        t=random number between 0 and n − 1 where
        D(0,:) = 0
        D(0,t) = m
        x(m,w) = t
        w = w + 1
        fill all other places in tth diagonal with md + 1
        for j = 1 to w − 2
            y = x(m,j)
            D(y,t) = m and D(t,y) = m
            Fill all other places in |t − y|th and
            n − |t − y|th diagonal with md + 1
        for j = 0 to w − 2
            y = x(m,j)
            if mod(t − y, 2) = D(0,(t + y)/2) = 0 then
            D(0, mod((y + t)/2, n) = md + 2
            if mod(n + y − t, 2) = D(0, mod((n +
            y + t)/2, n)) = 0 then D(0, mod((n + y +
            t)/2, n) = md + 2
            y = x(m, j + 1)
            for i = 1 to n − 1
                if D(y,i) = m + 1 and D(0,i) = 0
                then D(0,i) = m + 2
// At this time the numbers in x represent chip positions
   of md codewords
```

REFERENCES

[1] J. A. Salehi, "Code-division multiple-access techniques in optical fiber networks—Part I: Fundamental principles," *IEEE Trans. Commun.*, vol. 37, no. 8, pp. 824–833, Aug. 1989.

[2] J. A. Salehi and C. A. Brackett, "Code-division multiple-access techniques in optical fiber networks—Part II: Systems performance analysis," *IEEE Trans. Commun.*, vol. 37, no. 8, pp. 834–842, Aug. 1989.

[3] F. R. K. Chung, J. A. Salehi, and V. K. Wei, "Optical orthogonal codes: Design, analysis, and applications," *IEEE Trans. Inf. Theory*, vol. 35, no. 3, pp. 595–604, May 1989.

[4] G. C. Yang and T. E. Fuja, "Optical orthogonal codes with unequal auto- and cross-correlations," *IEEE Trans. Inf. Theory*, vol. 41, no. 1, pp. 96–106, Jan. 1995.

[5] H. Chung and P. V. Kumar, "Optical orthogonal codes—New bounds and an optimal construction," *IEEE Trans. Inf. Theory*, vol. 36, no. 4, pp. 866–873, Jul. 1990.

[6] R. Fuji-Hara and Y. Miao, "Optical orthogonal codes: Their bounds and new optimal constructions," *IEEE Trans. Inf. Theory*, vol. 46, no. 7, pp. 2396–2406, Nov. 2000.

[7] G. C. Yang and W. C. Kwong, *Prime Codes With Applications to CDMA Optical and Wireless Networks*. Norwood, MA: Artech House, 2002.

[8] C. J. Colbourn, J. H. Dinitz, and D. R. Stinson, "Applications of combinatorial designs to communications, cryptography, and networking," in *Surveys in Combinatorics*, ser. London Math. Soc. Lecture Note Ser. 187, K. Walker, Ed. Cambridge, U.K.: Cambridge Univ. Press, 1993.

[9] J. Yin, "Some combinatorial constructions for optical orthogonal codes," *Discr. Math.*, vol. 185, pp. 201–219, 1998.

[10] R. Fuji-Hara, Y. Miao, and J. Yin, "Optimal (9v,4,1) optical orthogonal codes," *SIAM J. Discr. Math.*, vol. 14, no. 2, pp. 256–266, 2001.

[11] Y. Chang and Y. Miao, "Constructions for optimal optical orthogonal codes," *Discr. Math.*, vol. 261, pp. 127–139, 2003.

[12] Y. Chang, R. Fuji-Hara, and Y. Miao, "Combinatorial construction of optimal optical orthogonal codes with weight 4," *IEEE Trans. Inf. Theory*, vol. 49, no. 5, pp. 1283–1292, May 2003.

[13] F. E. Hohn, *Elementary Matrix Algebra*. New York: Macmillan, 1973.