A Post-Processing Algorithm Applied to Reduce the Size of the Covering Arrays of the NIST Repository

Ph.D. Jose Torres-Jimenez

Laboratorio de Tecnologías de Información, CINVESTAV-Tamaulipas km 5.5
Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., México.

September 19, 2013

Contents

Table of Contents

Introduction
Software Testing

- Nowadays software products are present in almost all human activities, then it is necessary to assure a high level of functionality of the software products.
- It is estimated that at least 50% of the cost of developing a new software component is related to the testing process. [1]

[1] B. Hailpern y P. Santhanam, 2002 [1]. A. Hartman, 2005 [2].

- One option to test a software component is to use an exhaustive approach, i.e. test all the possible combinations of the input parameters.
- Another option is to use combinatorial testing, that guarentees that all the combinations of certain number of parameters is tested exactly or at least certain number of times. [2]

[2] Cohen *et al.* [3] y Cohen *et al.* [4].

Table of Contents

Definition

Let be N, t, k, v, and λ five positive integers, an orthogonal array $\mathrm{OA}_\lambda(N; t, k, v)$ is an $N \times k$ array $A = (a_{i,j})$, $0 \leq i \leq N-1$, $0 \leq j \leq k-1$, over $\mathbb{Z}_v = \{0, 1, \ldots, v-1\}$ with the property that for any t distinct columns, there are exactly λ rows that takes each value of $\mathbb{Z}_v^t = \{0, 1, \ldots, v-1\}^t$.

- When $\lambda = 1$ the OA is of *unitary index*.
- For OAs of unitary index and v a primepower there exists an optimal solution [3]

[3] Bush [5]

0	0	0	0	0	0	0
1	0	0	1	1	0	1
0	1	0	1	0	1	1
0	0	1	0	1	1	1
1	0	1	1	0	1	0
1	1	0	0	1	1	0
0	1	1	1	1	0	0
1	1	1	0	0	0	1

Figure 1: The orthogonal array $\mathsf{OA}_2(8; 2, 7, 2)$.

- The case when v is a prime power and $v \geq t$, is solved using arithmetic of a Galois Finite Field. The value of each cell of the OA is the evaluation of the number of row in the number of column, the last column is directly the leftmost coefficient of the number of row. This procedure can be implemented using a logaritm table for a Galois Finite Field [4]
- The case when v is a prime power and $v < t$ is solved using the zerosum algorithm.

[4] Torres et al. [6] and Torres et al. [7]

The OAs are a good option to test efficiently software components, but they have the drawback that they do not exist for certain combinations of parameters (λ, k, v, t). A relaxed version of the OAs are the CAs (each combination must exist at leas once). Given the relaxation, they exist for any combination of parameters and they have a lower number of rows.

Definition

Let be N, t, k, and v four positive integers, a covering array CA$(N; t, k, v)$[5] is an $N \times k$ array $A = (a_{i,j})$, $0 \leq i \leq N-1$, $0 \leq j \leq k-1$, over $\mathbb{Z}_v = \{0, 1, \ldots, v-1\}$ with the property that for any t distinct columns, at least one row takes each value of $\mathbb{Z}_v^t = \{0, 1, \ldots, v-1\}^t$.

[5] when the columns have different order it is called a Mixed Covering Array (MCA)

Basic Definitions

Covering Array Number(CAN)

The covering array construction problem (CACP) consists in constructing a covering array $\mathrm{CA}(N; t, k, v)$ given the parameters t, k, and v in such a way the number of rows N of the covering array is minimal. The smallest N for which a covering array exists is the covering array number (CAN) for the parameters t, k, and v, and it is denoted by

$$\mathrm{CAN}(t, k, v) = \min\{N \mid \exists\, \mathrm{CA}(N; t, k, v)\}. \tag{1}$$

- Even, there is no proof that CACP belongs to the class of NP-Complete problems, some related problems are NP-Complete. For instance: the proof if it is possible to add a new row that provides at least a certain number of t-wise missing combinations [6]
- The search space of the CACP denoted by $\mathcal{S}$ satisfies:

$$v^t \leq \mathcal{S} \leq \binom{v^k}{N} \tag{2}$$

[6] Colbourn [8]

There are reported only a small number of optimal covering arrays:

- The case $\mathrm{CA}(v^t; t, \max(v,t)+1, v)$ when v is a primepower [7]. The CAs constructed in this way are equivalent to OAs of unitary index.
- The case $(v=2) \wedge (t=2)$ where $k \leq \binom{N-1}{\lceil \frac{N}{2} \rceil}$ [8]

In general the determination of the CAN is very difficult and is the theme of a lot of research, but asymptotically (for large k)

$$CAN(t,k,v) \approx v^t \log(k) \tag{3}$$

[7] Bush [5]
[8] Rényi [9], Katona [10], Kleitman and Spencer [11]

Given that:

- The position of the rows in a CA is not relevant, all the CAs obtained by permuting the rows of a CA are isomorphic.
- If all the columns in a CA have the same order (alphabet), all the CAs obtained by permuting the columns of a CA are isomorphic.
- The coverage properties of a CA are not affected if the symbols in one column are permuted (for instance all zeros are exchanged with all ones), all the CAs obtained by permuting symbols within columns are isomorphic.

Definition

For a CA there are $N!k!(v!)^k$ isomorphic CAs. N! permutation of rows, k! permutations of columns, and $(v!)^k$ permutations of symbols in the columns.

Given two CAs A and B CA$(6; 2, 5, 2)$ they are isomorphic if one can be constructed from the other using some permutation of rows, permuation of columns, and permutation of symbols. Considering that τ defines the row permutation, π is the column permutation, and ϕ defines the symbol permutation. In the next figure we illustrate how to construct B from A.

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\tau = (0\ 1\ 2\ 3\ 4\ 5) \qquad \tau' = (3\ 1\ 2\ 5\ 4\ 0)$$
$$\pi = (0\ 1\ 2\ 3\ 4) \qquad \pi' = (4\ 2\ 0\ 3\ 1)$$
$$\phi = (0\ 0\ 0\ 0\ 0) \qquad \phi' = (0\ 1\ 0\ 1\ 0)$$

Figure 2: Using CA A we obtain CA B using τ', π', and ϕ'.

The two cases of optimal CAs mentioned previosuly (OAs with $(\lambda = 1) \wedge (\text{prime power}(v))$, and $(v = 2) \wedge (t = 2)$) result in CAs that do not have redundancy. For any other case the possibility that a CA has a lot of redundancy is very high. A redundant element (usually denoted as wildcard) can take any value and the coverage properties of a CA are not affected (usually are represented with the symbol *).

Definition

One element of a CA is redundant (wildcard) if we can change that element with any value of its alphabet and the coverage properties of the CA are not affected.

For instance the CA A CA$(11; 3, 5, 2)$ donwloaded from the NIST Repository [9] has one row totally redundant. The CA B indicates with * the redundant elements.

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

Figure 3: CA A can be reduced in one row through the detection of redundant symbols shown in CA B as *

[9] http://math.nist.gov/coveringarrays/ipof/tables/table.3.2.html

Table of Contents

There are three main repositories for uniform CAs:

- NIST Repository[10] that contains explicit CAs constructed using the IPOG-F algorithm for $(v = \{2, \ldots, 6\}) \wedge (t = \{2, \ldots, 6\})$.
- Charles Colbourn Repository [11] that lists only the best known sizes for CAs $(v = \{2, \ldots, 25\}) \wedge (t = \{2, \ldots, 6\}$, it doest not provide explicit CAs.
- CinvestavCA Repository [12] currently it provides explicit CAs. It contains good CAs for $(v = \{2, 3\}) \wedge (t = \{2, \ldots, 6\})$.

[10] http://math.nist.gov/coveringarrays/
[11] http://www.public.asu.edu/~ccolbou/src/tabby/catable.html
[12] http://www.tamps.cinvestav.mx/~jtj/authentication.php

Covering Arrays generated by IPOG-F

This page is an appendix to the paper ``Refining the In-Parameter-Order Strategy for Constructing Covering Arrays'' by M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker and D.R. Kuhn (paper in progress, abstract: HTML/PDF). It holds the covering arrays generated by the main algorithm described, IPOG-F. Many values of *t* and *v* are represented. Many of these covering arrays are smaller than currently known in literature, as indicated by the graphs shown in the individual pages. All covering arrays are accessible (in a custom format) for research purposes.

Back to main menu

Choose which CA(*t*,*k*,*v*) to explore:

CA(2,k,2) CA(3,k,2) CA(4,k,2) CA(5,k,2) CA(6,k,2)
CA(2,k,3) CA(3,k,3) CA(4,k,3) CA(5,k,3) CA(6,k,3)
CA(2,k,4) CA(3,k,4) CA(4,k,4) CA(5,k,4) CA(6,k,4)
CA(2,k,5) CA(3,k,5) CA(4,k,5) CA(5,k,5) CA(6,k,5)
CA(2,k,6) CA(3,k,6) CA(4,k,6) CA(5,k,6)

Last Updated: *2008-04-17*

For comments please mail the Covering Arrays Team: coveringarrays at nist dot gov

Table for CA(3,k,3)

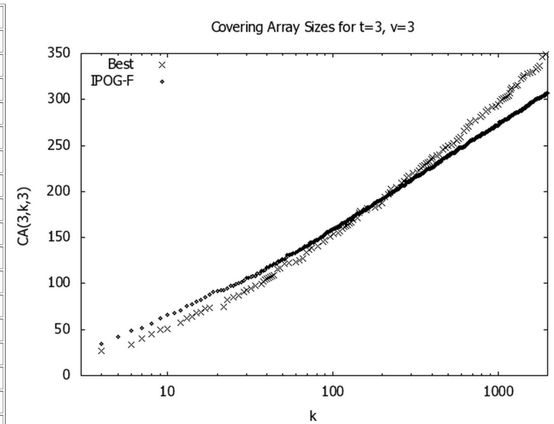
Covering array sizes are given for each k listed, where t=3 and v=3. The arrays, as generated by the IPOG-F program, are given as links to files of a specific format. For comparision purposes, the sizes are graphed as compared to the best known as listed by Charlie Colbourn's CATables page (as accessed when this page was last updated).

Back to menu

Change t: Decrease or Increase

Change v: Decrease or Increase

k	CA(3,k,3)	Filesize
4	34	1.0K
5	42	1.0K
6	49	1.0K
7	52	1.0K
8	56	1.0K
9	62	1.0K
10	66	1.0K
11	68	1.0K
12	71	1.0K
13	76	1.0K
14	77	1.0K
15	80	1.0K
16	82	1.0K
17	85	1.0K
18	88	1.0K
19	91	1.0K
20	92	1.0K



Covering Array Tables for t=2,3,4,5,6

These tables are maintained by Charlie Colbourn on an irregular basis. Please report updates and corrections.

For given t and v, the table (t,k,v) gives the current best known upper bound on CAN(t,k,v), the smallest number of rows in a uniform covering array having k factors each with v levels, with coverage at strength t. Covering array numbers are reported for each k up to 20000 for strength two, 10000 for strengths three through six. At present, the authorities are not given with references.

`Best known' means best reported in the literature, to me via email, or implied by a recursive construction. Sizes are reported when an explicit construction is known, not when a probabilistic argument guarantees existence. However, for certain values of v when t is 4, 5, or 6, a constructive conditional expectation algorithm yields better bounds than those implied by the direct and recursive methods -- in these cases, the accompanying graph shows two lines, of which the lower one shows the bounds from the conditional expectation method.

(2,k,2) (2,k,3) (2,k,4) (2,k,5) (2,k,6) (2,k,7) (2,k,8) (2,k,9) (2,k,10) (2,k,11) (2,k,12) (2,k,13) (2,k,14) (2,k,15) (2,k,16) (2,k,17) (2,k,18) (2,k,19) (2,k,20) (2,k,21) (2,k,22) (2,k,23) (2,k,24) (2,k,25)
(3,k,2) (3,k,3) (3,k,4) (3,k,5) (3,k,6) (3,k,7) (3,k,8) (3,k,9) (3,k,10) (3,k,11) (3,k,12) (3,k,13) (3,k,14) (3,k,15) (3,k,16) (3,k,17) (3,k,18) (3,k,19) (3,k,20) (3,k,21) (3,k,22) (3,k,23) (3,k,24) (3,k,25)
(4,k,2) (4,k,3) (4,k,4) (4,k,5) (4,k,6) (4,k,7) (4,k,8) (4,k,9) (4,k,10) (4,k,11) (4,k,12) (4,k,13) (4,k,14) (4,k,15) (4,k,16) (4,k,17) (4,k,18) (4,k,19) (4,k,20) (4,k,21) (4,k,22) (4,k,23) (4,k,24) (4,k,25)
(5,k,2) (5,k,3) (5,k,4) (5,k,5) (5,k,6) (5,k,7) (5,k,8) (5,k,9) (5,k,10) (5,k,11) (5,k,12) (5,k,13) (5,k,14) (5,k,15) (5,k,16) (5,k,17) (5,k,18) (5,k,19) (5,k,20) (5,k,21) (5,k,22) (5,k,23) (5,k,24) (5,k,25)
(6,k,2) (6,k,3) (6,k,4) (6,k,5) (6,k,6) (6,k,7) (6,k,8) (6,k,9) (6,k,10) (6,k,11) (6,k,12) (6,k,13) (6,k,14) (6,k,15) (6,k,16) (6,k,17) (6,k,18) (6,k,19) (6,k,20) (6,k,21) (6,k,22) (6,k,23) (6,k,24) (6,k,25)

If you are interested in explicit presentations of covering arrays, which are not necessarily the best known, a good place to start is at the NIST Covering Array Tables. Some explicit solutions are also available from Jose Torres Jimenez here -- click on Covering Arrays.

Table for CAN(6,k,10) for k up to 10000

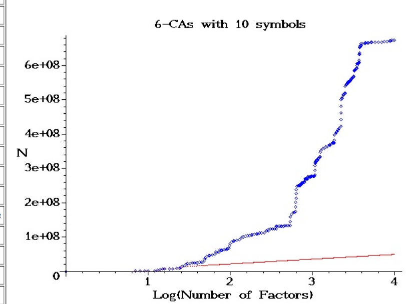
Last Updated Sun Aug 25 06:10:14 MST 2013

Locate the k in the first column that is at least as large as the number of factors in which you are interested. Then let N be the number of rows (tests) given in the second column. A CA(N;6,k,10) exists according to a construction in the reference (cryptically) given in the third column. The accompanying graph plots N vertically against log k (base 10).

Change t: - Change v: - + or go to Global Menu.

k	N	Source
7	1000000	composition
8	1234567	Add a symbol
9	1371741	Add a symbol
10	1524156	Add a symbol
12	1771559	orthogonal array fuse
13	3070619	Add a factor
14	4381199	Add a factor
15	6494489	Add a factor
16	7030109	Martirosyan-Tran van Trung
18	7415333	Martirosyan-Tran van Trung
20	7731599	Martirosyan-Tran van Trung
22	8213599	double OA (Colbourn-Zhou) fuse
24	8308707	Martirosyan-Tran van Trung
25	10927707	Add a factor
26	14200197	Add a factor
28	16953318	Martirosyan-Tran van Trung
29	21168918	Add a factor
30	22098186	Martirosyan-Tran van Trung

Graph:



CINVESTAV TAMAULIPAS
Laboratorio de Tecnologías de Información

Jose Torres-Jimenez

Home | Teaching | Publications | Human Resource Formation | TestDB | Covering Arrays

Authentication - Covering Arrays

For accessing the Covering Arrays Repository, type **guest** in both: Username and Password.

Username:

Password:

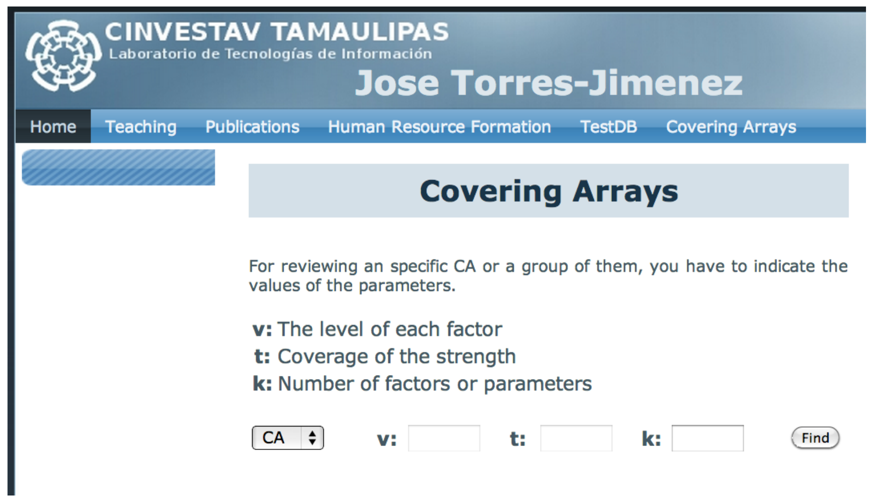
Submit

Please cite this repository like:

For v=3 cite:
H. Avila-George, J. Torres-Jimenez and V. Hernandez, "New bounds for ternary covering arrays using a parallel simulated annealing". *Mathematical Problems in Engineering*, **Volume 2012 (2012)**, Article ID 897027, 19 pages. doi:10.1155/2012/897027.

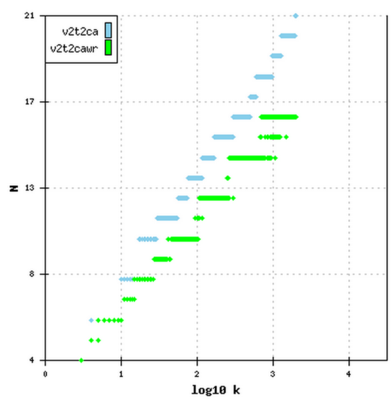
For v=2 cite:
J. Torres-Jimenez and E. Rodriguez-Tello, "New bounds for binary covering arrays using simulated annealing". *Information Sciences* **Volume 185 Issue 1 (2012)**, Pages 137-152. doi: 10.1016/j.ins.2011.09.020.

Note: For request an specific CA that is not available with **guest** account, you can send a request to the indicated e-mail in Contact Info.


CINVESTAV TAMAULIPAS
Laboratorio de Tecnologías de Información
Jose Torres-Jimenez
Home
Teaching
Publications
Human Resource Formation
TestDB
Covering Arrays
Covering Arrays
For reviewing an specific CA or a group of them, you have to indicate the values of the parameters.
v: The level of each factor
t: Coverage of the strength
k: Number of factors or parameters
CA
v:
t:
k:
Find

CINVESTAV TAMAULIPAS
Laboratorio de Tecnologías de Información
Jose Torres-Jimenez

Home | Teaching | Publications | Human Resource Formation | TestDB | Covering Arrays



Tables of Covering Arrays

Resulting tables for the values **v**={2}, **t**={2}

New query   Log Off

N	k	v	t	Save	C1	C2
4	3	2	2			
4	3	2	2		Inputfile= ./v2/t2/N4k3v2...	WC=0
5	4	2	2		Inputfile= ./v2/t2/N6k4v2...	WC=0
6	4	2	2			
6	5	2	2			
6	5	2	2		Inputfile= ./v2/t2/N6k5v2...	WC=2
6	6	2	2			
6	6	2	2		Inputfile= ./v2/t2/N6k6v2...	WC=0
6	7	2	2			
6	7	2	2		Inputfile= ./v2/t2/N6k7v2...	WC=1
6	8	2	2			
6	8	2	2		Inputfile= ./v2/t2/N6k8v2...	WC=0
6	9	2	2			

Table of Contents

Construction Methods

Due to the difficulty of solving the CACP a number of methods have been developed. The methods can be classified in five main categories:

- Exact methods. These methods guarantee to find an optimal CA, but given the exponential search space they are practical only for small CAs.
- Greedy methods. These methods do not guarantee to find an optimal CA, but they are very fast and can be used to construct any CA.
- Metaheuristic methods. These methods does not guarantee to find and optimal CA, they take more time than the greedy methods and in many cases they give better CAs than the ones obtained using greedy methods.
- Algebraic methods. The algebraic methods involved formulas or operations with mathematical objects such as vectors, finite fields, groups or another (usually) small covering arrays.
- Manipulation methods. These methods use covering arrays previously constructed to construct new ones, or transform a CA to more suitable CA.

Table of Contents

Manipulation of CAs

There are some useful operations that can be applied to a covering array previously constructed. This section describes four of them: maximization of constant rows, optimal reduction, wildcard detection, and fusion.

- The maximization of constant rows enable that the product of CAs, and the powering of CAs produce redundant rows that are eliminated easily.
- The optimal reduction of CAs, enable to construct small CAs taking as input CAs with greater number of rows and columns.
- The wildcard detection of CAs, will detect redundant elements in a CA.
- The fusion of CAs, enable to exploit systematically the wildcards to reduce the size of a CA

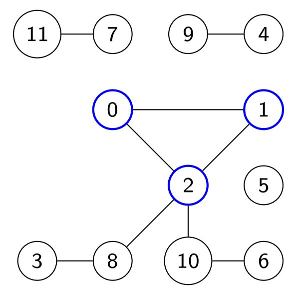
Manipulation of CAs

Maximization of Constant Rows in a CA

- A constant row in a covering array is a row having the same symbol in all its elements. Formally, the i-th row of a covering array $A = (a_{i,j})$ of dimensions $N \times k$ is constant if $a_{i,j} = a_{i,0}$ for $j = 1, 2, \ldots, k - 1$ [13].
- By means of the three operations that produce isomorphic covering arrays it is possible to arrange the symbols of the covering array in order to make constant some of its rows.
- The constant rows are very useful for the methods of multiplication and powering of covering arrays, because if the covering arrays used have constant rows, then it is possible to delete some rows in the resulting covering array.
- This problem can be solved in the domain of graphs. For each row a node is created, and for each pair of rows that are distinct (column by column) an edge is created. The problem is converted to the MAXCLIQUE problem.

[13] Quiz-Ramos [49]

$$\begin{pmatrix} 0 \to & 0 & 2 & 2 & 1 \\ 1 \to & 2 & 1 & 1 & 0 \\ 2 \to & 1 & 0 & 0 & 2 \\ 3 \to & 1 & 0 & 2 & 0 \\ 4 \to & 1 & 1 & 0 & 1 \\ 5 \to & 1 & 2 & 1 & 2 \\ 6 \to & 2 & 0 & 0 & 1 \\ 7 \to & 2 & 1 & 2 & 2 \\ 8 \to & 2 & 2 & 1 & 1 \\ 9 \to & 0 & 0 & 1 & 2 \\ 10 \to & 0 & 1 & 1 & 0 \\ 11 \to & 0 & 2 & 0 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 0 & 1 \\ 2 & 1 & 2 & 0 \\ 2 & 0 & 1 & 2 \\ 1 & 2 & 2 & 0 \\ 1 & 1 & 0 & 2 \\ 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 1 \end{pmatrix}$$

- Given a covering array A the **O**ptimal **S**hortening of **C**overing **AR**rays (**OSCAR**) problem consists in finding a submatrix B of a determined size such that the number of missing tuples in B is minimized [14].
- Let be δ and Δ two integers such that $0 \leq \delta \leq N - v^t$, $0 \leq \Delta \leq k - t$, with the condition that at least one of them is greater than zero.
- The OSCAR problem consists in finding a submatrix B of A of size $(N - \delta) \times (k - \Delta)$ such that the number of missing tuples in B is minimal.
- It was proved that the OSCAR problem is NP-Complete by reducing the MAXCOVER problem to the OSCAR problem.
- The search space of the OSCAR problem is $\binom{N}{N-\delta}\binom{k}{k-\Delta}$.

[14] Carrizales-Turrubiates [50]

0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	1	1	1	0	0	1
0	1	1	0	1	1	0
1	0	1	0	0	1	1
1	1	0	1	1	0	0

CA(6; 2, 7, 2)

0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	1	1	1	0	0	1
0	1	1	0	1	1	0
1	0	1	0	0	1	1
1	1	0	1	1	0	0

Row and columns to delete

0	0	0	0
0	0	1	1
0	1	0	1
1	0	0	1
1	1	1	0

CA(5; 2, 4, 2)

Figure 4: The deletion of one row and three columns of the covering array CA(6; 2, 7, 2) produces the covering array CA(5; 2, 4, 2).

Manipulation of CAs

Redundant Elements Detection in a CA

Sometimes a covering array has entries that can be freely modified without affecting the coverage properties of the covering array, that is, without affecting the number of missing tuples of the array. These entries are called *wildcards* and are commonly represented by the symbol $*$. Figure 5 shows at the left the covering array CA$(7; 2, 8, 2)$, and shows at right the same covering array with the wildcards it contains.

0	0	1	0	1	0	0	0
0	1	0	0	1	0	1	1
1	1	1	1	0	0	1	0
0	0	1	1	0	1	1	1
1	1	0	1	1	1	0	1
1	0	0	0	0	1	0	0
0	0	1	1	0	0	0	1

0	$*$	1	$*$	1	0	0	0
0	1	0	0	$*$	0	1	$*$
1	1	1	1	0	0	1	0
0	0	1	0	1	1	1	1
1	1	0	1	1	1	0	1
1	0	0	0	0	1	0	0
0	0	$*$	1	0	0	$*$	1

Figure 5: Wildcards in the covering array CA$(7; 2, 8, 2)$.

Wildcard detection is very important for some postoptimization process for covering arrays. A postoptimization process is a process that tries to reduce the number of rows of a given covering array. One of these process that uses wildcards is the method of Nayeri et al. [15]
A methodology to maximize the number of wildcards in a covering array [16] proposed three main steps:

- to determine the tuples covered only once;
- to determine the unfixed symbols;
- to enumerate all the possible wildcard configurations.

The first and second steps are seen in the next figure. The elements that paticipate in t-wise combinatios covered once are seen in the second matrix as 0 or 1, the unfixed elements are shown in the second matrix as U. The third step can be implemented using a greedy approach or an exact approach, but in any case both approaches tries to minimize the U elements that become fixed (to some value of Z_v), and in this way the remaining U elements are maximized and converted to wildcards ($*$).

[15] Nayeri et al. [51]
[16] Gonzalez-Hernandez et al. [52]

0	0	1	0	1	0	0	0
0	1	0	0	1	0	1	1
1	1	1	1	0	0	1	0
0	0	1	0	1	1	1	1
1	1	0	1	1	1	0	1
1	0	0	0	0	1	0	0
0	0	1	1	0	0	0	1

0	*U*	*U*	*U*	1	*U*	*U*	0
0	1	0	0	*U*	0	1	*U*
1	1	1	1	0	0	1	0
0	0	1	*U*	*U*	1	1	*U*
1	1	0	1	1	1	0	1
1	0	0	0	0	1	*U*	0
0	0	*U*	1	0	*U*	*U*	1

Figure 6: The entries of the covering array CA$(7; 2, 8, 2)$ marked with U are the entries that may become a wildcard.

Manipulation of CAs
Fusion of a CA

Colbourn [17] defined the *fusion* of a CA as:

$$\mathrm{CAN}(t,k,v) \leq \mathrm{CAN}(t,k,v+1) - \begin{cases} 3 \text{ if } t=2,\ k \leq v+1,\ v \text{ is a prime power} \\ 2 \text{ otherwise} \end{cases} \tag{4}$$

The basic mechanism of the fusion operator is to obtain from a covering array $A = \mathrm{CA}(N;t,k,v)$ another covering array $B = \mathrm{CA}(M;t,k,v-1)$ of smaller size by replacing the occurrences of the symbol v in A for symbols of the set $\{0,1,\ldots,v-2\}$, and by deleting three or two rows according the cases of expression (4).

[17] Colbourn [53], and Colbourn et al. [54]

A generalization of the fusion operator to CAs [18] was proposed. The generalization consists in three main components:

- A wildcard detector, that maximizes the wildcards in a CA. The implementation of this component could be greedy or exact.
- A row replacer that exploits the redundant elements to reduce the rows of the CA. This component copies non-wildcard elements to corresponding wildcard elements (in the same column) in order to obtain a redundant row that can be eliminated. This procedure could end when a redundant row is found, or when the best non-redundant row is found (one row that consumes the less number of wildcard elements). In case a redundant row could not be found, a quasi-CA with the minimum missing t-wise combinations is delivered.
- A metaheuristic algorithm that tries to make zero the number of missing tuples. This procedure is run until the number of missing tuples is zero or a maximum time is consumed.

When the order of the input CA is greater than the order of the output CA an order reductor algorithm is used.

[18] Rodriguez-Cristerna [55]

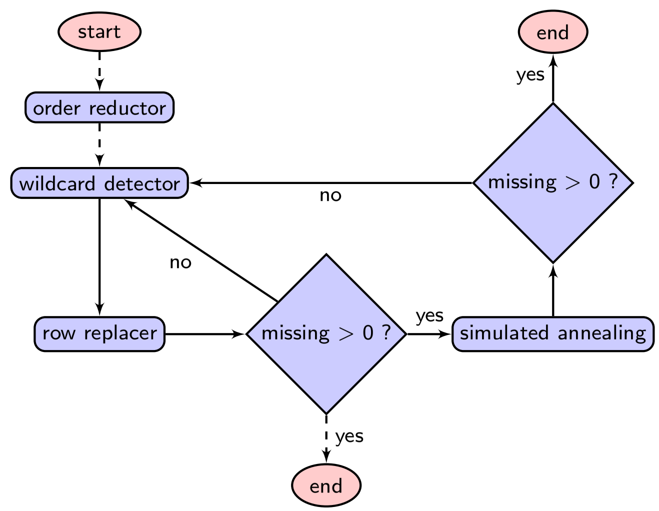
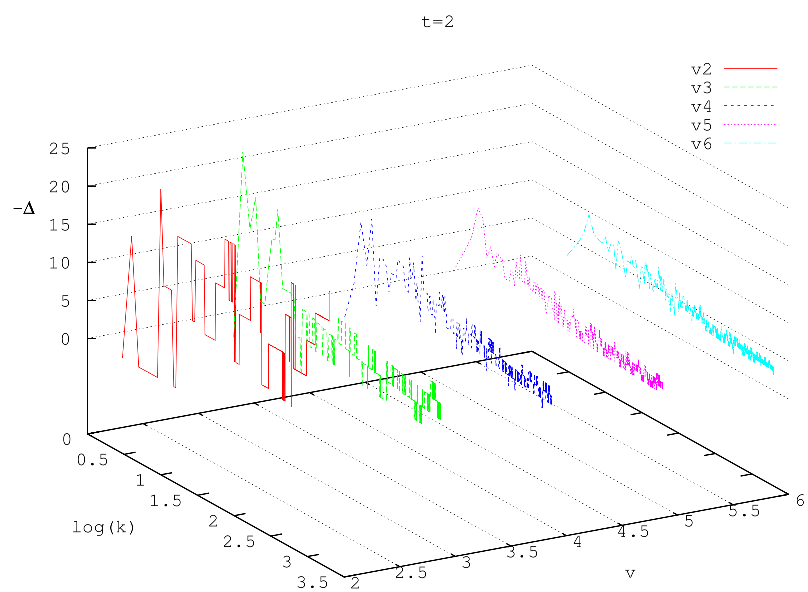
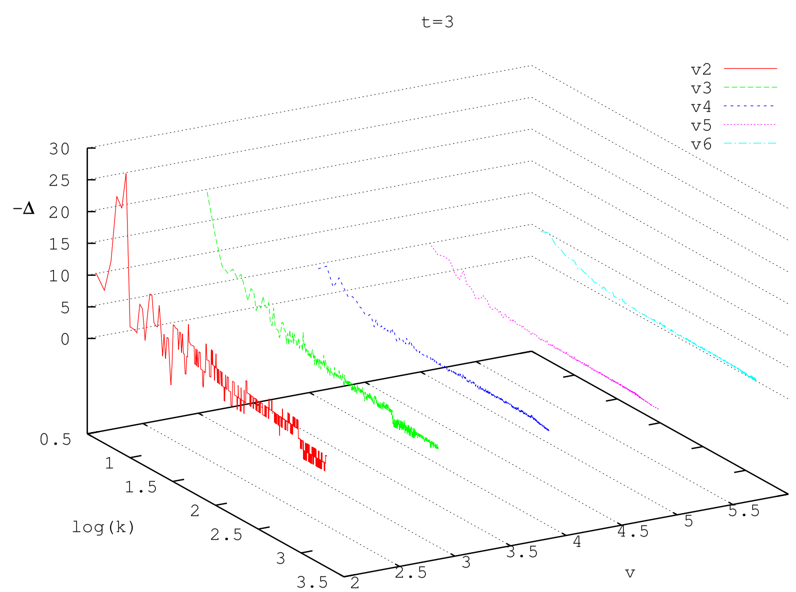

start
order reductor
wildcard detector
row replacer
missing > 0 ?
yes
end
no
simulated annealing
missing > 0 ?
yes
end
no

Table of Contents

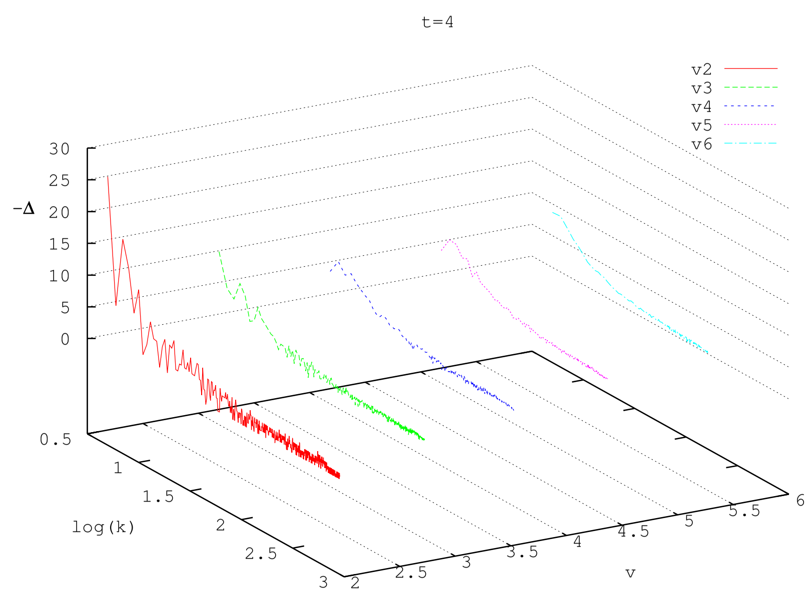
Postprocessig NIST Repository

- As part of the Automated Combinatorial Testing for Software (ACTS) project at NIST, a repository of covering arrays is publicly available. An important opportunity is the exploitation of the redundancy of coverage that the covering arrays have in order to reduce its size.
- The application of the Generalized fusion operator produces a lot of improvements of the CA of the NIST Repository.
- The NIST repository was processed using the generalized fusion and We have found 349 new upper bounds.
- We have employed more than 1.5 million of hours of the Xiuhcoatl cluster of the CINVESTAV.
- A total of 21,964 CAs were processed and 20,956 were improved.
- An average reduction of 3.85% was attained.

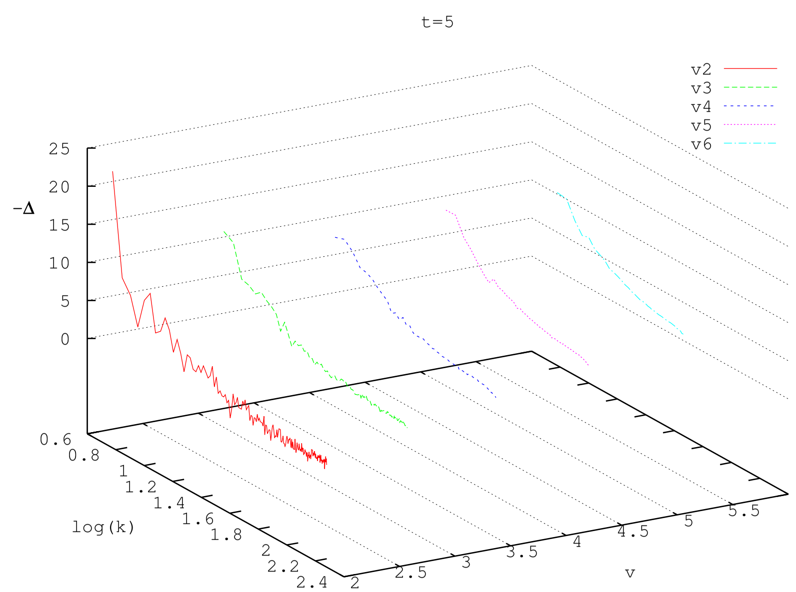

t=2
v2
v3
v4
v5
v6
-Δ
25
20
15
10
5
0
0
0.5
1
1.5
2
2.5
3
3.5
log(k)
2
2.5
3
3.5
4
4.5
5
5.5
6
v



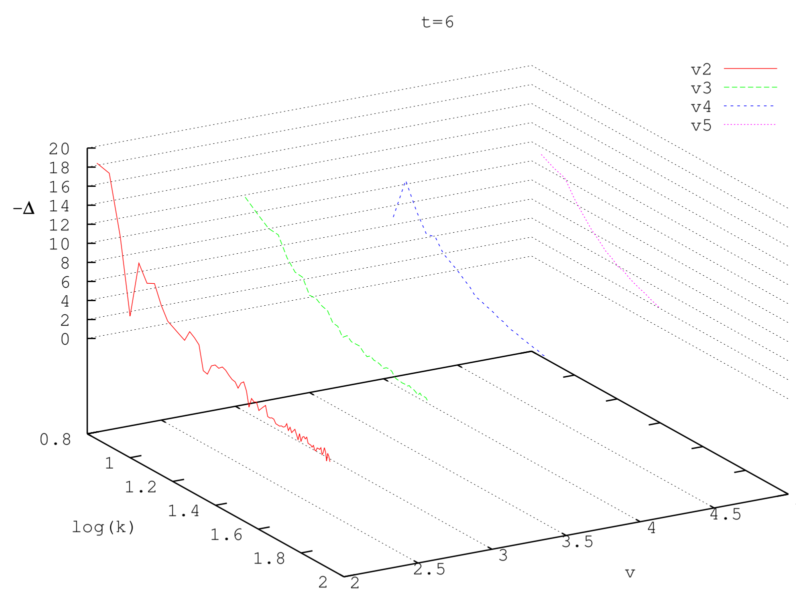
t=3
v2
v3
v4
v5
v6
-Δ
30
25
20
15
10
5
0
0.5
1
1.5
2
2.5
3
3.5
log(k)
2
2.5
3
3.5
4
4.5
5
5.5
6
v



t=4
v2
v3
v4
v5
v6
-Δ
30
25
20
15
10
5
0
0.5
1
1.5
2
2.5
3
log(k)
2
2.5
3
3.5
4
4.5
5
5.5
6
v



t=5
v2
v3
v4
v5
v6
-Δ
25
20
15
10
5
0
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
log(k)
2
2.5
3
3.5
4
4.5
5
5.5
6
v



t=6
v2
v3
v4
v5
-Δ
20
18
16
14
12
10
8
6
4
2
0
0.8
1
1.2
1.4
1.6
1.8
2
log(k)
2
2.5
3
3.5
4
4.5
5
v

		v3t4				*v3t4*				*v6t4*	
Id.	k	IPOG-F	FG-F1	Id.	k	IPOG-F	FG-F1	Id.	k	IPOG-F	FG-F1
1	315	968	964	44	377	1011	1003	85	85	11441	11384
2	316	969	964	45	379	1013	1007	86	86	11484	11407
3	317	969	963	46	386	1017	1009	87	87	11533	11478
4	318	971	965	47	405	1027	1024	88	88	11577	11504
5	319	971	963	48	439	1046	1045	89	89	11625	11581
6	320	971	963	49	447	1052	1050	90	90	11666	11591
7	321	974	965			*v6t4*		91	91	11710	11630
8	322	974	966	Id.	k	IPOG-F	FG-F1	92	92	11753	11671
9	323	975	970	50	49	9323	9212	93	93	11790	11729
10	324	976	965	51	50	9393	9294	94	94	11833	11762
11	325	976	970	52	51	9466	9397	95	95	11874	11800
12	326	976	974	53	52	9550	9463	96	96	11913	11884
13	327	977	973	54	53	9623	9540	97	97	11956	11918
14	328	978	969	55	55	9762	9673	98	98	11997	11924
15	329	979	975	56	56	9828	9742	99	99	12038	11949
16	330	979	976	57	57	9900	9813	100	100	12085	12003
17	331	981	977	58	58	9964	9869	101	101	12120	12036
18	332	981	973	59	59	10032	9948	102	102	12148	12140
19	333	983	975	60	60	10097	10013	103	103	12194	12120
20	335	983	979	61	61	10163	10067	104	104	12231	12185
21	336	983	978	62	62	10219	10142	105	105	12267	12196
22	337	984	977	63	63	10282	10198	106	106	12306	12240
23	338	985	977	64	64	10347	10250	107	107	12343	12268
24	339	986	977	65	65	10398	10328	108	109	12408	12388
25	340	987	978	66	66	10463	10368	109	110	12445	12380
26	341	987	979	67	67	10520	10441	110	112	12517	12449
27	342	987	981	68	68	10578	10478	111	116	12651	12593
28	343	990	985	69	69	10633	10572	112	118	12716	12698
29	345	991	984	70	70	10693	10599	113	120	12785	12734
30	346	992	985	71	71	10745	10676	114	121	12816	12757
31	347	992	985	72	72	10798	10744	115	128	13036	13007
32	348	992	987	73	73	10850	10758	116	129	13062	13056
33	349	992	986	74	74	10909	10821	117	133	13192	13146
34	351	993	991	75	75	10958	10882	118	149	13634	13606
35	353	994	987	76	76	11012	10959			*v3t5*	
36	360	999	994	77	77	11057	10992	Id.	k	IPOG-F	FG-F1
37	361	1001	995	78	78	11110	11017	119	35	1867	1826
38	362	1001	996	79	79	11158	11100	120	36	1895	1850
39	364	1002	997	80	80	11203	11121	121	37	1920	1882
40	368	1007	1002	81	81	11253	11187	122	38	1947	1909
41	370	1008	1001	82	82	11303	11219	123	39	1974	1933
42	373	1009	1003	83	83	11353	11282	124	40	1997	1949
43	375	1009	1005	84	84	11397	11319	125	41	2023	1975

v3t5

Id.	k	IPOG-F	FG-F1
126	42	2046	2002
127	43	2070	2022
128	44	2091	2050
129	45	2112	2071
130	46	2130	2086
131	47	2150	2112
132	48	2174	2134
133	49	2191	2154
134	50	2213	2182
135	51	2231	2198
136	52	2251	2217
137	53	2269	2232
138	54	2289	2246
139	55	2309	2277
140	56	2327	2281
141	57	2342	2304
142	58	2358	2316
143	59	2373	2335
144	60	2394	2347
145	61	2408	2370
146	62	2425	2387
147	63	2440	2397
148	64	2451	2413
149	65	2468	2439
150	66	2482	2447
151	67	2498	2459
152	68	2512	2477
153	69	2527	2487
154	70	2542	2504
155	71	2555	2518
156	72	2573	2531
157	73	2584	2546
158	74	2597	2564
159	75	2609	2588
160	76	2625	2593
161	77	2639	2608
162	78	2648	2615
163	79	2661	2625
164	80	2673	2635
165	81	2686	2651
166	82	2700	2668
167	83	2710	2677
168	84	2725	2684

v3t5

Id.	k	IPOG-F	FG-F1
169	85	2739	2700
170	86	2749	2706
171	87	2762	2728
172	88	2770	2736
173	89	2783	2747
174	90	2792	2761
175	91	2805	2772
176	92	2815	2783
177	93	2825	2798
178	94	2836	2797
179	95	2847	2813
180	96	2857	2819
181	97	2868	2834
182	98	2877	2838
183	99	2885	2854
184	100	2895	2862
185	101	2909	2871
186	102	2920	2878
187	103	2928	2898
188	104	2938	2901
189	105	2945	2908
190	107	2962	2928

v4t5

Id.	k	IPOG-F	FG-F1
191	26	6957	6775
192	27	7116	6937
193	28	7267	7088
194	29	7414	7247
195	30	7555	7379
196	31	7691	7527
197	32	7816	7649
198	33	7939	7782
199	34	8064	7907
200	35	8183	8023
201	36	8301	8137
202	37	8420	8259
203	38	8530	8376
204	39	8629	8477
205	40	8737	8583
206	41	8847	8693
207	42	8945	8791
208	43	9035	8890
209	44	9137	8979

v4t5

Id.	k	IPOG-F	FG-F1
210	45	9227	9104
211	46	9320	9176
212	47	9406	9262
213	48	9501	9357
214	49	9588	9453
215	50	9673	9520
216	51	9755	9621
217	52	9835	9682
218	53	9922	9769
219	54	9998	9849
220	55	10079	9927
221	56	10155	10007
222	57	10232	10082
223	59	10379	10231
224	60	10454	10302
225	62	10590	10442
226	63	10650	10509

v5t5

Id.	k	IPOG-F	FG-F1
227	21	18779	18260
228	22	114775	111818
229	23	118587	115802
230	24	122201	119500
231	25	125683	123108

v6t5

Id.	k	IPOG-F	FG-F1
232	17	40334	38976
233	18	42102	40820
234	19	43833	42554
235	20	45425	44224
236	21	46970	45784
237	22	48479	47352
238	23	49924	48838
239	24	51287	50180
240	25	52604	51505
241	26	53850	52814
242	27	55069	54032
243	28	56225	55275
244	29	57363	56353
245	30	58468	57503
246	31	59529	58576
247	32	60570	59612
248	33	61562	60608

Id.	k	IPOG-F	FG-F1
		v6t5	
249	34	62527	61612
250	35	63471	62557
251	36	64399	63519
		v2t6	
Id.	k	IPOG-F	FG-F1
252	41	572	556
253	42	579	561
254	43	590	570
255	44	594	580
256	45	603	589
257	46	611	596
258	47	617	602
259	48	625	612
260	49	630	617
261	50	636	623
262	51	643	629
263	52	650	636
264	53	656	641
265	54	662	644
266	55	667	653
267	56	672	655
268	57	677	663
269	58	683	667
270	59	689	671
271	60	695	680
272	61	699	687
273	62	703	686
274	63	709	697
275	64	715	699
276	65	721	704
277	66	725	709
278	67	728	710
279	68	732	718
280	69	738	724
281	70	743	729
282	71	747	734
283	72	751	736
284	73	755	743
285	74	761	749
286	75	766	751
287	76	770	755
288	77	773	758
289	78	777	760

Id.	k	IPOG-F	FG-F1
		v2t6	
290	79	782	770
291	80	785	771
292	81	791	772
293	82	794	778
294	83	796	787
295	84	800	784
296	85	804	790
297	86	809	799
		v3t6	
Id.	k	IPOG-F	FG-F1
298	26	5709	5544
299	27	5853	5667
300	28	6003	5827
301	29	6150	5969
302	30	6281	6103
303	31	6413	6245
304	32	6535	6348
305	33	6656	6461
306	34	6772	6583
307	35	6877	6715
308	36	6989	6832
309	37	7092	6932
310	38	7194	7036
311	39	7293	7131
312	40	7391	7233
313	41	7490	7315
314	42	7574	7411
315	43	7672	7506
316	44	7757	7600
317	45	7845	7702
318	46	7938	7766
319	47	8013	7856
320	50	8256	8108
321	51	8333	8179
		v4t6	
Id.	k	IPOG-F	FG-F1
322	19	26392	25430
323	20	27534	26564
324	21	28625	27676
325	22	29640	28735
326	23	30636	29720
327	24	31591	30724
328	25	32501	31654

Id.	k	IPOG-F	FG-F1
		v4t6	
329	26	33369	32513
330	27	34187	33356
331	28	35006	34198
332	29	35791	34971
333	30	36570	35778
334	31	37305	36515
335	32	38015	37255
		v5t6	
Id.	k	IPOG-F	FG-F1
336	11	56615	52471
337	12	63620	59622
338	13	70190	66275
339	14	76390	72680
340	15	82139	78480
341	16	87559	84102
342	18	97605	94263
343	19	102208	98994
344	20	106642	103514
345	21	110842	107773
346	22	114775	111818
347	23	118587	115802
348	24	122201	119500
349	25	125683	123108

Table of Contents

Conlusions

- The Covering Arrays (CA) represent an excelent option when it is desired to guarantee a certain level of coverage in the testing of a software component and to use a small number of test cases.
- We have presented basic definitions needed to understand much of the research related to covering arrays.
- We have grouped a lot of research reports related to the construction of CAs, the groups were: Exact methods, Greedy Methods, Metaheuristic Methods, and Manipulation Methods.
- The generalized fusion operator is based on the exploitation of the wildcards of a CA in order to reduce the number of rows of the CA.
- We have processed all the NIST CA Repository using more than 1.5 millon of CPU hours (Xiuhcoatl Cinvestav Cluster) and have found 349 new upper bounds.
- It is important to highlight that the exploitation of the redundancy of coverage in CAs can enable the creation of better CAs (with lower number of rows).

Thanks!!!

References I

[1] B. Hailpern and P. Santhanam, "Software debugging, testing, and verification," *IBM Systems Journal*, vol. 41, no. 1, pp. 4 –12, 2002.

[2] A. Hartman, "Software and hardware testing using combinatorial covering suites," in *Graph Theory, Combinatorics and Algorithms*, ser. Operations Research/Computer Science Interfaces Series. Springer US, 2005, vol. 34, pp. 237–266.

[3] D. Cohen, S. Dalal, J. Parelius, and G. Patton, "The combinatorial design approach to automatic test generation," *Software, IEEE*, vol. 13, no. 5, pp. 83 –88, sep 1996.

[4] D. Cohen, S. Dalal, M. Fredman, and G. Patton, "The aetg system: An approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol. 23, no. 7, pp. 437–444, jul 1997.

References II

[5] K. Bush, "Orthogonal arrays of index unity," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 426–434, 1952.

[6] J. Torres-Jimenez, N. Rangel-Valdez, L. Gonzalez-Hernandez, and H. Avila-George, "Construction of logarithm tables for Galois Fields," *International Journal of Mathematical Education in Science and Technology*, vol. 42, no. 1, pp. 91–102, Oct. 2010.

[7] *Cryptography and Security in Computing*. InTech, 2012, ch. Construction of Orthogonal Arrays of Index Unity Using Logarithm Tables for Galois Fields.

[8] C. Colbourn, "Combinatorial aspects of covering arrays," *Le Matematiche*, vol. 59, no. 1,2, pp. 125–172, 2004.

[9] *Foundations of Probability*. Holden-Day, 1970.

References III

[10] G. Katona, "Two applications (for search theory and truth functions) of sperner type theorems," *Periodica Mathematica Hungarica*, 1973.

[11] D. Kleitman and J. Spencer, "Families of k-independent sets," *Discrete Mathematics*, 1973.

[12] D. Kuhn, J. Higdon, J. Lawrence, R. Kacker, and L. Y., "Combinatorial methods for event sequence testing," in *Workshop on Combinatorial Testing (CT) In conjunction with International Conference on Software Testing*.

[13] J. Yan and J. Zhang, "Backtracking algorithms and search heuristics to generate test suites for combinatorial testing," in *Proceedings of the 30th Annual International Computer Software and Applications Conference - Volume 01*, ser. COMPSAC '06. Washington, DC, USA: IEEE Computer

References IV

Society, 2006, pp. 385–394. [Online]. Available: http://dx.doi.org/10.1109/COMPSAC.2006.33

[14] J. Bracho-Rios, J. Torres-Jimenez, and E. Rodriguez-Tello, "A new backtracking algorithm for constructing binary covering arrays of variable strength," in *MICAI 2009: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, A. Aguirre, R. Borja, and C. Garciá, Eds. Springer Berlin / Heidelberg, 2009, vol. 5845, pp. 397–407. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-05258-3_35

[15] B. Hnich, S. D. Prestwich, E. Selensky, and B. M. Smith, "Constraint models for the covering test problem," *Constraints*, vol. 11, no. 2-3, pp. 199–219, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1007/s10601-006-7094-9

References V

[16] D. Lopez-Escogido, J. Torres-Jimenez, E. Rodriguez-Tello, and N. Rangel-Valdez, "Strength two covering arrays construction using a sat representation," in *MICAI*, 2008, pp. 44–53.

[17] M. Banbara, H. Matsunaka, N. Tamura, and K. Inoue, "Generating combinatorial test cases by efficient sat encodings suitable for cdcl sat solvers," in *Proceedings of the 17th international conference on Logic for programming, artificial intelligence, and reasoning*, ser. LPAR'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 112–126.

[18] K. Meagher, "Non-isomorphic generation of covering arrays," University of Regina, Tech. Rep., 2002.

[19] R. Bryce and C. Colbourn, "The density algorithm for pairwise interaction testing," *Software Testing, Verification and Reliability*, vol. 17, no. 3, pp. 159–182, 2007.

References VI

[20] Y. Lei, R. Kacker, D. Kuhn, V. Okun, and J. Lawrence, "Ipog: a general strategy for t-way software testing," in *Proceedings of the 14th Annual IEEE International Conference and Workshops on theEngineering of Computer-Based Systems. ECBS '07.*, Tucson, AZ, USA, mar 2007, pp. 549 –556.

[21] M. Forbes, J. Lawrence, Y. Lei, R. Kacker, and D. Kuhn, "Refining the in-parameter-order strategy for constructing covering arrays," *Journal of Research of the National Institute of Standards and Technology*, vol. 113, no. 5, pp. 287–297, 2008.

References VII

[22] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "Ipog: A general strategy for t-way software testing," in *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, ser. ECBS '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 549–556. [Online]. Available: http://dx.doi.org/10.1109/ECBS.2007.47

[23] A. Calvagna and A. Gargantini, "T-wise combinatorial interaction test suites construction based on coverage inheritance," *Software Testing, Verification and Reliability*, vol. 22, no. 7, pp. 507–526, 2012. [Online]. Available: http://dx.doi.org/10.1002/stvr.466

References VIII

[24] E. Covarrubias-Flores, "Cálculo de covering arrays binarios de fuerza variable, usando un algoritmo de recocido simulado," Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2008.

[25] J. Torres-Jimenez and E. Rodriguez-Tello, "New bounds for binary covering arrays using simulated annealing," *Information Sciences*, vol. 185, no. 1, pp. 137–152, February 2012.

[26] H. Avila-George, J. Torres-Jimenez, and V. Hernández, "Parallel simulated annealing for the covering arrays construction problem," in *To appear in the 18th International Conference on Parallel and Distributed Processing Techniques and Applications*, 2012.

References IX

[27] H. Avila-George, J. Torres-Jimenez, and V. Hernandez, "New bounds for ternary covering arrays using a parallel simulated annealing," *Mathematical Problems in Engineering*, Accepted on July 07, 2012.

[28] K. Nurmela, "Upper bounds for covering arrays by tabu search," *Discrete Applied Mathematics*, vol. 138, no. 1-2, pp. 143 – 152, 2004, optimal Discrete Structures and Algorithms.

[29] L. Gonzalez-Hernandez, N. Rangel-Valdez, and J. Torres-Jimenez, "Construction of mixed covering arrays of variable strength using a tabu search approach," in *Combinatorial Optimization and Applications*, ser. Lecture Notes in Computer Science, W. Wu and O. Daescu, Eds. Springer Berlin / Heidelberg, 2010, vol. 6508, pp. 51–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17458-2_6

References X

[30] J. Stardom, "Metaheuristics and the search for covering and packing arrays," Master's thesis, Simon Fraser University, 2001.

[31] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, vol. 1, sept. 2004, pp. 72–77.

[32] D. T. Tang and L. S. Woo, "Exhaustive test pattern generation with constant weight vectors," *IEEE Trans. Comput.*, vol. 32, no. 12, pp. 1145–1150, Dec. 1983. [Online]. Available: http://dx.doi.org/10.1109/TC.1983.1676175

References XI

[33] J. Martinez-Pena and J. Torres-Jimenez, "A branch and bound algorithm for ternary covering arrays construction using trinomial coefficients," *Research in Computing Science*, vol. 49, pp. 61–71, 2010.

[34] J. Martinez-Pena, J. Torres-Jimenez, N. Rangel-Valdez, and H. Avila-George, "A heuristic approach for constructing ternary covering arrays using trinomial coefficients," in *Proceedings of the 12th Ibero-American conference on Advances in artificial intelligence*, ser. IBERAMIA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 572–581. [Online]. Available: http://dl.acm.org/citation.cfm?id=1948131.1948203

[35] C. Colbourn, "Covering arrays from cyclotomy," *Designs, Codes and Cryptography*, vol. 55, pp. 201–219, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10623-009-9333-8

References XII

[36] K. Meagher and B. Stevens, "Group construction of covering arrays," *Journal of Combinatorial Designs*, vol. 13, no. 1, pp. 70–77, 2005. [Online]. Available: http://dx.doi.org/10.1002/jcd.20035

[37] J. Lobb, C. Colbourn, P. Danziger, B. Stevens, and J. Torres-Jimenez, "Cover starters for covering arrays of strength two," *Discrete Mathematics*, 2011, in press.

[38] N. J. A. Sloane, "Covering arrays and intersecting codes," *Journal of Combinatorial Designs*, vol. 1, no. 1, pp. 51–63, 1993.

[39] G. Roux, "k-propriétés dans des tableaux de *n* colonnes; cas particulier de la k-surjectivité et de la k-permutivité," Ph.D. dissertation, University of Paris, 1987.

References XIII

[40] M. Chateauneuf and D. Kreher, "On the state of strength-three covering arrays," *Journal of Combinatorial Designs*, vol. 10, no. 4, pp. 217–238, 2002.

[41] M. B. Cohen, C. J. Colbourn, and A. C. Lin, "Constructing strength three covering arrays with augmented annealing," *Discrete Mathematics*, vol. 308, no. 13, pp. 2709–2722, 2008.

[42] C. J. Colbourn, S. S. Martirosyan, T. Trung, and R. A. Walker, II, "Roux-type constructions for covering arrays of strengths three and four," *Des. Codes Cryptography*, vol. 41, no. 1, pp. 33–57, Oct. 2006. [Online]. Available: http://dx.doi.org/10.1007/s10623-006-0020-8

[43] S. Martirosyan and T. v. Trung, "On t-covering arrays," *Designs, Codes and Cryptography*, vol. 32, pp. 323–339, 2004, 10.1023/B:DESI.0000029232.40302.6d. [Online]. Available: http://dx.doi.org/10.1023/B:DESI.0000029232.40302.6d

References XIV

[44] C. J. Colbourn, S. S. Martirosyan, G. L. Mullen, D. Shasha, G. B. Sherwood, and J. L. Yucas, "Products of mixed covering arrays of strength two," *Journal of Combinatorial Designs*, vol. 14, no. 2, pp. 124–138, 2006.

[45] A. Hartman, "Software and hardware testing using combinatorial covering suites," in *Graph Theory, Combinatorics and Algorithms*, ser. Operations Research/Computer Science Interfaces Series, M. C. Golumbic and I. B.-A. Hartman, Eds. Springer US, 2005, vol. 34, pp. 237–266. [Online]. Available: http://dx.doi.org/10.1007/0-387-25036-0_10

[46] H. Avila-George, "Verificación de covering arrays utilizando supercomputación y computación grid," Master's thesis, Universidad Politécnica de Valencia, 2010.

References XV

[47] H. Avila-George, J. Torres-Jimenez, V. Hernández, and N. Rangel-Valdez, “A parallel algorithm for the verification of covering arrays,” in *Proceedings of The International Conference on Parallel and Distributed Techniques and Applications(PDPTA 2011)*, Las Vegas Nevada, USA, 18-21 July 2011, pp. 879–885. [Online]. Available: http://world-comp.org/p2011/PDP8061.pdf

[48] ——, “Verification of general and cyclic covering arrays using grid computing,” in *Proceedings of the Third international conference on Data management in grid and peer-to-peer systems*, ser. Globe’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 112–123. [Online]. Available: http://dl.acm.org/citation.cfm?id=1885229.1885242