Experimental Analysis of Classification Algorithm for Orthogonal Arrays

Maya Hristova[1], Iliya Bouyukliev[2]

[1]Faculty of Mathematics and Informatics,
Veliko Tarnovo University

[2]Institute of Mathematics and Informatics,
Bulgarian Academy of Sciences

This research is supported by Bulgarian Science Fund under Contract DN 02/2,13.12.2016 "Codes and combinatorial configurations"

Table of Contents

Orthogonal arrays

Let S be a set of s symbols (levels). We will denote these levels by $0, 1, \cdots, s-1$.

Orthogonal array

An $N \times k$ array A with entries from S is said to be *an orthogonal array with s levels, strength t ($0 \leq t \leq k$) and index λ* if every $N \times t$ subarray of A contains each t-tuple based on S exactly λ as a rows. We will denote such orthogonal array by $OA(N, k, s, t)$.

The parameters N, t, s and k have various names depending on the application of the given orthogonal array.

History

- In 1940's C. R. Rao gave a general definitions of orthogonal arrays and their applications.
- Chakravarti (1956, 1961, 1963) introduced and studied partially balanced arrays in his Ph.D. dissertation.
- In 1980's Taguchi developed methods for industrial experimentation using orthogonal designs.
- Boyvalenkov, Stoyanova and Marinova investigated nonexistence of few binary orthogonal arrays (2015, 2016, 2017).

Application of OAs

- Designing experiments.
- Combinatorics.
- Finite fields.
- Error-correcting codes.

Theorem

If C is a $(k, N, d)_s$ linear code over $GF(s)$ with dual distance $d^\perp$ then the codewords of C form the rows of an $OA(N, k, s, d^\perp - 1)$ with entries from $GF(s)$. Conversely, the rows of a linear $OA(N, k, s, t)$ over $GF(s)$ form a $(k, N, d)_s$ linear code over $GF(s)$ with dual distance $d^\perp \geq t + 1$. If the orthogonal array has strength t but not $t + 1$, $d^\perp$ is precisely $t + 1$.

Isomorhism of orthogonal arrays

Classification problem

The classification problem is to determine exactly one element from every equivalence class.

Isomorhism of orthogonal arrays

Two arrays are said to be isomorphic if one array can be obtained by permuting rows, columns, or factor levels of the other array.

Known methods for constructing orthogonal arrays

Known methods

- To generate all orthogonal arrays (then to perform exhaustive isomorphism test among them).
- To use isomorph free generation via canonical form (total lexicographical order).

- Bulutoglu and Margot propose technique that implements linear programming.
- Schoen, Eendebak and Nguyen describe an algorithm for generating orthogonal arrays with isomorph rejection using a lexicographic order of the arrays.

Canonical augmentation

Main tasks

- Efficient constructing of orthogonal arrays
- Quick structure test
- Generation of only non-isomorphic objects

Canonical augmentation

Juxtaposition of binary matrices to orthogonal arrays

- To every element $s_i \in s$ we juxtapose vector with length S, $0 \mapsto (10\cdots0), 1 \mapsto (010\cdots0), \cdots, (s-1) \mapsto (0\cdots01)$.
- Extend the matrix with n rows, which mark the columns representing one coordinate.
- We consider extended binary matrices EM.

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 2 & 2 \\ 1 & 0 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 2 \\ 2 & 1 & 0 \\ 2 & 2 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ . & . & . & . & . & . & . & . & . \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Canonical augmentation

Two orthogonal arrays are isomorphic iff their corresponding binary matrices are isomorphic.

Isomorphism of binary matrices

Two binary matrices A and B of the same size are isomorphic if one can be obtained from the other by permutations of rows and columns.

Canonical augmentation

Let denote by Ω the search space.

Canonical representative map

A canonical representative map is called the function $\rho : \Omega \to \Omega$, which has the properties: for every $X \in \Omega : \rho(X) \cong X$; for every $X, Y \in \Omega$ from $X \cong Y$ follows that $\rho(X) \equiv \rho(Y)$.

The canonical form of an object X is its image according to the canonical representative $\rho(X)$. The object X is in canonical form if $\rho(X) = X$.

Canonical augmentation

- Any permutation of columns that represents the rows of matrix B in rows of the same matrix is called automorphism.
- The set of all automorphisms of B forms a group called automorphism group (denoted with $Aut(B)$).
- The $Aut(B)$ group splits the set of columns of B into disjoint subsets $O_1, O_2, \cdots, O_k$ called orbits.

- Ordering of the orbits of X (preceding).
- Special orbit (the last one).

Canonical augmentation

Invariant

An invariant of the column a of EM is a function $f : f(a) \in \mathbb{Z}$ such that if the columns a and b are in the same orbit then $f(a) = f(b)$.

- Pseudo-orbits: a set of one or more orbits with the same value of f.
- Special pseudo-orbit (the last one).

Invariant

- Invariants of the rows: based on the distances between every two rows of EM.
- Invariants of the columns: based on the position of ones in each column and already obtained invariants of the rows.

Parent test

The last added column passes the parent test if it is in the smallest last pseudo-orbit.

Algorithm

Algorithm 1 CanonicalAugmentationOA

```
1:  procedure CanonicalAugmentationOA(X: object)
2:      if X has k columns then Print(X);
3:      else
4:          M ← ∅;
5:          foreach Z ∈ C(X) do
6:              if pass the parent test then
7:                  M ← M ∪ Z
8:          remove isomorphic objects from M;
9:          foreach Z ∈ M do
10:             CanonicalAugmentationOA(Z)
```

Results

Table: A: Partial times for generating $OA(40, k, 2, 3)$.

k	Generating	Struct. test	Inv.	Can. form	Equiv. test
11	18.45 sec.	0.621 sec.	3.584 sec.	342 sec.	0.212 sec.
12	18.739 sec.	0.867 sec.	4.858 sec.	434 sec.	0.256 sec.
13	20.485 sec.	1.12 sec.	5.62 sec.	643 sec.	0.211 sec.
14	25.323 sec.	1.139 sec.	6.021 sec.	801 sec.	0.297 sec.
15	27.372 sec.	1.197 sec.	6.397 sec.	868 sec.	0.222 sec.
16	28.448 sec.	1.267 sec.	6.502 sec.	902 sec.	0.335 sec.
17	30.604 sec.	1.367 sec.	7.118 sec.	973 sec.	0.399 sec.
18	26.759 sec.	1.222 sec.	6.912 sec.	954 sec.	0.183 sec.
19	31.494 sec.	1.378 sec.	7.221 sec.	1003 sec.	0.299 sec.
20	31.485 sec.	1.411 sec.	7.453 sec.	1003 sec.	0.296 sec.

Results

Table: B

k	# OA INV	# OA CF	# OA PT	# OA NI	Total time
11	239384	12907	3880	206	369 sec.
12	296042	15096	4462	235	468 sec.
13	345592	17330	4870	132	682 sec.
14	372044	18798	5094	96	844 sec.
15	389710	19235	5248	36	912 sec.
16	395608	19475	5318	26	948 sec.
17	399246	19683	5352	7	1020 sec.
18	400024	19725	5374	6	998 sec.
19	400500	19749	5386	3	1049 sec.
20	400622	19755	5392	3	1049 sec.

Number of $OA(40, k, 2, 3)$ for which the invariant is calculated, canonical form is obtained, parent test is passed. The last two columns contain the total number of non-isomorphic orthogonal arrays and the total time for generating.

Results

Table: A: Partial times for generating $OA(64, k, 2, 3)$.

k	Generating	Struct. test	Inv.	Can. form	Equiv. test
4	0 sec.	0 sec.	0 sec.	0 sec.	0 sec.
5	0.084 sec.	0.016 sec.	0.022 sec.	0.1 sec.	0.016 sec.
6	39.622 sec.	1.721 sec.	7.493 sec.	116 sec.	1.216 sec.
7	14642 sec.	396 sec.	2436 sec.	223018 sec.	149 sec.

Table: B

k	# OA INV	# OA CF	# OA PT	# OA NI	Total time
4	29	10	10	5	0.06 sec.
5	2037	289	243	19	0.40 sec.
6	438220	25970	12835	358	180.10 sec.
7	109601830	7117448	904671	91789	244044.99 sec.

Number of $OA(64, k, 2, 3)$ for which the invariant is calculated, canonical form is obtained, parent test is passed. The last two columns contain the total number of non-isomorphic OAs and the total time for generating.

Results

Table: A: Partial times for generating $OA(128, k, 2, 4)$.

k	Generating	Struct. test	Inv.	Can. form	Equiv. test
5	0 sec.	0 sec.	0 sec.	0 sec.	0 sec.
6	10.044 sec.	0.028 sec.	0.032 sec.	0.15 sec.	0.11 sec.
7	24077 sec.	1.35 sec.	34.46 sec.	102.527 sec.	164.161 sec.

Table: B

k	# OA INV	# OA CF	# OA PT	# OA NI	Total time
5	73	11	11	5	0.1 sec.
6	4644	262	250	17	11.16 sec.
7	1831130	34007	12230	123	26752.39 sec.

*The results are obtained on computer with CPU Intel(R) Core(TM) i7, 4 Core.

All binary orthogonal arrays from [http://www.pietereendebak.nl/oapackage/series.html] can be obtained with our algorithm.

New result

With this algorithm, we obtain new classification result for **OA(64, 8, 2, 3)**.
The number of these arrays are **71 186 462** and they are calculated for about 4 days.
We use a computer with two CPU Intel(R) Xeon(R) E5, 6 Core.

Conclusion

Negligible amount of time

- Structure test
- Calculation of invariant

Computationally expensive.

- Generation (only for orthogonal arrays with relatively many rows)
- Calculation of canonical form

Future Work

- Generation of orthogonal arrays from already known part of the arrays.
- Extension of the alphabet.
- Using some already known information about the spectrum of the corresponding arrays.

References


I. Bouyukliev, (2000)
'Q - EXTENSION' – strategy in algorithms, in
Proceedings of the International Workshop ACCT, Bansko, Bulgaria, 2000, 84-89.


I. Bouyukliev, M. Hristova,
A Method for Construction of Orthogonal Arrays,
Eighth International Workshop on Optimal Codes and Related Topics, July 10-14, 2017, Sofia, Bulgaria.


I. Bouyukliev, M. Hristova,
About an Approach for Constructing Combinatorial Objects,
Cybernetics and Information Technologies, Volume 18, No 5.


P. Boyvalenkov, T. Marinova, M. Stoyanova,
Nonexistence of a few binary orthogonal arrays,
Discrete Applied Mathematics, 217(2), 2017, pp. 144-150.


D.A. Bulutoglu and F. Margot,
Classification of orthogonal arrays by integer programming, in
Journal of Statistical Planning and Inference, 138:654-666, 2008.


J. Chen, D.X. Sun, and C.F.J. Wu,
A catalogue of two-level and three level fractional factorial designs with small runs,
International Statistical Review, 61:131-145, 1993.