Generating Combinatorial Test Cases by Efficient SAT Encodings Suitable for CDCL SAT Solvers

Mutsunori Banbara[1], Haruki Matsunaka[2],
Naoyuki Tamura[1], and Katsumi Inoue[3]

[1] Information Science and Technology Center, Kobe University, Japan
{banbara,tamura}@kobe-u.ac.jp
[2] Graduate School of System Informatics, Kobe University, Japan
matsunaka@stu.kobe-u.ac.jp
[3] National Institute of Informatics, Japan
ki@nii.ac.jp

Abstract. Generating test cases for combinatorial testing is to find a covering array in Combinatorial Designs. In this paper, we consider the problem of finding optimal covering arrays by SAT encoding. We present two encodings suitable for modern CDCL SAT solvers. One is based on the order encoding that is efficient in the sense that unit propagation achieves the bounds consistency in CSPs. Another one is based on a combination of the order encoding and Hnich's encoding. CDCL SAT solvers have an important role in the latest SAT technology. The effective use of them is essential for enhancing efficiency. In our experiments, we found solutions that can be competitive with the previously known results for the arrays of strength two to six with small to moderate size of components and symbols. Moreover, we succeeded either in proving the optimality of known bounds or in improving known lower bounds for some arrays.

1 Introduction

Propositional Satisfiability (SAT) is fundamental in solving many application problems in Artificial Intelligence and Computer Science: logic synthesis, planning, theorem proving, hardware/software verification, and Constraint Satisfaction Problems (CSPs). Remarkable improvements in the efficiency of SAT solvers have been made over the last decade. Such improvements encourage researchers to solve CSPs by encoding them into SAT (i.e. "SAT encoding"). A number of SAT encoding methods have been therefore proposed: *direct encoding* [1, 2], *support encoding* [3, 4], *multivalued encoding* [5], *log encoding* [6, 7], *order encoding* [8, 9], and *log-support encoding* [10].

Hardware/software testing plays an important role in enhancing the reliability of products. However, it has become one of the most expensive tasks in the product development process in recent years. *Combinatorial testing* is an effective black-box testing method to detect elusive failures of hardware/software. The basic idea is based on the observations that most failures are caused by

C. Fermüller and A. Voronkov (Eds.): LPAR-17, LNCS 6397, pp. 112–126, 2010.

interactions of multiple components. The number of test cases is therefore much smaller than exhaustive testing. Generating test cases for combinatorial testing is to find a *Covering Array* (CA) in Combinatorial Designs. A covering array provides a set of test cases, where each row of the array can be regarded as a set of component symbols for an individual test case.

Since the usefulness of covering arrays for combinatorial testing was shown, there has been a great deal of work for finding covering arrays with as small number of rows as possible [11–21]. From the perspective of SAT, Hnich *et al.* proposed a SAT encoding designed for incomplete SAT solvers based on stochastic local search algorithms [20, 21]. A non-clausal encoding on incomplete SAT solvers was studied in [22]. Myra B. Cohen *et al.* proposed algorithms that synergistically integrate SAT with greedy methods [23]. However, there is very little literature on SAT encoding of this problem suitable for modern *Conflict-Driven Clause Learning* (CDCL) SAT solvers.

In this paper, we consider the problem of finding optimal covering arrays by SAT encoding. We present two encodings suitable for modern CDCL SAT solvers. One is based on the order encoding which an award-winning SAT-based CSP solver Sugar [1] adopted. The order encoding is efficient in the sense that unit propagation keeps the *bounds consistency* in CSPs [24]. Another one is based on a combination of the order encoding and Hnich's encoding. It is designed to reduce the number of clauses required, compared with the order encoding. CDCL SAT solvers have an important role in the latest SAT technology. The effective use of them is essential for enhancing efficiency. Our two encodings are based on the idea of order encoding, and the practical effectiveness of a combination of the order encoding and CDCL SAT solvers has been shown by the fact that Sugar became an award-winning system in the Fourth International CSP Solver Competition for the past two years.

In our experiments, we found solutions that can be competitive with the previously known results obtained by orthogonal arrays, theoretical works, and several computational search methods for the arrays of strength two to six with small to moderate size of components and symbols. Moreover, we succeeded either in proving the optimality of known bounds or in improving known lower bounds for some arrays. Our results include a solution to an open problem listed in Handbook of Satisfiability [25] published in 2009.

The rest of this paper is organized as follows. Section 2 presents the basic definitions of covering arrays and related work, especially Hnich's constraint programming model. Section 3 presents Hnich's SAT encoding. Our encodings are presented in Section 4 and 5. Section 6 shows comparison results of different encodings. Section 7 shows experimental results of finding optimal covering arrays. The paper is concluded in Section 8.

2 Covering Arrays and Related Work

The following definitions are based on Colbourn [26].

[1] `http://bach.istc.kobe-u.ac.jp/sugar/`

1	2	3	4	5
0	**0**	0	0	0
0	**1**	1	2	2
0	**2**	2	1	1
1	**0**	1	1	1
1	**1**	0	1	2
1	**2**	1	2	0
1	2	2	0	2
2	**0**	2	2	2
2	**1**	1	0	1
2	1	2	1	0
2	**2**	0	2	1

Fig. 1. An optimal covering array of $CA(11; 2, 5, 3)$

	(1,2)	(1,3)	(1,4)	(1,5)	(2,3)	(2,4)	(2,5)	(3,4)	(3,5)	(4,5)
0	0	0	0	0	0	0	0	0	0	0
1	1	2	2	4	5	5	5	5	8	
2	2	1	1	8	7	7	7	7	4	
3	4	4	4	1	1	1	4	4	4	
4	3	4	5	3	4	5	1	2	5	
5	4	5	3	7	8	6	5	3	6	
6	8	8	8	2	2	2	8	8	8	
7	7	6	7	4	3	4	3	4	1	
8	6	8	7	6	8	7	2	1	7	

Fig. 2. An alternative matrix of $CA(11; 2, 5, 3)$ shown in Fig. 1

Definition 1. A *covering array* $CA(b; t, k, g)$ is a $b \times k$ array (b rows and k columns) such that every $b \times t$ sub-array contains all t-tuples from g symbols at least once. The parameter t is the *strength* of the array, k is the number of components, and g is the number of symbols for each component.

Definition 2. The *covering array number* $CAN(t, k, g)$ is the smallest b for which a $CA(b; t, k, g)$ exists.

Definition 3. A covering array $CA(b; t, k, g)$ is *optimal* if $CAN(t, k, g) = b$.

For example, Fig. 1 shows an example of $CA(11; 2, 5, 3)$, a covering array of strength two ($t = 2$) with five components ($k = 5$) having three symbols ($g = 3$) each. It is an optimal covering array which has eleven rows ($b = 11$). We highlight the different 2-tuples from three symbols in the first 11×2 sub-array to show all possible 2-tuples occur at least once. This property holds for all sub-arrays.

In the case of $t = g = 2$, finding optimal covering arrays was solved in 1970s (see [27] for details). However, in general, the problem of finding optimal covering arrays is NP-complete [28]. To determine $CAN(t, k, g)$, the following basic properties are useful and will be used on Table 4 in Section 7.

Theorem 1 (Chateauneuf and Kreher [12] and Zhang [25]).

1. $g^t \leq CAN(t, k, g) \leq g^k$
2. $CAN(t, k-1, g) \leq CAN(t, k, g)$
3. $CAN(t, k, g-1) \leq CAN(t, k, g)$
4. $g \cdot CAN(t-1, k-1, g) \leq CAN(t, k, g)$

Table. 1 shows a list of current bounds on the covering array number $CAN(3, k, g)$ with small to moderate size of k and g. This table is based on Colbourn's CA tables [29] and the papers [12, 25]. The (k, g)-entry is either n when $CAN(t, k, g) = n$ or (n_ℓ, n_u) when $n_\ell \leq CAN(t, k, g) \leq n_u$.

In this paper, we define two kinds of problems to make our approach more understandable. For a given tuple $\langle t, k, g, b \rangle$, *CA decision problem* is the problem to decide whether a $CA(b; t, k, g)$ exists or not, and find it if exists. For a

Table 1. Current bounds on $CAN(3, k, g)$

$k\backslash g$	2	3	4	5	6	7	8
4	8	27	64	125	216	343	512
5	10	28,33	64	125	222,240	343	512
6	12	33	64	125	222,258	343	512
7	12	36,40	76,88	125,180	222,293	343	512
8	12	36,42	76,88	145,185	222,304	343	512
9	12	36,45	76,112	145,185	234,379	343,472	512
10	12	36,45	76,112	145,185	234,393	364,479	512
11	12	36,45	76,121	145,225	234,463	364,637	536,960
12	14,15	36,45	76,121	145,225	234,463	364,637	536,960
13	14,16	36,51	76,124	145,245	234,503	364,637	536,960
14	14,16	36,51	76,124	145,245	234,503	364,637	536,960
15	14,17	36,57	76,124	145,245	234,514	364,637	536,960
16	14,17	36,60	76,124	145,245	234,514	364,637	536,960

given tuple $\langle t, k, g\rangle$, *CA optimization problem* is the problem to find an optimal covering array $CA(b; t, k, g)$.

Hnich *et al.* proposed three different CSP representations for solving the CA decision problems: *naïve matrix model*, *alternative matrix model*, and *integrated matrix model* [20, 21]. In these models, the fact a CA exists is equivalent to its CSP representation being satisfiable. It can be easily applied to the CA optimization problems. The CSPs of CA decision problems with varying the value of b contain both satisfiable and unsatisfiable problems, and the optimal solution exists on the boundary. The integrated matrix model consists of two matrices and two kinds of constraints.

Original matrix is a $b\times k$ matrix of integer variables $x_{r,i}$ $(1 \le r \le b, 1 \le i \le k)$. The domain of each variable is $\{0, 1, 2, \ldots, g-1\}$. This matrix identifies a covering array itself.

Alternative matrix is a $b \times \binom{k}{t}$ matrix of integer variables $y_{r,i'} (1 \le r \le b, 1 \le i' \le \binom{k}{t})$. Each column expresses one of the possible t-tuple of columns in the original matrix. Each variable expresses a t-tuple of variables in the original matrix (called *compound variables*). The domain of each variable is $\{0, 1, 2, \ldots, g^t - 1\}$.

Coverage constraints specify the condition that all possible t-tuples from g symbols must occur at least once for all $b \times t$ sub-arrays. In the alternative matrix, the coverage constraints can be expressed by using *Global Cardinality Constraints* (GCC) [30]. That is, for every column (i.e. every sub-array in the original matrix), one GCC is enforced to ensure that every number in the range 0 to $g^t - 1$ occurs at least once and at most $b - g^t + 1$ times.

Channelling constraints associate each compound variable in the alternative matrix with the t-tuples of corresponding variables in the original matrix. Let $1 \le c_1^{i'} \le c_2^{i'} \le \cdots \le c_t^{i'} \le k$ be t distinct columns in the original matrix, which correspond to the column i' in the alternative matrix. The channelling constraints can be intensionally expressed as $y_{r,i'} = \sum_{\ell=1}^{t} g^{t-\ell} x_{r,c_\ell^{i'}}$.

For example, the channelling constraints of $CA(11; 2, 5, 3)$ can be intensionally expressed as $y_{r,(i,j)} = 3x_{r,i} + x_{r,j}$, or extensionally as follows:

$$(y_{r,(i,j)}, x_{r,i}, x_{r,j}) \in \{(0,0,0), (1,0,1), (2,0,2), (3,1,0), \\ (4,1,1), (5,1,2), (6,2,0), (7,2,1), (8,2,2)\}.$$

Fig. 2 shows an alternative matrix that corresponds to the array of $CA(11; 2, 5, 3)$ shown in Fig. 1.

The integrated matrix model uses the idea of compound variables, and the coverage constraints can be elegantly expressed by the global constraints. In this paper, we solve the CA optimization problems by encoding this constraint model into SAT. Before showing our encodings, we present an existing encoding.

3 Hnich's SAT Encoding

Since we are working in SAT encoding, the integrated matrix model needs to be expressed as a propositional formula in Conjunctive Normal Form (CNF).

In Hnich's SAT encoding [20, 21], the propositional variables for $CA(b; t, k, g)$ are $p\,(x_{r,i} = v)$ and $p\,(y_{r,i'} = w)$ with $1 \le r \le b$, $1 \le i \le k$, $0 \le v \le g-1$, $1 \le i' \le \binom{k}{t}$, and $0 \le w \le g^t - 1$. The variable $p\,(x_{r,i} = v)$ and $p\,(y_{r,i'} = w)$ are intended to denote $x_{r,i} = v$ in the original matrix and $y_{r,i'} = w$ in the alternative matrix respectively. The formula for $CA(b; t, k, g)$ are defined to be

$$\bigvee_{v} p\,(x_{r,i} = v) \tag{1}$$

$$\neg p\,(x_{r,i} = v) \vee \neg p\,(x_{r,i} = v') \tag{2}$$

$$\bigvee_{w} p\,(y_{r,i'} = w) \tag{3}$$

$$\neg p\,(y_{r,i'} = w) \vee \neg p\,(y_{r,i'} = w') \tag{4}$$

$$\bigvee_{r} p\,(y_{r,i'} = w) \tag{5}$$

$$\neg p\,(y_{r,i'} = w) \vee p\,(x_{r,i} = v) \tag{6}$$

where $1 \le r \le b$, $1 \le i \le k$, $0 \le v < v' \le g-1$, $1 \le i' \le \binom{k}{t}$, and $0 \le w < w' \le g^t - 1$; the clauses (6) are defined for all r,i,i',v, and w such that $y_{r,i'} = w$ and $x_{r,i} = v$ are permitted by the channelling constraints between $y_{r,i'}$ and $x_{r,i}$.

The clauses (1,3) express the condition that each CSP variable is assigned to at least one domain value. The clauses (2,4) express the condition that each CSP variable is assigned to at most one domain value. The clauses (5) express the lower bound on the coverage constraints that every number in the range 0 to $g^t - 1$ occurs at least once in every column of the alternative matrix. There are no clauses for the upper bound on the coverage constraints since it is an implied constraint and can be omitted. The clauses (6) express the channelling constraints. Note that the clauses (1,3,4) can be omitted (see [20, 21] for details).

Hnich's SAT encoding, with the use of a stochastic local search algorithm, found many solutions that are competitive with the previously known results for the arrays of strength $2 \leq t \leq 4$ with small to moderate size of k and g. It also found an improved solution for a large array $CA(40; 3, 7, 3)$.

However, a good encoding for backtrack search is not necessarily the same as that for local search. We thus propose two encodings suitable for modern CDCL SAT solvers.

4 Order Encoding

4.1 Overview of Order Encoding

The order encoding [8, 9] is a method that encodes a finite linear CSP into SAT. In order encoding, we introduce one propositional variable $p\,(x \leq i)$ for each CSP variable x and each integer constant i $(\ell(x) - 1 \leq i \leq u(x))$, where $\ell(x)$ and $u(x)$ are the lower and upper bounds of x respectively[2]. The variable $p\,(x \leq i)$ is intended to indicate $x \leq i$. The key feature of this encoding is the natural representation of the order structure on integers.

For each CSP variable x, we require the following clauses as axioms expressing the bounds and the order relation, where $\ell(x) \leq i \leq u(x)$.

$$\neg p\,(x \leq \ell(x) - 1) \quad p\,(x \leq u(x)) \quad \neg p\,(x \leq i - 1) \vee p\,(x \leq i)$$

Constraints are encoded into clauses expressing conflict regions instead of conflict points. When all points $(x_1, \ldots, x_n)$ in the region $i_1 < x_1 \leq j_1,\ \ldots,\ i_n < x_n \leq j_n$ violate the constraint, the following clause is added.

$$p\,(x_1 \leq i_1) \vee \neg p\,(x_1 \leq j_1) \vee \cdots \vee p\,(x_n \leq i_n) \vee \neg p\,(x_n \leq j_n)$$

Any finite linear comparison $\sum_{i=1}^{n} a_i\, x_i \leq c$ can be encoded into the following CNF formula, where a_i's are non-zero integer constants, c is an integer constant, and x_i's are mutually distinct integer variables.

$$\bigwedge_{\sum_{i=1}^{n} b_i = c-n+1} \bigvee_{i} (a_i\, x_i \leq b_i)^{\#}$$

The parameters b_i's range over integers satisfying $\sum_{i=1}^{n} b_i = c - n + 1$ and $\ell(a_i x_i) - 1 \leq b_i \leq u(a_i x_i)$ for all i where functions ℓ and u give the lower and upper bounds of the given expression respectively. The translation $()^{\#}$ is defined as follows.

$$(a\, x \leq b)^{\#} \equiv \begin{cases} p\left(x \leq \left\lfloor \dfrac{b}{a} \right\rfloor\right) & (a > 0) \\ \neg p\left(x \leq \left\lceil \dfrac{b}{a} \right\rceil - 1\right) & (a < 0) \end{cases}$$

[2] The variables $p\,(x \leq \ell(x) - 1)$ and $p\,(x \leq u(x))$ are redundant because they are always false and true respectively. However, we use them for simplicity of explanation.

Let us consider an example of encoding $x+y \leq 7$ with $x, y \in \{2,3,4,5,6\}$. First, we introduce the following twelve variables.

$$\begin{array}{cccccc} p\,(x \leq 1) & p\,(x \leq 2) & p\,(x \leq 3) & p\,(x \leq 4) & p\,(x \leq 5) & p\,(x \leq 6) \\ p\,(y \leq 1) & p\,(y \leq 2) & p\,(y \leq 3) & p\,(y \leq 4) & p\,(y \leq 5) & p\,(y \leq 6) \end{array}$$

Second, we require the following fourteen axiom clauses for encoding the integer variables x and y.

$$\begin{array}{c} \neg p\,(x \leq 1) \qquad p\,(x \leq 6) \\ \neg p\,(x \leq 1) \vee p\,(x \leq 2) \quad \neg p\,(x \leq 2) \vee p\,(x \leq 3) \\ \neg p\,(x \leq 3) \vee p\,(x \leq 4) \quad \neg p\,(x \leq 4) \vee p\,(x \leq 5) \quad \neg p\,(x \leq 5) \vee p\,(x \leq 6) \\ \text{(Similar clauses for } y\text{)} \end{array}$$

Finally, the constraint $x + y \leq 7$ is encoded into the following five clauses.

$$\begin{array}{c} p\,(x \leq 1) \vee p\,(y \leq 5) \quad p\,(x \leq 2) \vee p\,(y \leq 4) \\ p\,(x \leq 3) \vee p\,(y \leq 3) \quad p\,(x \leq 4) \vee p\,(y \leq 2) \quad p\,(x \leq 5) \vee p\,(y \leq 1) \end{array}$$

4.2 An Order Encoding of CA

Now, we present an order encoding of the integrated matrix model. We introduce the variables $p\,(x_{r,i} \leq v)$ and $p\,(y_{r,i'} \leq w)$ with $1 \leq r \leq b$, $1 \leq i \leq k$, $-1 \leq v \leq g-1$, $1 \leq i' \leq \binom{k}{t}$, and $-1 \leq w \leq g^t - 1$. The formula for $CA(b;t,k,g)$ are defined as follows except for the channelling constraints:

$$\neg p\,(x_{r,i} \leq -1) \tag{7}$$
$$p\,(x_{r,i} \leq g-1) \tag{8}$$
$$\neg p\,(x_{r,i} \leq v-1) \vee p\,(x_{r,i} \leq v) \tag{9}$$
$$\neg p\,(y_{r,i'} \leq -1) \tag{10}$$
$$p\,(y_{r,i'} \leq g^t - 1) \tag{11}$$
$$\neg p\,(y_{r,i'} \leq w-1) \vee p\,(y_{r,i'} \leq w) \tag{12}$$
$$\bigvee_r \left(\neg p\,(y_{r,i'} \leq w-1) \wedge p\,(y_{r,i'} \leq w)\right) \tag{13}$$

where $1 \leq r \leq b$, $1 \leq i \leq k$, $0 \leq v \leq g-1$, $1 \leq i' \leq \binom{k}{t}$, and $0 \leq w \leq g^t - 1$.

The clauses (7,8,9) and (10,11,12) are axiom clauses expressing the bounds and the order relation of integer variables in the original and alternative matrices respectively. Obviously, the formula (13) is not clausal form. We thus translate it into equi-satisfiable CNF formula by using the well-known *Tseitin transformation* with introducing new additional b variables.

Each channelling constraint $y_{r,i'} = \sum_{\ell=1}^{t} g^{t-\ell} x_{r,c_\ell^{i'}}$ in the integrated matrix model is first replaced with the following conjunction of two linear comparisons:

$$\left(y_{r,i'} \leq \sum_{\ell=1}^{t} g^{t-\ell} x_{r,c_\ell^{i'}}\right) \wedge \left(y_{r,i'} \geq \sum_{\ell=1}^{t} g^{t-\ell} x_{r,c_\ell^{i'}}\right).$$

And then each linear comparison is encoded into SAT in the same way as $\sum_{i=1}^{n} a_i\, x_i \le c$, as described in previous subsection.

The drawback of this encoding is the number of clauses required for the coverage constraints. We need in total $O(b\binom{k}{t}g^t)$ clauses since additional $O(b)$ clauses are required for each of (13). To avoid this problem, we propose another encoding, called mixed encoding.

5 Mixed Encoding

The mixed encoding is based on a combination of the order encoding and Hnich's encoding. It is designed to slightly reduce the number of clauses required compared with the order encoding. The basic idea is that the original matrix is encoded by the order encoding, and the alternative matrix by Hnich's encoding.

In the mixed encoding, we introduce the variables $p\,(x_{r,i} \le v)$ and $p\,(y_{r,i'} = w)$ with $1 \le r \le b$, $1 \le i \le k$, $-1 \le v \le g-1$, $1 \le i' \le \binom{k}{t}$, and $0 \le w \le g^t - 1$. The formula for $CA(b;t,k,g)$ are defined as follows:

$$\neg p\,(x_{r,i} \le -1) \tag{14}$$

$$p\,(x_{r,i} \le g-1) \tag{15}$$

$$\neg p\,(x_{r,i} \le v-1) \vee p\,(x_{r,i} \le v) \tag{16}$$

$$\bigvee_{w} p\,(y_{r,i'} = w) \tag{17}$$

$$\neg p\,(y_{r,i'} = w) \vee \neg p\,(y_{r,i'} = w') \tag{18}$$

$$\bigvee_{r} p\,(y_{r,i'} = w) \tag{19}$$

$$\neg p\,(y_{r,i'} = w) \vee \neg p\,(x_{r,i} \le v-1) \tag{20}$$

$$\neg p\,(y_{r,i'} = w) \vee p\,(x_{r,i} \le v) \tag{21}$$

where $1 \le r \le b$, $1 \le i \le k$, $0 \le v \le g-1$, $1 \le i' \le \binom{k}{t}$, $0 \le w < w' \le g^t - 1$; the clauses (20,21) are defined for all r,i,i',v, and w such that $y_{r,i'} = w$ and $x_{r,i} = v$ (i.e. $x_{r,i} \ge v \wedge x_{r,i} \le v$) are permitted by the channelling constraints between $y_{r,i'}$ and $x_{r,i}$.

The clauses (14,15,16) are the same as (7,8,9) of our order encoding. The clauses (17,18,19) are the same as (3,4,5) of Hnich's encoding. The channelling constraints are expressed by the clauses (20,21) that are slightly modified to adjust the order encoding variables compared with (6) of Hnich's encoding.

In SAT encoding, it is often effective to keep the number of clauses relatively small with respect to the size of problems. In this sense, we can omit (18) since these clauses can be derived from (16,20,21) by the resolution principle. For example, in the case of $CA(11;2,5,3)$, $\neg p\left(y_{r,(i,j)} = 0\right) \vee \neg p\left(y_{r,(i,j)} = 2\right)$ is derived from $\neg p\,(x_{r,j} \le 0) \vee p\,(x_{r,j} \le 1)$, $\neg p\left(y_{r,(i,j)} = 0\right) \vee p\,(x_{r,j} \le 0)$, and $\neg p\left(y_{r,(i,j)} = 2\right) \vee \neg p\,(x_{r,j} \le 1)$. We can also omit (17) since it can happen that some of the entries of the array are not needed in order to cover all t-tuples.

Table 2. Comparison of different encodings for $CA(b;t,k,g)$

	Hnich's encoding	Order encoding	Mixed encoding
Original matrix	$\{bk\} + bk\binom{g}{2}$	$bk(g-1)$	$bk(g-1)$
Alternative matrix	$\left\{b\binom{k}{t} + b\binom{k}{t}\binom{g^t}{2}\right\}$	$b\binom{k}{t}(g^t-1)$	$\left\{b\binom{k}{t} + b\binom{k}{t}\binom{g^t}{2}\right\}$
Coverage constraints	$\binom{k}{t}g^t$	$O(b\binom{k}{t}g^t)$	$\binom{k}{t}g^t$
Channelling constraints	$b\binom{k}{t}g^t t$	$O(b\binom{k}{t}g^t)$	$O(b\binom{k}{t}g^t t)$

Even if the clauses (17,18) may be omitted, we can still get a CSP solution by decoding a SAT solution. For any SAT solutions, the clauses (19) ensure that every number in the range 0 to $g^t - 1$ occurs at least once in every column of the alternative matrix. For each of such occurrences, the corresponding entries of the original matrix (i.e. a t-tuple from g symbols) is derived from the clauses (20,21). The condition that each entry is assigned to exactly one domain value is ensured by the axiom clauses (14,15,16).

6 Comparison and Symmetry

We compare the number of clauses required for $CA(b;t,k,g)$ of three different encodings. Table 2 [3] shows the comparison results between Hnich's encoding, the order encoding, and the mixed encoding. The bracket "{}" means the bracketed number of clauses can be omitted. As a result, each encoding has strength and weakness. Without omitting any clauses, the order encoding is the best except for the coverage constraints. In contrast, for the reduced number of clauses, the mixed encoding is better than the order encoding except for the channelling constraints. The length of each clause for the channelling constraints is two in the mixed and Hnich's encodings, but $t+1$ in the order encoding.

On the other hand, it is common that symmetry breaking techniques can considerably reduce complete backtrack search. A covering array is highly symmetric, and we treat two kinds of symmetries in this paper.

One is the row and column symmetry [31]. For given a covering array, any row and/or column can be permuted with any other row and/or column. Hnich *et al.* reduce this symmetry by using *lexicographic ordering constraints* in their matrix models. In the integrated matrix model, they break the column symmetry by ordering adjacent pairs of columns of the original matrix lexicographically, and the row symmetry by ordering adjacent pairs of rows of either the original or the alternative matrix lexicographically. Alternatively, we can use an incomplete symmetry breaking method called *snake lex* [32].

Another one is the value symmetry. For given a covering array, the symbols in any column of the array can be swapped. Hnich *et al.* proposed two methods for breaking this symmetry. Let $f_{i,v}$ be the frequency of occurrences of the symbol v $(0 \le v \le g-1)$ in the column i $(1 \le i \le k)$ of the original matrix. They impose

[3] In our two encodings, the number of some redundant clauses including literals such as $p\,(x_{r,i} \le -1)$, $p\,(x_{r,i} \le g-1)$, $p\,(y_{r,i'} \le -1)$, and $p\,(y_{r,i'} \le g^t - 1)$ are omitted.

the constraints such that $f_{i,0} \leq f_{i,1} \leq \ldots \leq f_{i,g-1}$ for all i. Alternatively, when $g = 2$, they constrain every symbol in the first (or the last) row of the original matrix to be 0 (or 1). Yan and Zhang proposed another method called LNH (*Least Number Heuristic*) for breaking this symmetry [33].

In our experiments, we use the same constraints as Hnich's methods for breaking two symmetries mentioned above. We note that applying these constraints does not lose any solutions.

7 Experiments

To evaluate the effectiveness of our encodings, we solve CA optimization problems (97 problems in total) of strength $2 \leq t \leq 6$ with small to moderate size of k and g. We then compare our two encodings with Hnich's encoding.

For each problem, we encode multiple CA decision problems of $CA(b; t, k, g)$ with varying the value of b into SAT. Such SAT-encoded problems contain both satisfiable and unsatisfiable problems and the optimal solution exists on the boundary. For every encoding, we add the clauses for breaking the row and column symmetry and value symmetry as discussed in the previous section. We omit the clause (17,18) in the mixed encoding and (3,4) in Hnich's encoding.

We use the MiniSat solver [34] as a high-performance CDCL SAT solver. More precisely, we use two implementations of the MiniSat solver: MiniSat 2.0 (simp) and the preview version of MiniSat 2.2 (simp). Main differences of MiniSat 2.2 (simp) are rapid restart, phase saving, blocking literals, and robust CNF-simplification.

First, Table 3 shows CPU time of the MiniSat solver in seconds for solving SAT-encoded $CA(b; t, k, g)$. We only shows our best lower and/or upper bounds of b for each CA optimization problem. We use the symbol "$*$" to indicate that the value of b is optimal. Each CPU time is better one of MiniSat 2.0 (simp) and MiniSat 2.2 (simp). We highlight the best time of different encodings for each problem. The column "Result" indicates whether it is satisfiable (`SAT`) or unsatisfiable (`UNSAT`). The columns "H.E.", "O.E.", and "M.E." indicate Hnich's encoding, the order encoding, and the mixed encoding respectively. All times were collected on a Linux machine with Intel Xeon 3.00GHz and 8GB memory. We set a timeout ($T.O$) for the MiniSat solver to 1800 seconds for each SAT-encoded $CA(b; t, k, g)$, except that the timeout of $CA(14; 3, 12, 2)$ is set to 12 hours.

Each of our encodings reproduced and re-proved 47 previously known optimal solutions, rather than 29 by Hnich's encoding. Moreover, our encodings found and proved the optimality of previously known upper bounds for $CAN(3, 12, 2)$ and $CAN(6, 8, 2)$. The previously known bound of $CAN(3, 12, 2)$ was $14 \leq CAN(3, 12, 2) \leq 15$. We found and proved $CAN(3, 12, 2) = 15$ since there is no solution to $CA(14; 3, 12, 2)$ as can be seen in Table 3. $CAN(3, 12, 2) = 15$ is the solution for an open problem listed in Handbook of Satisfiability [25][4]. We also improved on previously known lower bounds [12] for some arrays. Table 4 shows the summary of our newly obtained results to the best of our knowledge.

[4] We found and proved $CAN(3, 5, 3) = 33$. It is also the solution for an open problem listed in the same Handbook, but that was already closed in [19].

Table 3. Benchmark results of different encodings for $CA(b; t, k, g)$

t	k	g	b	Result	H.E.	O.E.	M.E.
2	3	3	**9***	SAT	**0.00**	0.01	**0.00**
2	4	3	**9***	SAT	0.01	**0.00**	0.01
2	5	3	10	UNSAT	0.59	**0.02**	**0.02**
2	5	3	**11***	SAT	0.03	0.03	**0.01**
2	6	3	11	UNSAT	7.55	**0.04**	0.05
2	6	3	**12***	SAT	0.03	**0.02**	0.03
2	7	3	**12***	SAT	**0.05**	0.38	0.32
2	8	3	13	SAT	**4.22**	1263.58	263.76
2	9	3	13	SAT	760.05	**228.39**	*T.O*
2	10	3	14	SAT	518.18	79.75	**14.60**
2	11	3	15	SAT	0.15	0.77	**0.13**
2	12	3	15	SAT	1.31	0.49	**0.21**
2	13	3	15	SAT	**16.40**	18.53	30.60
2	14	3	15	SAT	372.95	**192.93**	373.64
2	15	3	16	SAT	155.21	31.71	**30.80**
2	16	3	16	SAT	743.74	1674.69	**390.72**
2	3	4	**16***	SAT	0.02	**0.01**	**0.01**
2	4	4	**16***	SAT	0.04	0.04	**0.02**
2	5	4	**16***	SAT	0.05	0.05	**0.04**
2	6	4	18	UNSAT	*T.O*	**4.31**	9.90
2	6	4	**19***	SAT	21.28	**0.87**	3.14
2	7	4	19	UNSAT	*T.O*	177.23	**174.89**
2	7	4	22	SAT	71.18	20.71	**4.66**
2	8	4	23	SAT	550.61	22.89	**3.39**
2	9	4	24	SAT	*T.O*	**230.44**	469.66
2	10	4	25	SAT	*T.O*	440.30	**254.51**
2	11	4	26	SAT	*T.O*	*T.O*	**884.22**
2	3	5	**25***	SAT	0.08	**0.05**	0.06
2	4	5	**25***	SAT	0.26	0.18	**0.14**
2	5	5	**25***	SAT	0.47	0.46	**0.15**
2	6	5	**25***	SAT	3.21	**0.51**	0.76
2	7	5	29	SAT	*T.O*	394.38	**90.50**
2	8	5	36	SAT	*T.O*	**654.95**	*T.O*
2	9	5	38	SAT	*T.O*	914.78	**279.84**
2	10	5	41	SAT	*T.O*	*T.O*	**386.87**
2	11	5	42	SAT	*T.O*	**1330.56**	*T.O*
2	3	6	**36***	SAT	0.46	0.16	**0.14**
2	4	6	37	SAT	22.72	**4.19**	4.24
2	5	6	40	SAT	*T.O*	**627.67**	*T.O*
2	6	6	45	SAT	*T.O*	**614.53**	*T.O*
2	7	6	50	SAT	*T.O*	1765.78	**1110.20**
2	8	6	52	SAT	*T.O*	*T.O*	**1236.90**
2	9	6	57	SAT	*T.O*	*T.O*	**773.33**
2	10	6	62	SAT	*T.O*	*T.O*	**407.40**
2	11	6	63	SAT	*T.O*	*T.O*	**1287.25**
2	3	7	**49***	SAT	1.67	**0.64**	0.66
2	4	7	**49***	SAT	460.07	**98.10**	171.55
2	5	7	57	SAT	*T.O*	*T.O*	**570.96**
2	6	7	65	SAT	*T.O*	**1513.01**	*T.O*
2	7	7	68	SAT	*T.O*	*T.O*	**1446.27**
2	8	7	72	SAT	*T.O*	*T.O*	**1362.44**
2	9	7	81	SAT	*T.O*	*T.O*	**1098.09**
2	10	7	88	SAT	*T.O*	*T.O*	**442.81**
2	11	7	93	SAT	*T.O*	*T.O*	**449.59**

t	k	g	b	Result	H.E.	O.E.	M.E.
3	4	2	**8***	SAT	**0.00**	**0.00**	**0.00**
3	5	2	9	UNSAT	**0.01**	**0.01**	**0.01**
3	5	2	**10***	SAT	**0.00**	**0.00**	**0.00**
3	6	2	11	UNSAT	0.30	0.02	**0.01**
3	6	2	**12***	SAT	0.02	0.01	**0.00**
3	7	2	**12***	SAT	0.02	0.03	**0.01**
3	8	2	**12***	SAT	0.03	0.04	**0.01**
3	9	2	**12***	SAT	0.05	0.07	**0.03**
3	10	2	**12***	SAT	0.08	0.12	**0.05**
3	11	2	**12***	SAT	0.13	0.15	**0.05**
3	12	2	14	UNSAT	*T.O*	**5607.25**	6228.16
3	12	2	**15***	SAT	0.61	0.89	**0.44**
3	13	2	16	SAT	24.91	7.57	**4.24**
3	14	2	16	SAT	*T.O*	**15.09**	23.68
3	15	2	17	SAT	*T.O*	435.35	**1.97**
3	16	2	17	SAT	*T.O*	86.07	**14.93**
3	17	2	20	SAT	*T.O*	**62.40**	125.27
3	18	2	21	SAT	**2.46**	44.99	35.62
3	19	2	22	SAT	**1.54**	176.79	4.16
3	4	3	**27***	SAT	0.07	0.05	**0.04**
3	5	3	32	UNSAT	*T.O*	**16.07**	27.85
3	5	3	**33***	SAT	632.27	**2.81**	16.85
3	6	3	**33***	SAT	*T.O*	15.46	**12.44**
3	7	3	46	SAT	*T.O*	**1180.95**	*T.O*
3	8	3	52	SAT	**407.37**	1148.09	*T.O*
3	9	3	56	SAT	*T.O*	*T.O*	**202.78**
3	4	4	**64***	SAT	9.00	0.68	**0.56**
3	5	4	**64***	SAT	*T.O*	2.01	**1.35**
3	6	4	**64***	SAT	*T.O*	3.13	**1.54**
3	4	5	**125***	SAT	*T.O*	**6.13**	6.99
3	5	5	**125***	SAT	*T.O*	86.51	**54.96**
3	6	5	**125***	SAT	*T.O*	177.13	**59.09**
4	5	2	**16***	SAT	0.02	**0.01**	**0.01**
4	6	2	20	UNSAT	*T.O*	0.07	**0.05**
4	6	2	**21***	SAT	0.36	0.08	**0.03**
4	7	2	23	UNSAT	*T.O*	0.35	**0.32**
4	7	2	**24***	SAT	529.12	0.68	**0.37**
4	8	2	**24***	SAT	107.13	0.79	**0.63**
4	9	2	**24***	SAT	399.55	**1.13**	1.20
4	10	2	**24***	SAT	279.46	6.47	**0.96**
4	11	2	**24***	SAT	26.20	4.66	**1.92**
4	12	2	**24***	SAT	1573.40	11.28	**2.12**
4	13	2	36	SAT	*T.O*	**372.87**	734.72
4	5	3	**81***	SAT	840.28	**1.13**	1.52
4	5	4	**256***	SAT	*T.O*	105.37	**104.00**
5	6	2	**32***	SAT	4.66	0.13	**0.11**
5	7	2	41	UNSAT	*T.O*	1.53	**1.10**
5	7	2	**42***	SAT	849.79	1.41	**1.24**
5	8	2	52	SAT	*T.O*	**95.01**	134.37
5	9	2	49	UNSAT	*T.O*	**344.65**	521.59
5	9	2	54	SAT	*T.O*	1431.35	**1097.27**
5	10	2	60	SAT	*T.O*	**550.67**	*T.O*
5	6	3	**243***	SAT	*T.O*	**156.64**	197.40
6	7	2	**64***	SAT	922.29	**2.97**	3.42
6	8	2	84	UNSAT	*T.O*	86.91	**52.94**
6	8	2	**85***	SAT	*T.O*	76.28	**48.49**

Table 4. New results found and proved by our encodings. We note that $80 \leq CAN(3,8,4)$ and $15 \leq CAN(3,k,2)$ with $k \geq 13$ were proved by our experimental results $20 \leq CAN(2,7,4)$ and $CAN(3,12,2) = 15$ with the help of Theorem 1 of (4) and (2) respectively.

New results	Previously known results
$\mathbf{20} \leq CAN(2,7,4) \leq 21$	$19 \leq CAN(2,7,4) \leq 21$
$\mathbf{80} \leq CAN(3,8,4) \leq 88$	$76 \leq CAN(3,8,4) \leq 88$
$CAN(3,12,2) = \mathbf{15}$	$14 \leq CAN(3,12,2) \leq 15$
$\mathbf{15} \leq CAN(3,k,2)$ $(k \geq 13)$	$14 \leq CAN(3,k,2)$ $(k \geq 13)$
$\mathbf{50} \leq CAN(5,9,2) \leq 54$	$48 \leq CAN(5,9,2) \leq 54$
$CAN(6,8,2) = \mathbf{85}$	$84 \leq CAN(6,8,2) \leq 85$

Second, Table 5 shows the comparison results of different approaches on the best known upper bounds of $CAN(t,k,g)$. Our comparison includes our two encodings with MiniSat ("O.E. & M.E."), Hnich's encoding with a new variant of the walksat [21] ("HEW"), and the integrated matrix model with the ILOG solver [21] ("CSP"). We also include Colbourn's CA tables [29] ("CAT"). These tables include a list of current upper bounds on $CAN(t,k,g)$ for $(2 \leq t \leq 6)$. Their results have been obtained from orthogonal arrays, several theoretical works, and computational search methods such as greedy methods, tabu search, hill-climbing, simulated annealing, and so on. We highlight the best value of different approaches for each $CAN(t,k,g)$. The symbol "-" is used to indicate that the result is not available in published literature.

Our encodings with MiniSat were able to produce competitive bounds with those in Colbourn's CA tables for $CAN(t,k,g)$ of strength $2 \leq t \leq 6$ with small to moderate size of k and g. Although not able to match Hnich's encoding for some $CAN(2,k,g)$ and $CAN(3,k,3)$, our encodings were able to give a greater number of bounds than the integrated matrix model with ILOG and Hnich's encoding with walksat.

Finally, we discuss some details of our experimental results. Both of our encodings found and proved optimal solutions for the same number of problems (49 problems) including two open problems. The main difference between both encodings in Table 3 is that the mixed encoding gave approximate solutions for some arrays of strength two not solved in timeout by the order encoding. Compared to MiniSat 2.0 (simp), MiniSat 2.2 (simp) was better especially for the arrays of strength two. For example, our best approximate solutions for $CA(b;2,k,6)$ with $5 \leq k \leq 11$ and $CA(b;2,k,7)$ with $6 \leq k \leq 11$ were obtained by only MiniSat 2.2 (simp). As mentioned in Section 5, the clauses (17,18) can be omitted in the mixed encoding. In the case of $CA(15;3,12,2)$ with symmetry breaking, when we omit those clauses, the mixed encoding requires 77,442 clauses, but it requires 175,826 clauses when do not omit any clauses. For breaking the row and column symmetry, we applied *double lex* to the original matrix by encoding it into SAT. This greatly reduced the search space and execution time. Using snake lex [32] instead of double lex was less effective in our further experiments not presented in this paper.

Table 5. Comparison results of different approaches on the best known upper bounds of $CAN(t, k, g)$

t	k	g	O.E.& M.E.	HEW [21]	CAT [29]
2	3	3	**9**	**9**	**9**
2	4	3	**9**	**9**	**9**
2	5	3	**11**	**11**	**11**
2	6	3	**12**	**12**	**12**
2	7	3	**12**	**12**	**12**
2	8	3	**13**	14	**13**
2	9	3	**13**	**13**	**13**
2	10	3	**14**	**14**	**14**
2	11	3	**15**	**15**	**15**
2	12	3	**15**	–	**15**
2	13	3	**15**	–	**15**
2	14	3	**15**	–	**15**
2	15	3	16	–	**15**
2	16	3	16	–	**15**
2	3	4	**16**	**16**	**16**
2	4	4	**16**	**16**	**16**
2	5	4	**16**	**16**	**16**
2	6	4	**19**	**19**	**19**
2	7	4	22	**21**	**21**
2	8	4	23	23	**22**
2	9	4	24	24	**23**
2	10	4	25	25	**24**
2	11	4	26	25	**24**
2	3	5	**25**	**25**	**25**
2	4	5	**25**	**25**	**25**
2	5	5	**25**	**25**	**25**
2	6	5	**25**	**25**	**25**
2	7	5	**29**	**29**	**29**
2	8	5	36	34	**33**
2	9	5	38	**35**	**35**
2	10	5	41	38	**36**
2	11	5	42	39	**38**
2	3	6	**36**	**36**	**36**
2	4	6	**37**	**37**	**37**
2	5	6	40	**39**	**39**
2	6	6	45	42	**41**
2	7	6	50	45	**42**
2	8	6	52	48	**42**
2	9	6	57	51	**46**
2	10	6	62	53	**49**
2	11	6	63	55	**52**
2	3	7	**49**	**49**	**49**
2	4	7	**49**	**49**	**49**
2	5	7	57	52	**49**
2	6	7	65	58	**49**
2	7	7	68	61	**49**
2	8	7	72	63	**49**
2	9	7	81	66	**59**
2	10	7	88	71	**61**
2	11	7	93	73	**67**

t	k	g	O.E.& M.E.	CSP [21]	HEW [21]	CAT [29]
3	4	2	**8**	**8**	–	**8**
3	5	2	**10**	**10**	–	**10**
3	6	2	**12**	**12**	–	**12**
3	7	2	**12**	**12**	–	**12**
3	8	2	**12**	**12**	–	**12**
3	9	2	**12**	**12**	**12**	**12**
3	10	2	**12**	**12**	**12**	**12**
3	11	2	**12**	**12**	**12**	**12**
3	12	2	**15**	–	**15**	**15**
3	13	2	**16**	–	**16**	**16**
3	14	2	**16**	–	17	**16**
3	15	2	**17**	–	18	**17**
3	16	2	**17**	–	18	**17**
3	17	2	20	–	**18**	**18**
3	18	2	21	–	20	**18**
3	19	2	22	–	–	**18**
3	4	3	**27**	–	–	**27**
3	5	3	**33**	–	**33**	**33**
3	6	3	**33**	–	**33**	**33**
3	7	3	46	–	**40**	**40**
3	8	3	52	–	46	**42**
3	9	3	56	–	51	**45**
3	4	4	**64**	–	–	**64**
3	5	4	**64**	–	–	**64**
3	6	4	**64**	–	–	**64**
3	4	5	**125**	–	–	**125**
3	5	5	**125**	–	–	**125**
3	6	5	**125**	–	–	**125**
4	5	2	**16**	**16**	–	**16**
4	6	2	**21**	**21**	–	**21**
4	7	2	**24**	–	**24**	**24**
4	8	2	**24**	–	**24**	**24**
4	9	2	**24**	–	**24**	**24**
4	10	2	**24**	–	**24**	**24**
4	11	2	**24**	–	–	**24**
4	12	2	**24**	–	–	**24**
4	13	2	36	–	–	**32**
4	5	3	**81**	–	**81**	**81**
4	5	4	**256**	–	–	**256**
5	6	2	**32**	–	–	**32**
5	7	2	**42**	–	–	**42**
5	8	2	**52**	–	–	**52**
5	9	2	**54**	–	–	**54**
5	10	2	60	–	–	**56**
5	6	3	**243**	–	–	**243**
6	7	2	**64**	–	–	**64**
6	8	2	**85**	–	–	**85**

8 Conclusion

In this paper, we considered the problem of finding optimal covering arrays by SAT encoding. We presented two encodings suitable for modern CDCL SAT solvers. To evaluate the effectiveness of our encodings, we solved CA optimization problems (97 problems in total) of strength $2 \leq t \leq 6$ with small to moderate size of components k and symbols g. Each of our encodings found and proved 49

optimal solutions including two previously unknown results and also improved known lower bounds for some arrays, as shown in Table 4.

CDCL SAT solvers have an essential role in enhancing efficiency. We investigated a SAT-based approach to generate test cases for combinatorial testing. Our approach is based on an effective combination of the order encoding (or the mixed encoding) and CDCL SAT solvers. There are several future topics for making our approach scalable to large problems. Among them, it is very important to encode the upper bound of the coverage constraints into SAT with as small number of clauses as possible.

References

1. de Kleer, J.: A comparison of ATMS and CSP techniques. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 1989), pp. 290–296 (1989)
2. Walsh, T.: SAT v CSP. In: Dechter, R. (ed.) CP 2000. LNCS, vol. 1894, pp. 441–456. Springer, Heidelberg (2000)
3. Kasif, S.: On the parallel complexity of discrete relaxation in constraint satisfaction networks. Artificial Intelligence 45(3), 275–286 (1990)
4. Gent, I.P.: Arc consistency in SAT. In: Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002), pp. 121–125 (2002)
5. Selman, B., Levesque, H.J., Mitchell, D.G.: A new method for solving hard satisfiability problems. In: Proceedings of the 10th National Conference on Artificial Intelligence (AAAI 1992), pp. 440–446 (1992)
6. Iwama, K., Miyazaki, S.: SAT-variable complexity of hard combinatorial problems. In: Proceedings of the IFIP 13th World Computer Congress, pp. 253–258 (1994)
7. Gelder, A.V.: Another look at graph coloring via propositional satisfiability. Discrete Applied Mathematics 156(2), 230–243 (2008)
8. Tamura, N., Taga, A., Kitagawa, S., Banbara, M.: Compiling finite linear CSP into SAT. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 590–603. Springer, Heidelberg (2006)
9. Tamura, N., Taga, A., Kitagawa, S., Banbara, M.: Compiling finite linear CSP into SAT. Constraints 14(2), 254–272 (2009)
10. Gavanelli, M.: The log-support encoding of CSP into SAT. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 815–822. Springer, Heidelberg (2007)
11. Williams, A.W.: Determination of test configurations for pair-wise interaction coverage. In: Proceedings of 13th International Conference on Testing Communicating Systems (TestCom 2000), pp. 59–74 (2000)
12. Chateauneuf, M.A., Kreher, D.L.: On the state of strength-three covering arrays. Journal of Combinatorial Designs 10(4), 217–238 (2002)
13. Hartman, A., Raskin, L.: Problems and algorithms for covering arrays. Discrete Mathematics 284(1-3), 149–156 (2004)
14. Cohen, D.M., Dalal, S.R., Fredman, M.L., Patton, G.C.: The AETG system: An approach to testing based on combinatiorial design. IEEE Transactions on Software Engineering 23(7), 437–444 (1997)
15. Lei, Y., Tai, K.C.: In-parameter-order: A test generation strategy for pairwise testing. In: Proceedings of 3rd IEEE International Symposium on High-Assurance Systems Engineering (HASE 1998), pp. 254–261 (1998)

16. Nurmela, K.J.: Upper bounds for covering arrays by tabu search. Discrete Applied Mathematics 138(1-2), 143–152 (2004)
17. Cohen, M.B., Gibbons, P.B., Mugridge, W.B., Colbourn, C.J.: Constructing test suites for interaction testing. In: Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), pp. 38–48 (2003)
18. Shiba, T., Tsuchiya, T., Kikuno, T.: Using artificial life techniques to generate test cases for combinatorial testing. In: Proceedings of 28th International Computer Software and Applications Conference (COMPSAC 2004), pp. 72–77 (2004)
19. Bulutoglu, D., Margot, F.: Classification of orthogonal arrays by integer programming. Journal of Statistical Planning and Inference 138, 654–666 (2008)
20. Hnich, B., Prestwich, S.D., Selensky, E.: Constraint-based approaches to the covering test problem. In: Faltings, B.V., Petcu, A., Fages, F., Rossi, F. (eds.) CSCLP 2004. LNCS (LNAI), vol. 3419, pp. 172–186. Springer, Heidelberg (2005)
21. Hnich, B., Prestwich, S.D., Selensky, E., Smith, B.M.: Constraint models for the covering test problem. Constraints 11(2-3), 199–219 (2006)
22. Lopez-Escogido, D., Torres-Jimenez, J., Rodriguez-Tello, E., Rangel-Valdez, N.: Strength two covering arrays construction using a sat representation. In: Gelbukh, A., Morales, E.F. (eds.) MICAI 2008. LNCS (LNAI), vol. 5317, pp. 44–53. Springer, Heidelberg (2008)
23. Cohen, M.B., Dwyer, M.B., Shi, J.: Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach. IEEE Trans. Software Eng. 34(5), 633–650 (2008)
24. Drescher, C., Walsh, T.: A translational approach to constraint answer set solving. TPLP 10(4-6), 465–480 (2010)
25. Zhang, H.: Combinatorial designs by sat solvers. In: Handbook of Satisfiability, pp. 533–568. IOS Press, Amsterdam (2009)
26. Colbourn, C.J.: Strength two covering arrays: Existence tables and projection. Discrete Mathematics 308(5-6), 772–786 (2008)
27. Sloane, N.J.A.: Covering arrays and intersecting codes. Journal of Combinatorial Designs 1, 51–63 (1993)
28. Seroussi, G., Bshouty, N.H.: Vector sets for exhaustive testing of logic circuits. IEEE Transactions on Information Theory 34(3), 513–522 (1988)
29. Colbourn, C.J.: Covering array tables (2010), `http://www.public.asu.edu/~ccolbou/src/tabby/catable.html` (Last Accessed on April 26, 2010)
30. Régin, J.C.: Generalized arc consistency for global cardinality constraint. In: Proceedings of the 13th National Conference on Artificial Intelligence (AAAI 1996), vol. 1, pp. 209–215 (1996)
31. Flener, P., Frisch, A.M., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., Walsh, T.: Breaking row and column symmetries in matrix models. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 462–476. Springer, Heidelberg (2002)
32. Grayland, A., Miguel, I., Roney-Dougal, C.M.: Snake lex: An alternative to double lex. In: Gent, I.P. (ed.) CP 2009. LNCS, vol. 5732, pp. 391–399. Springer, Heidelberg (2009)
33. Yan, J., Zhang, J.: A backtracking search tool for constructing combinatorial test suites. Journal of Systems and Software 81(10), 1681–1693 (2008)
34. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)