

## An efficient algorithm for the identification of isomorphic orthogonal arrays

H. Evangelaras

C. Koukouvinos\*

E. Lappas

*Department of Mathematics*

*National Technical University of Athens*

*Zografou 15773*

*Athens*

*Greece*

---

### Abstract

In this paper we propose an efficient algorithm for discarding a large number of isomorphic two level orthogonal arrays. The proposed algorithm can also be used with any two-level designs and can easily be extended to cover multi level designs.

---

*Keywords* : *Orthogonal arrays, isomorphism, complexity, efficiency.*

### 1. Introduction

An *orthogonal array*  $OA(n, q, s, t)$  is an  $n \times q$  array with entries from a set of  $s$  distinct symbols arranged so that, for any collection of  $t$  columns of the array, each of the  $s^t$  row vectors appears equally often. Thus we see that  $s^t$  divides  $n$ . We call  $n$  the number of runs in the orthogonal array,  $q$  the number of columns,  $s$  the number of levels in each column and  $t$  the strength of the array.

Orthogonal arrays are useful in various fields such as, Design of Experiments, Coding Theory, etc. For more details on the use of

---

\*E-mail: ckoukou@math.ntua.gr

---

*Journal of Discrete Mathematical Sciences & Cryptography*

Vol. 9 (2006), No. 1, pp. 125–132

© Taru Publications

orthogonal arrays, we refer the interested reader to Hedayat, Sloane and Stufken [3].

One can obtain a wide list of orthogonal arrays with specific parameters  $n, q$  and  $s$ , which can be classified into isomorphic classes with respect to the following definition.

We use the terms “factor” and “column” alternatively in the whole paper.

**Definition 1 (Isomorphism (IOA)).** Two orthogonal arrays are said to be isomorphic if, one can be obtained from the other by a sequence of permutations of the columns, the rows and the levels of each factor.

The identification of the complete non isomorphic classes of orthogonal arrays with specific parameters  $n, q$  and  $s$ , is a difficult combinatorial problem which seems to have important impact in various fields of Discrete Mathematics. For example, two non isomorphic orthogonal arrays may behave differently in certain experimental situations. Isomorphism checking is an NP hard problem as  $n, q$  and  $s$  increases. The complexity of an algorithm based on the definition of isomorphism is  $n!q!(s!)^q$ . Therefore, it is practically impossible to check for isomorphic classes of a wide list of orthogonal arrays, using an algorithm based on the definition (see also [1, 2, 4]).

In this paper we are interested in two level orthogonal arrays,  $OA(n, q, 2, t)$ . We propose an efficient algorithm for identifying isomorphic orthogonal arrays. Our method quickly discards isomorphic orthogonal arrays, leaving a small remainder which can easily be checked using the definition, in a reasonable CPU time. The method works fine with any two-level designs as well. It can also be extended to cover multi-level designs. However, we have focused on the isomorphism of two-level orthogonal arrays since their use in various fields of Applied Statistics, such as Design of Experiments, off-line Quality Control etc., receives great attention in the literature as well as in practice. The knowledge of the full list of non isomorphic orthogonal arrays with specific parameters offers great flexibility to researchers and practitioners with regard to certain experimental situations.

## 2. Basic definitions

We state some basic definitions and notations that are used in this paper.

**Definition 2.** Let  $A$  be an  $\text{OA}(n, q, 2, t)$ . With  $\mathcal{A}(A)$  we denote the set of matrices that are produced from  $A$  by all possible permutations of columns, rows and levels of each factor. Clearly  $|\mathcal{A}(A)| = n!q!2^q$ .

Using the notations given in Definition 2, we give an alternative definition of isomorphism.

**Definition 3 (Alternative definition of IOA).** Two  $\text{OA}(n, q, 2, t)$ ,  $A$  and  $B$ , are isomorphic if  $A \in \mathcal{A}(B)$  or equivalently  $B \in \mathcal{A}(A)$ .

We now define the Max-int-row form of an  $\text{OA}(n, q, 2, t)$ .

**Definition 4 (Max-int-row form of an OA).** The Max-int-row form of an  $\text{OA}(n, q, 2, t)$   $A$ , can be obtained using this algorithm  
repeat

- $B := A$
- sort the rows of  $A$
- sort the columns of  $A$

until  $(B = A)$ .

Similarly, we obtain the Max-int-col form of an  $\text{OA}$ , as it is described in the following definition.

**Definition 5 (Max-int-col form of an OA).** The Max-int-col form of an  $\text{OA}(n, q, 2, t)$   $A$ , can be obtained using this algorithm  
repeat

- $B := A$
- sort the columns of  $A$
- sort the rows of  $A$

until  $(B = A)$ .

**Remark 1.** The Max-int-col and the max-int-row of the same  $\text{OA}(n, q, 2, t)$   $A$ , are not necessarily equal as it can be verified in the following example.

**Example 1.** Let  $\mathbf{A}$  be an  $\text{OA}(12, 6, 2, t)$ . With  $\mathbf{A}_r$  we denote the max-int-row form of  $\mathbf{A}$ , and with  $\mathbf{A}_c$  we denote the max-int-col form of  $\mathbf{A}$ , where  $-$  stands for  $-1$ .

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & - \\ 1 & 1 & - & - & - & 1 \\ 1 & - & 1 & - & - & 1 \\ 1 & - & - & 1 & - & - \\ 1 & - & - & - & 1 & - \\ - & 1 & 1 & - & - & - \\ - & 1 & - & 1 & - & 1 \\ - & 1 & - & - & 1 & - \\ - & - & 1 & 1 & - & - \\ - & - & 1 & - & 1 & 1 \\ - & - & - & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{A}_r = \begin{bmatrix} - & - & - & 1 & 1 & 1 \\ - & - & 1 & - & 1 & 1 \\ - & - & 1 & 1 & - & - \\ - & 1 & - & - & - & 1 \\ - & 1 & - & 1 & 1 & - \\ - & 1 & 1 & - & - & - \\ 1 & - & - & - & - & 1 \\ 1 & - & - & 1 & - & - \\ 1 & - & - & - & 1 & - \\ 1 & 1 & - & - & 1 & - \\ 1 & 1 & 1 & 1 & - & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

$$\mathbf{A}_c = \begin{bmatrix} - & - & - & - & 1 & 1 \\ - & - & - & 1 & - & 1 \\ - & - & 1 & 1 & - & - \\ - & 1 & - & - & 1 & - \\ - & 1 & 1 & - & - & - \\ - & 1 & 1 & 1 & 1 & 1 \\ 1 & - & - & 1 & 1 & - \\ 1 & - & 1 & - & - & 1 \\ 1 & - & 1 & - & 1 & - \\ 1 & 1 & - & - & - & 1 \\ 1 & 1 & - & 1 & - & - \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

### 3. The Max-int algorithm

We need to introduce some specific sets of matrices that are produced from a given orthogonal array.

**Definition 6.** Let  $A$  be an  $\text{OA}(n, q, 2, t)$ .  $\mathcal{M}_r(A)$  (or  $\mathcal{M}_c(A)$ ) denotes the set of all maxint-row (or max-int-col) matrices that are produced from  $A$  by all possible permutations of the levels of each factor.

**Remark 2.**  $|\mathcal{M}_r(A)| = |\mathcal{M}_c(A)| = 2^q$ .

The set of all matrices produced from max-int-row and max-int-col procedures when applied to the orthogonal array  $A$ , is called *max-int set* and is denoted as  $\mathcal{M}(A)$ . Clearly,  $\mathcal{M}(A) = \mathcal{M}_r(A) \cup \mathcal{M}_c(A)$ .

**Proposition 1.** *For a given orthogonal array  $A$ , the max-int set of  $A$  is a subset of  $\mathcal{A}(A)$ . In other words,  $\mathcal{M}(A) \subset \mathcal{A}(A)$ .*

The following proposition is useful for checking the isomorphism of two orthogonal arrays.

**Proposition 2.** *Let  $A, B$  be two  $\text{OA}(n, q, 2, t)$ . If there exists  $C \in \mathcal{M}(A)$  (or  $\mathcal{M}_c(A)$  or  $\mathcal{M}_r(A)$ ) and  $D \in \mathcal{M}(B)$  (or  $\mathcal{M}_c(A)$  or  $\mathcal{M}_r(A)$ ) such that  $C = D$  then  $A$  and  $B$  are isomorphic.*

**Remark 3.** The inverse does not hold in general.

#### 4. Implementation

Suppose that  $A_1, A_2, \dots, A_N$  are  $N$   $\text{OA}(n, q, 2, t)$ , for which is needed to check which are not isomorphic.

##### 4.1 Typical algorithm

For simplicity in the algorithm presented below we denote with  $\mathcal{N}(A)$  one of  $\mathcal{A}(A)$  or  $\mathcal{M}_r(A)$  or  $\mathcal{M}_c(A)$  or  $\mathcal{M}(A)$ . A typical algorithm that would be used throughout this paper is:

- for  $i$  from 1 to  $N$  do create all  $\mathcal{N}(A_i)$
- Set  $\mathcal{F} := \{A_1\}$  and  $f := 1$
- for  $i$  from 2 to  $N$  do
  - for  $j$  from 1 to  $f$  do
    - Check if there exists a pair  $(i, j)$ 
      - for which  $C \in \mathcal{N}(A_i)$  and  $D \in \mathcal{N}(A_j)$  such that  $C = D$ 
        - If there exists at least one such pair then  $A_i$  and  $A_j$  are IOA
        - Else add  $A_i$  in  $\mathcal{F}$  and  $f := f + 1$
- The set  $\mathcal{F}$  contains the OA that are or might not be isomorphic depending on the choice of  $\mathcal{N}$ .

##### 4.2 Complexity

Throughout this paper, we will consider the quick sort algorithm with complexity  $n \log_2 n$  when we sort any  $n$  elements. We need to study and draw conclusions for three separate cases.

**Case I.** *The complexity of the isomorphism checking.* Using the definition, we need to generate for each OA  $n!q!2^q$  isomorphic arrays to compare it with others. For example, from an OA with parameters  $(12, 6, 2, t)$ , the number of the isomorphic OA it generates is approximately  $2.2072 \cdot 10^{13}$ .

**Case II.** *The complexity of max-int-row (or max-int-col)  $\mathcal{M}_r$  or  $\mathcal{M}_c$ .* For each OA we need to generate only  $2^q$  isomorphic arrays. Then, each of them should be sorted by row and columns according to the definitions 4, 5. Through specific case studies we have done in some OAs, with orders  $n \leq 40$  and  $q \leq 11$ , we concluded that we do not have to make more than 5 loops in the max-int-row (or max-int-col) definition. That means we need at most 5 sorts of  $n$  elements along the rows and at most 5 sorts of  $q$  elements along the columns of an OA with parameters  $(n, q, 2, t)$ . So, the complexity is estimated to be  $25 \cdot 2^q nq \log_2(nq)$ . Continuing our example using an OA with parameters  $(12, 6, 2, t)$  the complexity of this proposed algorithm is approximately  $7.1078 \cdot 10^5$  which is much smaller than the complexity of the definition.

**Case III.** *The complexity of the max-int set  $\mathcal{M}$ .* In this last case for each OA we need to generate only  $2^{q+1}$  isomorphic arrays. Then, each of them should be sorted by row and columns according to the definition of the max-int set. Similarly to the Case II, the complexity is estimated to be  $25 \cdot 2^{q+1} nq \cdot \log_2(nq)$ . That is, for the OA with parameters  $(12, 6, 2, t)$  the complexity is approximately  $1.4216 \cdot 10^6$ .

### 4.3 Efficiency

In the implementation of the proposed algorithm presented in 4.1, if we use one of the  $\mathcal{M}$ ,  $\mathcal{M}_c$  or  $\mathcal{M}_r$ , it is obvious that the returned set  $\mathcal{F}$  will not contain only nonisomorphic OAs. We will now compare the set  $\mathcal{F}$  produced when we deliberately use each of the  $\mathcal{M}$ ,  $\mathcal{M}_c$  or  $\mathcal{M}_r$  sets, with the one produced using the  $\mathcal{A}$ . The efficiency is defined to be:

$$\text{eff} = \frac{\text{number OA that are cut off by one of the } \mathcal{M}, \mathcal{M}_c \text{ or } \mathcal{M}_r}{\text{number of OA that are cut off by the definition}}$$

#### 4.3.1 Case I

We took an  $\text{OA}(24, 6, 2, t)$  and generated isomorphic OAs using all possible permutations of the columns and of the levels of each factor, that is  $6!2^6 = 46080$  isomorphic OAs. We applied the algorithm presented

in 4.1 with the use of  $\mathcal{M}$ ,  $\mathcal{M}_c$ ,  $\mathcal{M}_r$  and  $\mathcal{A}$  respectively. In Table 1, the column  $\mathcal{F}$  is the number of OAs returned by each method.

**Table 1**  
Efficiency for OA(24, 6, 2,  $t$ )

Set used	$\mathcal{F}$ -returned	eff
$\mathcal{A}$	1	100%
$\mathcal{M}_r$	58	99.8763%
$\mathcal{M}_c$	30	99.9371%
$\mathcal{M}$	11	99.98%

We report that the higher efficiency is obtained using the  $\mathcal{M}$  set, leaving aside the optimal use of the definition of isomorphism. This fascinating conclusion came up in every case study we tested the algorithm.

It is obvious that the  $\mathcal{M}$ ,  $\mathcal{M}_c$  or  $\mathcal{M}_r$  can be used to quickly reduce a vast number of OAs leaving a small remainder that can be further checked using the definition of isomorphism. We further need to check the reduction in the CPU time using these methods.

#### 4.3.2 Case II

We constructed a random sample of 10000 OA(16, 6, 2,  $t$ ) and we applied four different schemes in order to find the set of non-isomorphic OAs. More precisely:

- (i) We used the definition of isomorphism
- (ii) We used the  $\mathcal{M}_r$  set and then the definition of isomorphism in the remainder obtained.
- (iii) We used the  $\mathcal{M}_c$  set and then the definition of isomorphism in the remainder obtained.
- (iv) We used the  $\mathcal{M}$  set and then the definition of isomorphism in the remainder obtained. The interesting results are summarized in the Table 2 where with 1 time unit we declare the time needed for the complete search for non-isomorphic OAs using the definition. The column "remainder" shows the number of OAs needed to be tested with the definition, after the first compilation. The column "extra time" declares the time needed to check only the remainder using

the definition, while the final column “total time” shows the time needed for the complete compilation of every method.

**Table 2**  
CPU time

Scheme	time (unit)	remainder	extra time (unit)	non-isomorphic	total time (unit)
(i)	1	–	–	27	1
(ii)	0.01	335	0.032	27	0.042
(iii)	0.002	50	0.0055	27	0.0075
(iv)	0.003	39	0.004	27	0.007

The most efficient method seems to be the scheme (iv) while it does not appear to have much difference from scheme (iii). However, as the sample of the OAs increases scheme (iv) proves its superiority as, in spite of its slightly greater complexity, it always leaves a smaller remainder to be further checked with the most time consuming algorithm based on the definition.

#### References

- [1] J. B. Clark and A. M. Dean, Equivalence of fractional factorial designs, *Statistica Sinica*, Vol. 11 (2001), pp. 537–547.
- [2] K.-T. Fang and G. Ge, A sensitive algorithm for detecting the inequivalence of Hadamard matrices, *Mathematics of Computation*, Vol. 73 (2004), pp. 843–851.
- [3] A. S. Hedayat, N. J. A. Sloane and J. Stufken, *Orthogonal Arrays: Theory and Applications*, Springer-Verlag, New York, 1999.
- [4] C.-X. Ma, K.-T. Fang and D. K. J. Lin, On the isomorphism of fractional factorial designs, *Journal of Complexity*, Vol. 17 (2001), pp. 86–97.

*Received February, 2005*