[CAAI Transactions on Intelligence Technology 2 (2017) 31](http://dx.doi.org/10.1016/j.trit.2016.12.006)e[38](http://dx.doi.org/10.1016/j.trit.2016.12.006)



Contents lists available at [ScienceDirect](http://www.sciencedirect.com/science/journal/24682322)

CAAI Transactions on Intelligence Technology

journal homepage: [http://www.journals.elsevier.com/caai-transactions-on-intelligence- technology/](http://www.journals.elsevier.com/caai-transactions-on-intelligence-technology/)

Original article

Metaheuristic post-optimization of the NIST repository of covering arrays

Jose Torres-Jimenez[\*](#_bookmark0), Arturo Rodriguez-Cristerna

*CINVESTAV-Tamaulipas, Information Technology Laboratory, Km. 5.5 Carretera Cd., Victoria, Tamaulipas, Mexico*

# a r t i c l e i n f o

*Article history:*

Received 19 September 2016

Accepted 25 December 2016

Available online 16 February 2017

*Keywords:*

Covering arrays

NIST repository of covering arrays Metaheuristic post-processing algorithms

# a b s t r a c t

Construction of Covering Arrays (CA) with minimum possible number of rows is challenging. Often the available CA have redundant combinatorial interaction that could be removed to reduce the number of rows. This paper addresses the problem of removing redundancy of CA using a metaheuristic post- optimization (MPO) approach. Our approach consists of three main components: a redundancy detec- tor (RD); a row reducer (RR); and a missing-combinations reducer (MCR). The MCR is a metaheuristic component implemented using a simulated annealing algorithm. MPO was instantiated with 21, 964 CA taken from the National Institute of Standards and Technology (NIST) repository. It is a remarkable result that this instantiation of MPO has delivered 349 new upper bounds for these CA.

© 2017 Chongqing University of Technology. Production and hosting by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

The use of software has permeated many areas of human ac- tivity, so the reliability of software has become important world- wide. It is estimated that software testing consumes about 50% of the cost of developing a new piece of software. A 2002 NIST report

[[23]](#_bookmark25) indicates that the cost of an inadequate infrastructure for software testing was in the range of $22.2 to $59.5 billion (US dollars). Reducing this cost is not only important but the design and implementation of adequate software testing procedures is critical for the reliability of many electronic and mechanical systems, even

more so in complex and important systems, such as space shuttles Lions and et al. [[16]](#_bookmark22).

According to Myers et al. [[17]](#_bookmark23) functional software testing methods may be divided into two main categories: white-box testing and black-box testing. The design of white-box testing suites requires source code of the software under examination. Some testing strategies based on the white-box approach are: statement coverage, decision coverage, condition coverage, decision-condition coverage and multiple-condition coverage. The building of test suites using white-box strategies is more

\* Corresponding author.

*E-mail addresses:* [jtj@cinvestav.mx](mailto:jtj@cinvestav.mx) (J. Torres-Jimenez), [arodriguez@tamps.](mailto:arodriguez@tamps.cinvestav.mx) [cinvestav.mx](mailto:arodriguez@tamps.cinvestav.mx) (A. Rodriguez-Cristerna).

Peer review under responsibility of Chongqing University of Technology.

challenging than for black-box strategies, since white-box strate- gies are based on knowledge of the internal structure of the system. Furthermore, if the system is modified, then tests must be rede- signed to satisfy the new version of the system. On the other hand, the design of black box testing suites does not require source code of the software under examination. It compares actual behaviour against expected behaviour based on the functionality and the specification of the software system under examination. Some black-box testing strategies are: exhaustive input testing, equiva- lence partitioning, boundary-value analysis, cause-effect graphing, error guessing, and *combinatorial interaction testing*.

It is easy to construct test suites using a random black-box approach, but they rarely cover a large percentage of the func- tionality of the system under examination. A black-box approach that covers 100 percent of the functionality is the exhaustive approach, but it is impractical in most cases because too many tests are required. As an example: if we need to design a test suite for a system that has 20 parameters and each parameter has 10 possible values, it would require 1020 tests; however, using a combinatorial interaction testing approach that covers the combinations of all pairs of parameter values, the test suite will require only 155 tests. The number of tests required with combinatorial interaction testing grows logarithmically according to the number of parameters [[11]](#_bookmark18). Empirical studies in software testing have shown that combinato- rial interaction testing is a useful approach Kuhn et al. [[14]](#_bookmark20), Bell [[4]](#_bookmark13). The mathematical objects that support combinatorial interaction

<http://dx.doi.org/10.1016/j.trit.2016.12.006>

2468-2322/© 2017 Chongqing University of Technology. Production and hosting by Elsevier B.V. This is an open access article under the CC BY-NC-ND license ([http://](http://creativecommons.org/licenses/by-nc-nd/4.0/) [creativecommons.org/licenses/by-nc-nd/4.0/](http://creativecommons.org/licenses/by-nc-nd/4.0/)).

testing are Covering Arrays (CA) and Mixed Covering Arrays (MCA). CA and MCA are combinatorial structures that have been used successfully in various areas. The most reported application of CA and MCA is in the design of test suites for software combinatorial interaction testing [[7,8]](#_bookmark15) which is based on the concept that software faults are caused by unexpected combinatorial interactions of certain size between components. Another application is found in the field of parameter fine-tuning of metaheuristic algorithms

[[12,19,21,22]](#_bookmark19).

A CA, denoted by CA(N; t, k, v), is an N × k array, where every that every N × t sub-array contains at least once all possible v*t* t- entry of the array takes values from a set of symbols of size *v*, such tuples of symbols. An MCA is a generalization of a CA where the

alphabets of the columns could have different cardinalities. The test cases are represented by the rows, the parameters are represented by the columns, the parameter values are taken from the set

{0, 1…, v—1} which is called the alphabet, and t is the strength or

combinatorial interaction degree between parameters covered by

the CA. [Fig. 1](#_bookmark1) shows an example of a CA(9; 2,4,3), and an MCA(6; 2,4,3123) is shown in [Fig. 2](#_bookmark2).

The Covering Array Number (CAN) is the minimum N such that for fixed k, v, and t a CA exists. The CAN is denoted by CAN(t,k,v).

The construction of CA with N=CAN(t,k,v) is a challenging problem

whether we use mathematical structures or metaheuristic

algorithms.

When we have non-optimal CA (i.e. a CA with N > CAN(t,k,v)), it usually has many t-tuples that are covered more than once. This fact presents the opportunity to reduce number of rows of CA, given that it may then be possible to identify redundant rows [[18]](#_bookmark24) that

can be removed.

In this paper we introduce a Metaheuristic Post-Optimization (MPO) approach to reduce the size of a CA by exploiting redun- dant elements in CA. MPO is composed of three main components:

a) a redundancy detector (RD); a row reducer (RR); and a missing- combination reducer (MCR) implemented using a simulated annealing algorithm (the metaheuristic component of our approach). MPO was extensively tested using 21, 964 CA (taken from the CA NIST repository). We have improved almost all 21, 964 CA, but the most remarkable result is that MPO has set 349 new upper bounds for these CA.

The remaining of the paper is structured in three sections. In section [2](#_bookmark3) we present in detail MPO approach giving details of the redundancy detector, row reducer and missing-combination reducer components. In section [3](#_bookmark6) we present the results of instantiating the MPO with the whole National Institute of Stan- dards and Technology repository of covering arrays. Finally in



Fig. 1. Transposed matrix of a *CA*(9; 2, 4, 3).



Fig. 2. Transposed matrix of an *MCA*(6; 2, 4, 3123 ).



section [4](#_bookmark8) we present some conclusions.

1. Metaheuristic post-optimization (MPO)

In this section we present implementation details of the MPO approach. We firstly give an overview of the whole process of MPO, secondly, we present details of each of the components RD, RR, MCR.

* 1. *Design of MPO approach*

The design and implementation of MPO approach is briefly described in algorithm 1, where it can be observed that it has three components and two main loops. The inner loop executes the components: *Redundancy Detector* (RD) and *Row Reducer* (RR). After the inner loop is executed, the *Missing-Combinations Reducer* (MCR) runs. When the MCR (implemented with a simulated annealing (SA) algorithm) is not able to make the number of missing combi- nations equal to zero, MPO ends.

MPO (algorithm 1) receives as input A = CA(N; t,k,v) and gives as output B = CA(N’; t, k, v) with N’ ≤ N and no missing t-wise combinations. The function t computes the number of missing t-

wise combinations of the parameter passed to it. t has temporal

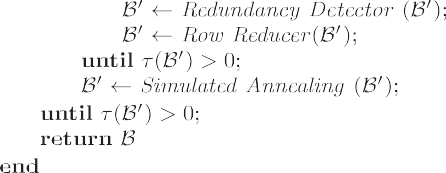
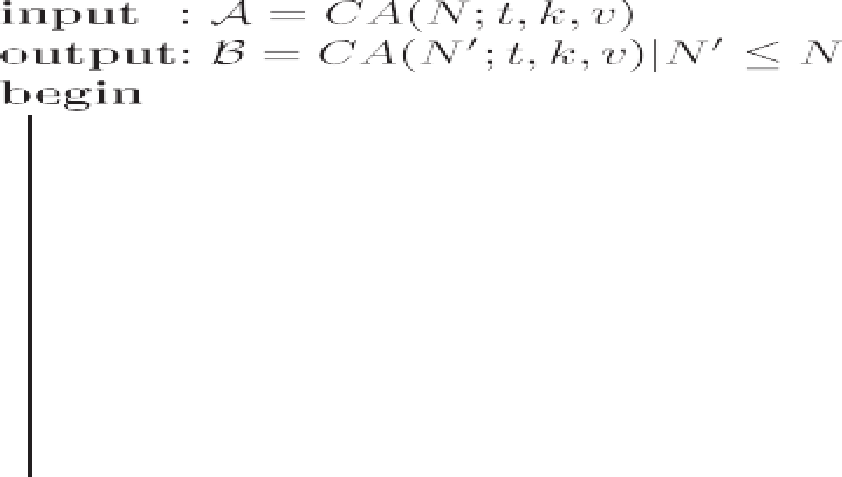
complexity *O* *N* *k* (a more detailed description of how to

*t*

compute the missing interactions for CA was presented by Avila- George et al. [[2]](#_bookmark10).

* 1. *Redundancy detector (RD)*

The goal of the Redundancy Detector (RD) algorithm is to find a





large number of redundant entries in the CA given as input. RD does its job by doing three scans of the input, the first two scans visiting all t-wise combinations of the matrix (each scan with temporal complexity *O* *N* *k* ), the third scan visiting all elements of the

*t*

poral complexity *O*(*Nk*)). The total temporal complexity is matrix, searching for rows that are totally redundant (with tem- *O* 2*N* *k* + *Nk* .

*t*

The purpose of the first scan is to set as ‘Fixed Symbol’ (FS) cells that participate in t-wise combinations covered only once, and as ‘Possible Redundant Cell’ (PRC) all other cells. The second scan works with cells marked as PRC and decides which cells transform to the status of FS, while making sure coverage property (all t-wise combinations must be covered at least once) is satisfied, and number of cells with status of PRC is maximized. The third scan removes all rows in which all elements have status of PRC.

* 1. *Row reducer (RR)*

The Row Reducer algorithm receives as input a CA and works in a greedy manner searching for a row *i* such that its removal mini- mizes missing combinations. In the worst case RR tests all rows of CA, but when a row with no missing t-wise combinations is found RR ends. If this is not the case the row removed is the one that gives the fewest number of missing t-wise combinations.

The logic of the operation of the RR algorithm is simple: replace the FS cells of the *i*-th row in all PRC cells of remaining rows, and then verify number of missing t-wise combinations (after removal of row *i*). RR removes the first row that minimizes missing t-wise combinations. The worst case temporal complexity of the algorithm

is *O* *N*(*N* — 1) *k* for the determination of row to be removed, and *O*(*Nk*) for the removal of the row. Then the total temporal complexity of the RR algorithm is *O* *N*(*N* — 1) *k* + *Nk* .

*t*

*t*

* 1. *Missing-combinations reducer (MCR)*

The MCR component of MPO is in charge of reducing to zero the number of missing t-tuples of the input parameter (a matrix with missing combinations). We decided to implement MCR using a simulated annealing (SA) algorithm given that SA has been applied successfully for solving related problems [[3,6,20,24]](#_bookmark12). The core ele- ments of the SA are: the neighbourhood functions F F ; and the cooling schedule.

We used two neighbourhood functions N F 1 and N F 2. N F 1 searches for a random missing t-tuple and sets one such tuple in every row, selecting the row that gives the fewest number of missing t-wise combinations. N F 2 selects t cells in a row (cells

and rows are selected randomly) then tests every v*t* possible t-tuple in those cells, and selects combination that gives the lower number of missing t-wise combinations. SA uses a mixture of the two neighbourhood functions in such a way that N F 1 is applied with a

1 — *pr*. probability *pr* and consequently N F 2 is applied with a probability

The cooling schedule configuration in SA involves [[1,15]](#_bookmark11): (a) an

initial temperature (*temp*0); (b) a decreasing function to reduce the temperature value; (c) an ending temperature (*tempf* ); and (d) a finite number of iterations of the local search at the same tem- perature (L) (L size of a Markov chain [[5]](#_bookmark14). The parameters of the cooling schedule control the behaviour of the algorithm and therefore affect drastically the quality of the final solution. We selected static geometric cooling schedule controlled by a param- eter a. The parameter L is static during execution of the algorithm [[3,20]](#_bookmark12). Also a parameter called frozen factor (*ff* ) was added to control number of temperature reductions without improvement towards solution, which works as an alternative termination cri- terion that is triggered when search stagnates.

SA algorithm (algorithm 2) is based on definition given by

parameter fine-tuning, and they are: *temp*0 = 1; a = 0.99; Kirkpatrick et al. [[13]](#_bookmark21). Parameter values were selected after a L = Nkv2; *pr* = 0.5; *tempf* = 1 × 10—14; and *ff* = 11.

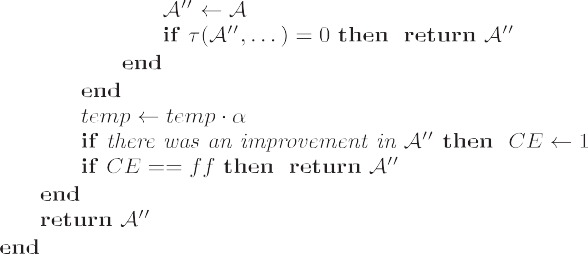
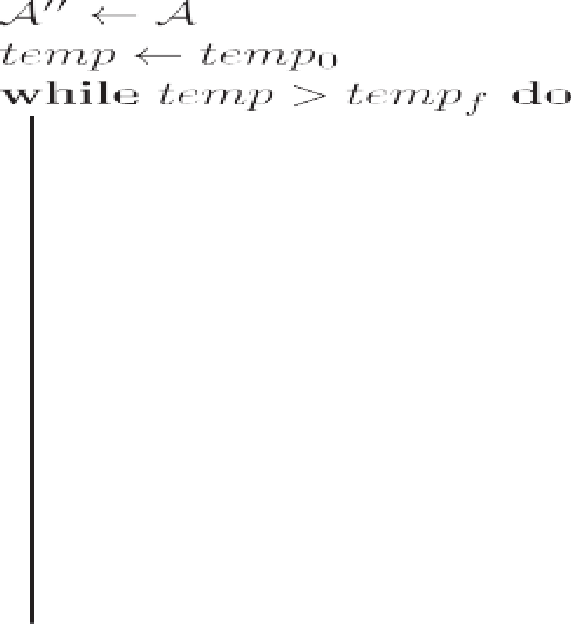
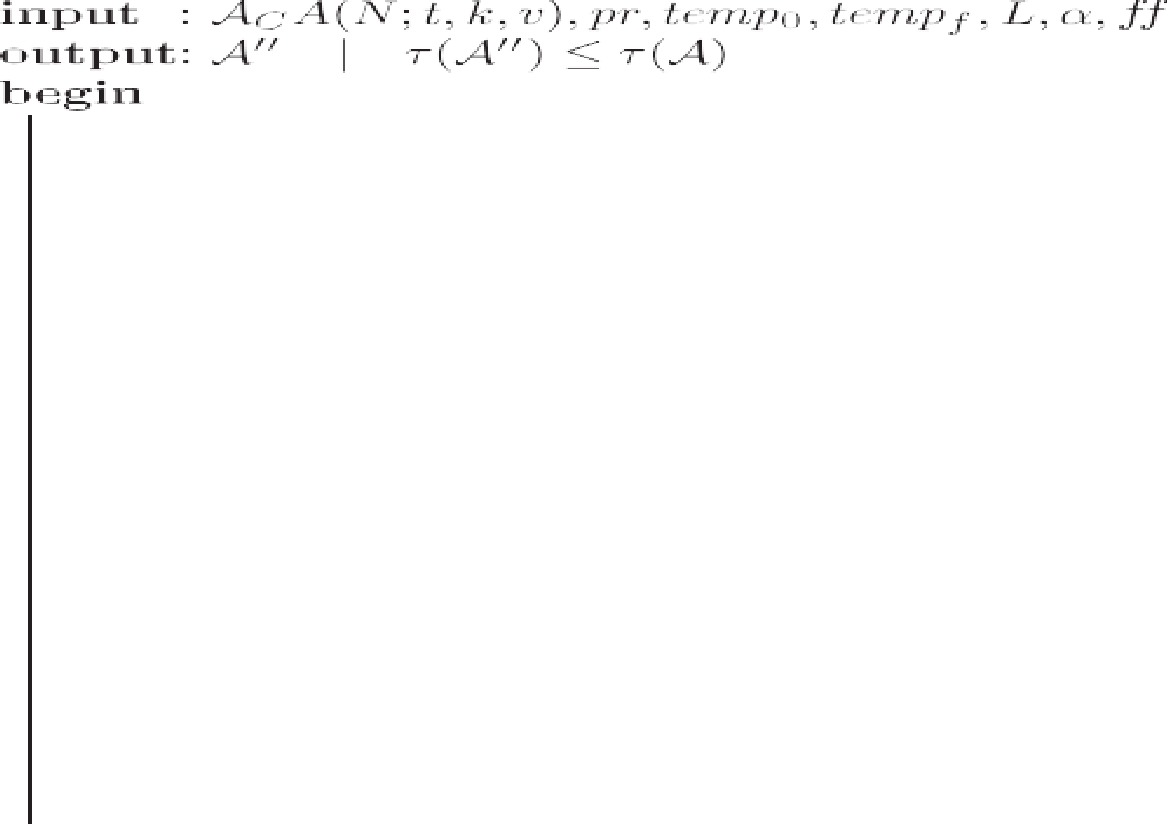




Table 1

Results of MPO algorithm after processing entire repository [[10]](#_bookmark17). Information is organized in triplets containing: average percentage reduction of rows (Y); average time in minutes (G); and number of instances (*I*) (Continues in [Table 2](#_bookmark5)).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| v\*t* | 2 |  |  |  | 3 |  |  |  | 4 |  |  |  | 5 |  | | |
|  | Y | G (m) | I |  | Y | G (m) | I |  | Y | G (m) | I |  | Y | G (m) | I |  |
| 2 | 16.21 | 3.27 | 1998 |  | 2.95 | 670.96 | 1997 |  | 1.17 | 19809.28 | 90 |  | 2.09 | 6650.83 | 186 |  |
| 3 | 6.16 | 9.25 | 1998 |  | 1.42 | 1753.65 | 1997 |  | 1.06 | 21141.30 | 496 |  | 2.35 | 10851.70 | 111 |  |
| 4 | 5.20 | 9.01 | 1998 |  | 0.57 | 2140.89 | 1997 |  | 0.89 | 8161.35 | 304 |  | 2.47 | 16584.87 | 76 |  |
| 5 | 3.82 | 23.95 | 1998 |  | 0.35 | 2280.55 | 1968 |  | 0.99 | 9531.80 | 204 |  | 2.60 | 11895.09 | 56 |  |
| 6 | 3.37 | 64.55 | 1998 |  | 0.37 | 5097.92 | 1303 |  | 1.07 | 13944.75 | 159 |  | 2.93 | 12399.77 | 41 |  |
| avg | 6.95 | 20.47 | 1998 |  | 1.19 | 2236.92 | 1852.4 |  | 1.08 | 21384.18 | 250.6 |  | 2.35 | 16769.35 | 94 |  |

Table 2

(Continued from [Table 1](#_bookmark4)) Results of MPO algorithm after processing entire re- pository [[10]](#_bookmark17). Information is organized in triplets containing: average percentage reduction of rows (Y); average time in minutes (G); and number of instances (*I*).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| v\*t* | 6 |  | | |
|  | Y | G (m) | *I* |  |
| 2 | 3.83 | 4045.90 | 80 |  |
| 3 | 4.08 | 8526.52 | 45 |  |
| 4 | 4.20 | 8715.07 | 30 |  |
| 5 | 5.16 | 9690.55 | 19 |  |
| 6 | e | e | e |  |
| avg | 4.10 | 11647.32 | 43.5 |  |

* 1. *Implementation note*

The proposed algorithms were coded using C language and

compiled with GCC 4.3.5 with -O3 optimization flag, and run in cores of the type AMD® 8435 (2.6 Ghz).

1. Results

To measure the effectiveness of MPO, the NIST repository of CA

v2{2, …, 6} and *t*2{2, …, 6}. For each instance we report: average [[10]](#_bookmark17) was processed. NIST repository of CA consists of 21,964 CA with percentage reduction of rows (Y), average time in minutes (G), and

number of instances (*I*).

[Tables 1 and 2](#_bookmark4) summarize the results of processing NIST re- pository [[10]](#_bookmark17). Information is organized in triplets containing: Y, G, and *I*. It is shown that many instances were optimized, resulting in the construction of 349 state of the art upper bounds for CA by using MPO algorithm. The comparison between the MPO new up- per bounds and the IPOG-F bounds is shown in [Tables 3](#_bookmark7)e[5](#_bookmark7). Results

Table 3

New best upper bounds constructed with MPO algorithm. Part 1 of 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | v3*t*4 |  |  |  |  | v3*t*4 |  |  |  |  | v6*t*4 |  | | | |
| id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |
|  | 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 |  | v6*t*4 |  |  |  |  | 91 | 91 | 11710 | 11630 |  |
|  | 8 | 322 | 974 | 966 |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | 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 |  |

Table 3 (*continued* )

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | v3*t*4 |  |  |  |  | v3*t*4 |  |  |  |  | v6*t*4 |  |  |  |  |
|  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |
|  | 34 | 351 | 993 | 991 |  | 75 | 75 | 10958 | 10882 |  | 118 | 149 | 13634 | 13606 |  |
|  | 35 | 353 | 994 | 987 |  | 76 | 76 | 11012 | 10959 |  | v3*t*5 |  |  |  |  |
|  | 36 | 360 | 999 | 994 |  | 77 | 77 | 11057 | 10992 |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |
|  | 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 |  |

Table 4

New best upper bounds constructed with MPO algorithm. Part 2 of 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | v3*t*5 |  |  |  |  | v3*t*5 |  |  |  |  | v4*t*5 |  | | | |
| id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |
|  | 126 | 42 | 2046 | 2002 |  | 169 | 85 | 2739 | 2700 |  | 210 | 45 | 9227 | 9104 |  |
|  | 127 | 43 | 2070 | 2022 |  | 170 | 86 | 2749 | 2706 |  | 211 | 46 | 9320 | 9176 |  |
|  | 128 | 44 | 2091 | 2050 |  | 171 | 87 | 2762 | 2728 |  | 212 | 47 | 9406 | 9262 |  |
|  | 129 | 45 | 2112 | 2071 |  | 172 | 88 | 2770 | 2736 |  | 213 | 48 | 9501 | 9357 |  |
|  | 130 | 46 | 2130 | 2086 |  | 173 | 89 | 2783 | 2747 |  | 214 | 49 | 9588 | 9453 |  |
|  | 131 | 47 | 2150 | 2112 |  | 174 | 90 | 2792 | 2761 |  | 215 | 50 | 9673 | 9520 |  |
|  | 132 | 48 | 2174 | 2134 |  | 175 | 91 | 2805 | 2772 |  | 216 | 51 | 9755 | 9621 |  |
|  | 133 | 49 | 2191 | 2154 |  | 176 | 92 | 2815 | 2783 |  | 217 | 52 | 9835 | 9682 |  |
|  | 134 | 50 | 2213 | 2182 |  | 177 | 93 | 2825 | 2798 |  | 218 | 53 | 9922 | 9769 |  |
|  | 135 | 51 | 2231 | 2198 |  | 178 | 94 | 2836 | 2797 |  | 219 | 54 | 9998 | 9849 |  |
|  | 136 | 52 | 2251 | 2217 |  | 179 | 95 | 2847 | 2813 |  | 220 | 55 | 10079 | 9927 |  |
|  | 137 | 53 | 2269 | 2232 |  | 180 | 96 | 2857 | 2819 |  | 221 | 56 | 10155 | 10007 |  |
|  | 138 | 54 | 2289 | 2246 |  | 181 | 97 | 2868 | 2834 |  | 222 | 57 | 10232 | 10082 |  |
|  | 139 | 55 | 2309 | 2277 |  | 182 | 98 | 2877 | 2838 |  | 223 | 59 | 10379 | 10231 |  |
|  | 140 | 56 | 2327 | 2281 |  | 183 | 99 | 2885 | 2854 |  | 224 | 60 | 10454 | 10302 |  |
|  | 141 | 57 | 2342 | 2304 |  | 184 | 100 | 2895 | 2862 |  | 225 | 62 | 10590 | 10442 |  |
|  | 142 | 58 | 2358 | 2316 |  | 185 | 101 | 2909 | 2871 |  | 226 | 63 | 10650 | 10509 |  |
|  | 143 | 59 | 2373 | 2335 |  | 186 | 102 | 2920 | 2878 |  | v5*t*5 |  |  |  |  |
|  | 144 | 60 | 2394 | 2347 |  | 187 | 103 | 2928 | 2898 |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |
|  | 145 | 61 | 2408 | 2370 |  | 188 | 104 | 2938 | 2901 |  | 227 | 21 | 18779 | 18260 |  |
|  | 146 | 62 | 2425 | 2387 |  | 189 | 105 | 2945 | 2908 |  | 228 | 22 | 114775 | 111818 |  |
|  | 147 | 63 | 2440 | 2397 |  | 190 | 107 | 2962 | 2928 |  | 229 | 23 | 118587 | 115802 |  |
|  | 148 | 64 | 2451 | 2413 |  | v4*t*5 |  |  |  |  | 230 | 24 | 122201 | 119500 |  |
|  | 149 | 65 | 2468 | 2439 |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | 231 | 25 | 125683 | 123108 |  |
|  | 150 | 66 | 2482 | 2447 |  | 191 | 26 | 6957 | 6775 |  | v6*t*5 |  |  |  |  |
|  | 151 | 67 | 2498 | 2459 |  | 192 | 27 | 7116 | 6937 |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |
|  | 152 | 68 | 2512 | 2477 |  | 193 | 28 | 7267 | 7088 |  | 232 | 17 | 40334 | 38976 |  |
|  | 153 | 69 | 2527 | 2487 |  | 194 | 29 | 7414 | 7247 |  | 233 | 18 | 42102 | 40820 |  |
|  | 154 | 70 | 2542 | 2504 |  | 195 | 30 | 7555 | 7379 |  | 234 | 19 | 43833 | 42554 |  |
|  | 155 | 71 | 2555 | 2518 |  | 196 | 31 | 7691 | 7527 |  | 235 | 20 | 45425 | 44224 |  |
|  | 156 | 72 | 2573 | 2531 |  | 197 | 32 | 7816 | 7649 |  | 236 | 21 | 46970 | 45784 |  |
|  | 157 | 73 | 2584 | 2546 |  | 198 | 33 | 7939 | 7782 |  | 237 | 22 | 48479 | 47352 |  |
|  | 158 | 74 | 2597 | 2564 |  | 199 | 34 | 8064 | 7907 |  | 238 | 23 | 49924 | 48838 |  |
|  | 159 | 75 | 2609 | 2588 |  | 200 | 35 | 8183 | 8023 |  | 239 | 24 | 51287 | 50180 |  |
|  | 160 | 76 | 2625 | 2593 |  | 201 | 36 | 8301 | 8137 |  | 240 | 25 | 52604 | 51505 |  |
|  | 161 | 77 | 2639 | 2608 |  | 202 | 37 | 8420 | 8259 |  | 241 | 26 | 53850 | 52814 |  |
|  | 162 | 78 | 2648 | 2615 |  | 203 | 38 | 8530 | 8376 |  | 242 | 27 | 55069 | 54032 |  |
|  | 163 | 79 | 2661 | 2625 |  | 204 | 39 | 8629 | 8477 |  | 243 | 28 | 56225 | 55275 |  |
|  | 164 | 80 | 2673 | 2635 |  | 205 | 40 | 8737 | 8583 |  | 244 | 29 | 57363 | 56353 |  |
|  | 165 | 81 | 2686 | 2651 |  | 206 | 41 | 8847 | 8693 |  | 245 | 30 | 58468 | 57503 |  |
|  | 166 | 82 | 2700 | 2668 |  | 207 | 42 | 8945 | 8791 |  | 246 | 31 | 59529 | 58576 |  |
|  | 167 | 83 | 2710 | 2677 |  | 208 | 43 | 9035 | 8890 |  | 247 | 32 | 60570 | 59612 |  |
|  | 168 | 84 | 2725 | 2684 |  | 209 | 44 | 9137 | 8979 |  | 248 | 33 | 61562 | 60608 |  |

are shown in [Figs. 3](#_bookmark9)e[7](#_bookmark9) where instances are grouped by combina- torial interaction coverage degree (*t*).

1. Conclusions

In this paper we have presented a Metaheuristic Post-

Table 5

New best upper bounds constructed with MPO algorithm. Part 3 of 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | v6*t*5 |  |  |  |  | v2*t*6 |  |  |  |  | v4*t*6 |  |  |  |  |
|  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |
|  | 249 | 34 | 62527 | 61612 |  | 290 | 79 | 782 | 770 |  | 329 | 26 | 33369 | 32513 |  |
|  | 250 | 35 | 63471 | 62557 |  | 291 | 80 | 785 | 771 |  | 330 | 27 | 34187 | 33356 |  |
|  | 251 | 36 | 64399 | 63519 |  | 292 | 81 | 791 | 772 |  | 331 | 28 | 35006 | 34198 |  |
|  | v2*t*6 |  |  |  |  | 293 | 82 | 794 | 778 |  | 332 | 29 | 35791 | 34971 |  |
|  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | 294 | 83 | 796 | 787 |  | 333 | 30 | 36570 | 35778 |  |
|  | 252 | 41 | 572 | 547 |  | 295 | 84 | 800 | 784 |  | 334 | 31 | 37305 | 36515 |  |
|  | 253 | 42 | 579 | 550 |  | 296 | 85 | 804 | 790 |  | 335 | 32 | 38015 | 37255 |  |
|  | 254 | 43 | 590 | 565 |  | 297 | 86 | 809 | 799 |  | v5*t*6 |  |  |  |  |
|  | 255 | 44 | 594 | 565 |  | v3*t*6 |  |  |  |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |
|  | 256 | 45 | 603 | 578 |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  | 336 | 11 | 56615 | 52471 |  |
|  | 257 | 46 | 611 | 588 |  | 298 | 26 | 5709 | 5544 |  | 337 | 12 | 63620 | 59622 |  |
|  | 258 | 47 | 617 | 590 |  | 299 | 27 | 5853 | 5667 |  | 338 | 13 | 70190 | 66275 |  |
|  | 259 | 48 | 625 | 600 |  | 300 | 28 | 6003 | 5827 |  | 339 | 14 | 76390 | 72680 |  |
|  | 260 | 49 | 630 | 604 |  | 301 | 29 | 6150 | 5969 |  | 340 | 15 | 82139 | 78480 |  |
|  | 261 | 50 | 636 | 612 |  | 302 | 30 | 6281 | 6103 |  | 341 | 16 | 87559 | 84102 |  |
|  | 262 | 51 | 643 | 620 |  | 303 | 31 | 6413 | 6245 |  | 342 | 18 | 97605 | 94263 |  |
|  | 263 | 52 | 650 | 630 |  | 304 | 32 | 6535 | 6348 |  | 343 | 19 | 102208 | 98994 |  |
|  | 264 | 53 | 656 | 630 |  | 305 | 33 | 6656 | 6461 |  | 344 | 20 | 106642 | 103514 |  |
|  | 265 | 54 | 662 | 640 |  | 306 | 34 | 6772 | 6583 |  | 345 | 21 | 110842 | 107773 |  |
|  | 266 | 55 | 667 | 645 |  | 307 | 35 | 6877 | 6715 |  | 346 | 22 | 114775 | 111818 |  |
|  | 267 | 56 | 672 | 650 |  | 308 | 36 | 6989 | 6832 |  | 347 | 23 | 118587 | 115802 |  |
|  | 268 | 57 | 677 | 663 |  | 309 | 37 | 7092 | 6932 |  | 348 | 24 | 122201 | 119500 |  |
|  | 269 | 58 | 683 | 665 |  | 310 | 38 | 7194 | 7036 |  | 349 | 25 | 125683 | 123108 |  |
|  | 270 | 59 | 689 | 665 |  | 311 | 39 | 7293 | 7131 |  |  |  |  |  |  |
|  | 271 | 60 | 695 | 675 |  | 312 | 40 | 7391 | 7233 |  |  |  |  |  |  |
|  | 272 | 61 | 699 | 675 |  | 313 | 41 | 7490 | 7315 |  |  |  |  |  |  |
|  | 273 | 62 | 703 | 685 |  | 314 | 42 | 7574 | 7411 |  |  |  |  |  |  |
|  | 274 | 63 | 709 | 685 |  | 315 | 43 | 7672 | 7506 |  |  |  |  |  |  |
|  | 275 | 64 | 715 | 695 |  | 316 | 44 | 7757 | 7600 |  |  |  |  |  |  |
|  | 276 | 65 | 721 | 695 |  | 317 | 45 | 7845 | 7702 |  |  |  |  |  |  |
|  | 277 | 66 | 725 | 705 |  | 318 | 46 | 7938 | 7766 |  |  |  |  |  |  |
|  | 278 | 67 | 728 | 705 |  | 319 | 47 | 8013 | 7856 |  |  |  |  |  |  |
|  | 279 | 68 | 732 | 710 |  | 320 | 50 | 8256 | 8108 |  |  |  |  |  |  |
|  | 280 | 69 | 738 | 724 |  | 321 | 51 | 8333 | 8179 |  |  |  |  |  |  |
|  | 281 | 70 | 743 | 729 |  | v4*t*6 |  |  |  |  |  |  |  |  |  |
|  | 282 | 71 | 747 | 734 |  | id | *k* | IPOG-F Forbes et al. [[9]](#_bookmark16) | MPO |  |  |  |  |  |  |
|  | 283 | 72 | 751 | 736 |  | 322 | 19 | 26392 | 25430 |  |  |  |  |  |  |
|  | 284 | 73 | 755 | 743 |  | 323 | 20 | 27534 | 26564 |  |  |  |  |  |  |
|  | 285 | 74 | 761 | 749 |  | 324 | 21 | 28625 | 27676 |  |  |  |  |  |  |
|  | 286 | 75 | 766 | 751 |  | 325 | 22 | 29640 | 28735 |  |  |  |  |  |  |
|  | 287 | 76 | 770 | 755 |  | 326 | 23 | 30636 | 29720 |  |  |  |  |  |  |
|  | 288 | 77 | 773 | 758 |  | 327 | 24 | 31591 | 30724 |  |  |  |  |  |  |
|  | 289 | 78 | 777 | 760 |  | 328 | 25 | 32501 | 31654 |  |  |  |  |  |  |

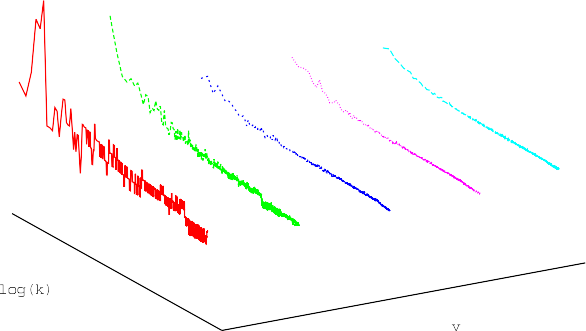
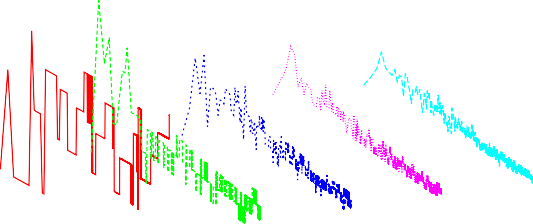


Fig. 3. Results of MPO after processing instances of repository [[10]](#_bookmark17) with *t* = 2. (—D = *N* — *N*').

Fig. 4. Results of MPO after processing instances of repository [[10]](#_bookmark17) with *t* = 3. (—D = *N* — *N*').

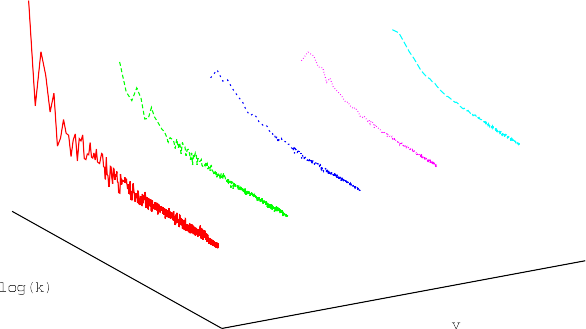


Fig. 5. Results of MPO after processing instances of repository [[10]](#_bookmark17) with *t* = 4. (—D = *N* — *N*').

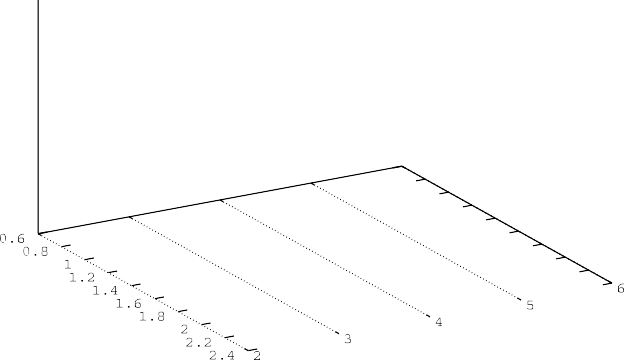


Fig. 6. Results of MPO after processing instances of repository [[10]](#_bookmark17) with *t* = 5. (—D = *N* — *N*').

Fig. 7. Results of MPO after processing instances of repository [[10]](#_bookmark17) with *t* = 6. (—D = *N* — *N*').

Optimization (MPO) approach to reduce the cardinality of Covering Arrays. MPO was implemented using three components: a

redundancy detector, a row reducer, and a missing-combination reducer. The redundancy detector has the mission of detecting el- ements in CA that could be changed without affecting degree of coverage of CA. The row reducer takes advantage of redundant el- ements of CA to reduce number of rows. When the removal of a row produces missing combinations, then control is given to the missing-combination reducer. The missing-combination reducer is implemented with simulated annealing (the metaheuristic component of MPO) and tries to make the number of missing combinations equal to zero. Even though all three components are key to the success of MPO, we believe metaheuristic component is the most important part of MPO, given that through its use it is possible to reduce iteratively number of rows of CA.

We have conducted big-scale experimentation through instan- tiation of MPO with the whole NIST repository of CA. NIST re- pository consists of 21, 964 CA, and while improving almost all CA in repository, the most remarkable result is that we have set 349 new upper bounds for these CA. All CA are available at [http://www.](http://www.tamps.cinvestav.mx/~oc/) [tamps.cinvestav.mx/~oc/](http://www.tamps.cinvestav.mx/~oc/).

Disclaimer

Any mention of commercial products in this paper is for infor- mation only; it does not imply recommendation or endorsement by NIST.

Acknowledgements

The authors acknowledge ”Xiuhcoatl”-CGSTIC of CINVESTAV and ABACUS-CINVESTAV, CONACYT grant EDOMEX-2011-COI-

165873, for providing access to high performance computing. This research was partially funded by the project: CONACyT 238469 -

Me´todos Exactos para Construir Covering Arrays. Part of this paper

was written during first author's research stay at NIST in 2014e2015.

References

1. [E. Aarts, J. Lenstra, Local Search in Combinatorial Optimization, Princeton Univ](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref1) [Pr, 2003. ISBN: 0691115222](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref1).
2. [H. Avila-George, J. Torres-Jimenez, L.V.H. Gonzalez-Hernandez, Metaheuristic](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref2) [approach for constructing functional test-suites, IET Softw. 7 (2) (2013)](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref2) [104](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref2)e[117](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref2).
3. [H. Avila-George, J. Torres-Jimenez, V. Hern](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3)a´[ndez, L. Gonzalez-Hernandez,](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3)

[Simulated annealing for constructing mixed covering arrays, in: Proceedings](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3) [of the 9th International Symposium on Distributed Computing and Arti](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3)fi[cial](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3) [Intelligence - DCAI. Vol. 151 of Advances in Intelligent and Soft Computing.](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3) [Springer Berlin/Heidelberg, Salamanca, Spain, from 28th to 30th March, 2012,](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3)

[pp. 657](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3)e[664](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref3).

1. [K. Bell, Optimizing Effectiveness and Ef](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref4)fi[ciency of Software Testing: a Hybrid](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref4) [Approach. Ph.D. Thesis, North Carolina State University, 2006](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref4).
2. [D. Bertsimas, J. Tsitsiklis, Simulated annealing, Stat. Sci. 8 (1) (1993) 10](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref5)e[15](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref5).
3. [M. Cohen, C. Colbourn, A. Ling, Augmenting simulated annealing to build](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref6) [interaction test suites, in: Software Reliability Engineering, 2003. ISSRE 2003.](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref6) [14th International Symposium on, Nov. 2003, pp. 394](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref6)e[405](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref6).
4. [M. Cohen, P. Gibbons, W. Mugridge, C. Colbourn, Constructing test suites for](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref7) [interaction testing, in: Proceedings of the 25th International Conference on](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref7) [Software Engineering. Portland, Oregon, USA, May 2003, pp. 38](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref7)e[48](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref7).
5. [M.B. Cohen, Designing Test Suites for Software Interaction Testing. Ph.D.](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref8) [Thesis, University of Auckland, 2004](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref8).
6. [M. Forbes, J. Lawrence, Y. Lei, R. Kacker, D. Kuhn, Refining the in-parameter-](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref9) [order strategy for constructing covering arrays, J. Res. Natl. Inst. Stand.](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref9) [Technol. 113 (5) (2008) 287](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref9)e[297](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref9).
7. M. Forbes, J. Lawrence, Y. Lei, R. Kacker, D. Kuhn, Repository of CAs. Last Ac- cess September, 2014. URL, [http://math.nist.gov/coveringarrays/ipof/ipof-](http://math.nist.gov/coveringarrays/ipof/ipof-results.html) [results.html](http://math.nist.gov/coveringarrays/ipof/ipof-results.html).
8. [A. Hedayat, N. Sloane, J. Stufken, Orthogonal Arrays: Theory and Applications,](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref11) [Springer Verlag, 1999. ISBN: 0-387-98766-5](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref11).
9. [A. Jose-Garcia, H. Romero-Monsivais, C. Hernandez-Morales, A. Rodriguez-](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref12) [Cristerna, I. Rivera-Islas, J. Torres-Jimenez, A simulated annealing algorithm](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref12) [for the problem of minimal addition chains, in: Progress in Arti](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref12)fi[cial Intelli-](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref12) [gence. Vol. 7026 of Lecture Notes in Computer Science. Springer Berlin/Hei-](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref12) [delberg, 2011, pp. 311](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref12)e[325](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref12).
10. [S. Kirkpatrick, C. Gelatt Jr., M. Vecchi, Optimization by simulated annealing,](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref13) [science 220 (4598) (1983) 671](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref13)e[680](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref13).
11. [D. Kuhn, D. Wallace, A. Gallo Jr., Software fault interactions and implications](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref14) [for software testing, IEEE Trans. Softw. Eng. 30 (6) (2004) 418](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref14)e[421](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref14).
12. [J. Lam, An Ef](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref15)fi[cient Simulated Annealing Schedule. Ph.D. Thesis, Yale Univer-](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref15) [sity New Haven CT, 1988](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref15).
13. J. Lions, et al., Ariane 5 Flight 501 Failure, Accesed 2014-09. Tech. Rep, 09

2014. <http://www.di.unito.it/damiani/ariane5rep.htm>.

1. [G. Myers, C. Sandler, T. Badgett, T. Thomas, The Art of Software Testing, second](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref17) [ed., John Wiley](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref17) & [Sons Inc, 2004. ISBN: 0471469122](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref17).
2. [P. Nayeri, C. Colbourn, G. Konjevod, Randomized postoptimization of covering](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref18) [arrays, in: Combinatorial Algorithms. Vol. 5874 of Lecture Notes in Computer](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref18) [Science. Springer Berlin/Heidelberg, 2009, pp. 408](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref18)e[419](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref18).
3. [N. Rangel-Valdez, J. Torres-Jim](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19)e´[nez, J. Bracho-R](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19)ı[os, P. Quiz-Ramos, Problem](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19)

[and algorithm](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19) fi[ne-tuning - a case of study using bridge club and simulated](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19) [annealing, in: A. Dourado, A.C. Rosa, K. Madani (Eds.), IJCCI. IJCCI 2009-Pro-](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19) [ceedings of the International Joint Conference on Computational Intelligence,](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19) [Funchal, Madeira, Portugal, October 5-7, 2009, INSTICC Press, 2009, pp.](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19)

[302](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19)e[305](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref19).

1. [A. Rodriguez-Cristerna, J. Torres-Jimenez, A simulated annealing with variable](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref20) [neighborhood search approach to construct mixed covering arrays, Electron.](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref20) [Notes Discrete Math. 39 (0) (2012) 249](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref20)e[256 (eURO Mini Conference)](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref20).
2. [A. Rodriguez-Cristerna, J. Torres-Jimenez, A genetic algorithm for the problem](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref21) [of minimal brauer chains for large exponents, in: Soft Computing Applications](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref21) [in Optimization, Control, and Recognition, Springer, 2013, pp. 27](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref21)e[51](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref21).
3. [A. Rodriguez-Cristerna, J. Torres-Jimnez, I. Rivera-Islas, C. Hernandez-Morales,](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref22)

[H. Romero-Monsivais, A. Jose-Garcia, A mutation-selection algorithm for the](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref22) [problem of minimum brauer chains, in: I. Batyrshin, G. Sidorov (Eds.), Ad-](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref22) [vances in Soft Computing. Vol. 7095 of Lecture Notes in Computer Science,](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref22) [Springer Berlin/Heidelberg, 2011, pp. 107](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref22)e[118](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref22).

1. [G. Tassey, The Economic Impacts of Inadequate Infrastructure for Software](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref23) [Testing. Tech. Rep, Research Triangle Institute, National Institute of Standards](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref23) [and Technology, U.S. Deparment of Commerce, 05 2002](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref23).
2. [J. Torres-Jimenez, E. Rodriguez-Tello, New bounds for binary covering arrays](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref24) [using simulated annealing, Inf. Sci. 185 (1) (Feb. 2012) 137](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref24)e[152](http://refhub.elsevier.com/S2468-2322(16)30031-2/sref24).