Combinatorial Testing with Shielding Parameters

Baiqiang Chen[1,3], Jun Yan[2] and Jian Zhang[1]
[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
{chenbq, zj}@ios.ac.cn
[2]Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences
junyan@acm.org
[3]Graduate University of Chinese Academy of Sciences

Abstract

Combinatorial testing is an important approach to detecting interaction errors for a system with several parameters. Existing research in this area assumes that all parameters of the system under test are always effective. However, in many realistic applications, there may exist some parameters that can disable other parameters in certain conditions. These parameters are called shielding parameters. Shielding parameters make test cases generated by the existing test model, which uses the Mixed Covering Array (MCA), fail in exposing some potential errors that should be detected. In this paper, the Mixed Covering Array with Shielding parameters (MCAS) is proposed to describe such problems. Then test cases can be generated by constructing MCAS's in three different approaches. According to the experimental results, our test model can generate satisfactory test cases for combinatorial testing with shielding parameters.

1 Introduction

Software quality increasingly receives attention from both academic and industrial researchers nowadays. Combinatorial testing is a widely used technology for software testing, which is an important approach to improving software quality. A *System Under Test (SUT)* usually has a number of parameters[1], and each parameter can take different values. For the sake of the exponential explosion, it is quite impractical to exhaust all possible value combinations due to the testing budgets of both time and costs. The main idea of combinatorial testing is, given an SUT with k parameters, to cover every possible value combination of no more than t parameters. Here, t is called the test strength. The strength is usually fixed as a small number ($e.g.$, $t = 2$ or 3), and combinatorial testing with strength t is often called t-way testing. By using combinatorial testing, the number of test cases can be dramatically reduced compared with exhaustive testing. At the same time, many analyses [6, 7, 8] have shown that the error detection rate of combinatorial testing is quite high.

Combinatorial testing is quite attractive since it achieves a good trade-off between effectiveness and efficiency. Test cases are generated by constructing Mixed Covering Arrays (MCA's). For example, given an SUT with three parameters $p_0 \in \{a, b\}$, $p_1 \in \{a, b\}$ and $p_2 \in \{a, b, c\}$. Test cases of 2-way testing can be generated by constructing the MCA shown in Table 1, where each column represents a parameter in the SUT, and each row represents a test case. As we can see, every possible value pair of any two parameters is covered by at least one test case of the table.

Table 1. MCA

p_0	p_1	p_2
a	a	a
b	a	b
a	b	b
b	b	a
a	a	c
b	b	c

Table 2. Valid Data

p_0	p_1	p_2
a	a	#
b	a	b
a	b	#
b	b	a
b	b	c

To the best of our knowledge, all the previous research assumes that every parameter of the SUT is effective all the time, $i.e.$, each entry takes an effective value in the MCA. However, when we try to test a railway control system ATP with the existing methods, we encounter a new problem. In this system, once we assign some value to a specific parameter, some of the other parameters are disabled consequently. Moreover, we also find such phenomenon in many other real systems (refer to Section 3).

Under such circumstances, using the traditional test model of MCA's to carry out t-way testing will miss some value combinations. In the preceding example, if p_0 can

[1]Parameters refer to any factors that can affect the running of an SUT such as external input data, configuration options, and running environments.

1530-1362/10 $26.00 © 2010 IEEE
DOI 10.1109/APSEC.2010.40

disable p_2 when it takes the value "a", the effective test cases of Table 1 will be transformed into Table 2 ("#" denotes ineffective data). As a result, many value pairs can not be covered by Table 2 now. For example, 50% value pairs of parameter p_1 and p_2 ($\langle a, a\rangle$, $\langle a, c\rangle$ and $\langle b, b\rangle$) can not be covered by these test cases. If there exist potential errors triggered by some of these missing value combinations, they can never be exposed and may contribute to fatal consequences. Therefore, a new test model is required to solve such problems, and this is exactly the motivation of our work.

In this paper, parameters that can disable other parameters are defined as shielding parameters. Our main contributions are as follows.

- Based on realistic applications, we propose the concept of shielding parameters for the first time, and point out the importance of paying attention to them in the t-way combinatorial testing.
- We establish principles on combinatorial testing with shielding parameters, and introduce a new test model, Mixed Covering Array with Shielding parameter (MCAS), to solve this problem.
- We give three different approaches to constructing MCAS's, and integrate them into a prototype tool CTES to generate test cases.

According to our experimental results on SUT's with shielding parameters, compared with the previous work, test cases generated by our method have the ability to expose all potential errors that should be detected in t-way testing. Since shielding parameters are quite common in real applications, this work can be employed to improve practical software testing.

The remainder of this paper is organized as follows. In the next section, we introduce the traditional test model of Mixed Covering Array (MCA). In Section 3, we introduce the definition of shielding parameters based on some practical examples, and show the importance of handling shielding parameters in combinatorial testing. Then in Section 4, we propose the definition of Mixed Covering Array with Shielding parameters (MCAS). Section 5 proposes three different algorithms to construct MCAS's. Section 6 demonstrates some experimental results. In Section 7, we discuss some related work. Finally, we conclude the paper in Section 8.

2 Mixed Covering Array (MCA)

Mixed Covering Array (MCA) is an important combinatorial structure used in previous combinatorial testing.

Definition 1. *$MCA(N; t, k, v_1 v_2 \cdots v_k)$ is an $N \times k$ matrix having the following properties:*

1. *symbols in the i-th column ($1 \le i \le k$) compose a set of size v_i;*
2. *any $N \times t$ sub-matrix covers all value combinations of the corresponding t columns.*

Here, t is called the MCA's strength. For example, Table 1 shows an instance of *MCA*$(6; 2, 3, 2{\cdot}2{\cdot}3)$ whose strength is two. In this array, the first and second columns have two symbols ("a" and "b"), and the third column has three symbols ("a", "b" and "c"). All tuples of any two columns are covered by the sub-matrix of size 6×2. *MCA*$(N; t, k, v_1 v_2 \cdots v_k)$ is often written as *MCA*$(N; t, k, w_1^{r_1} w_2^{r_2} \cdots w_s^{r_s})$, where $k = \sum_1^s r_i$ and $w_j \subseteq \{v_1, v_2, \ldots, v_k\}$ $(1 \le j \le k)$. For instance, *MCA*$(6; 2, 3, 2{\cdot}2{\cdot}3)$ is written as *MCA*$(6; 2, 3, 2^2 3)$.

In the previous test model, for an SUT with k parameters (where the i-th parameter has v_i values), the test suite of t-way combinatorial testing is represented by *MCA*$(N; t, k, v_1 v_2 \cdots v_k)$. In the MCA, each row represents a test case, and each column represents a parameter. For instance, the first row of Table 1 represents a test case $\langle p_0, p_1, p_2\rangle = \langle 0, 1, 0\rangle$.

There are many algorithms and off-the-shelf tools available to construct MCA's. Refer to Nie and Leung's survey [11] or the website maintained by Czerwonka [3] to find more details.

3 Shielding Parameters

3.1 Examples

The previous test model assumes every parameter of the SUT is effective in all conditions, but this is not the case for many realistic applications, *e.g.*, the ATP system. ATP is a railway control system used by China Academy of Railway Science, which has 17 different switches. Each switch takes charge of a module of the system, and can be set to either "ON" or "OFF". Due to the budget of test costs, we suggested carrying out 3-way testing and designed a test suite using the previous test model. When applying the test cases to real system, we found something special. In ATP, not all of the 17 switches are effective under all of the circumstances. When the switch in charge of bypass circuit module is set to "ON", the switch that controls the system's portal module no longer works. This is a phenomenon the previous combinatorial test model has not considered. Whereas, it is often seen in practice. The following are some other examples.

1. In TCP/IP configuration panel, there are two ways to obtain IP addresses (p_0): automatically (v_0) or manually (v_1). When we choose to obtain IP addresses automatically, parameters of "IP address"(p_1), "subnet mask" (p_2) and "default gateway" (p_3) are disabled.

2. In the Microsoft Office Word's font configuration panel, there are different choices for parameters "Underline(U)" and "Color of Underline(I)". Yet, when we select the optional value "None" for the former parameter, the latter one is disabled.

3. This phenomenon is quite common in command line applications. The classic example we are all familiar with is that, when we select "-h" (help option) for a command, the other options are ignored consequently.

3.2 Formalization

In fact, it is easy for us to find more examples from both routine software and specific applications. We think the above examples are sufficient to convince the motivation and necessity of our work. Each of them has one or more special parameters, which can cause some other parameters to be ineffective (disabled/ignored). This kind of parameters can be formalized as follows.

Definition 2. *Given an SUT with k parameters, let $P = \{p_0, p_1, \ldots, p_{k-1}\}$ be the set of these k parameters, where the domain of p_i is denoted by D_i $(0 \leq i < k)$. If there exists a parameter $p_a \in P$, a value set $\phi \subset D_a$ and a parameter set $\psi \subset P/\{p_a\}$, such that when p_a is assigned values from ϕ, all parameters in set ψ are either disabled or ignored. Then, parameter p_a is called a shielding parameter. Meanwhile, values in ϕ are called p_a's shielding values, variables in ψ are called p_a's dependent parameters. A parameter that is neither a shielding parameter nor a dependent parameter is called an ordinary parameter.*

Relationship between shielding parameter p_a and its dependent parameters is denoted by $S(p_a|\phi) = \psi$. Let us take the TCP/IP configuration panel for example. According to Definition 2, there is a shielding parameter p_0 in the application, which has one shielding value v_0 and three dependent parameters p_1, p_2 and p_3. Relationship between them is represented by $S(p_0|\{v_0\}) = \{p_1, p_2, p_3\}$.

3.3 Side Effects on MCA

We should pay special attention to shielding parameters. Given an SUT with five boolean parameters (p_0 to p_4), suppose we intend to generate test cases for 3-way combinatorial testing. Using the existing test model, we can obtain twelve test cases by constructing $MCA(12; 3, 5, 2^5)$ listed in Table 3. If there exists a shielding parameter p_0 satisfying $S(p_0|\{0\}) = \{p_1, p_2, p_3\}$, it will affect the generated test cases in the following aspects.

1. Invalidation. When shielding parameter p_0 takes its shielding value 0, values for parameters p_1, p_2 and p_3 become ineffective, which are marked by deleting lines in Table 3.

2. Redundancy. Many redundant test cases appear in the test cases after the invalid values are removed. For instance, lines number 1, 3 and 10 are duplicated as well as lines number 5, 7 and 11.

3. Unsoundness. Note that *t-way combinatorial testing aims to expose all potential errors caused by interactions of no more than t parameters*. After removing all the invalid data and redundant test cases from Table 3, we get the final effective results in Table 4. As we can see from the table, these test cases do not satisfy the requirement for 3-way testing any longer. Take three parameters p_1, p_2 and p_3 for example, there are totally 2^3 possible interactions, but among these interactions, $\langle 0, 1, 1\rangle$ and $\langle 0, 0, 0\rangle$ are not included in Table 4. Thus, if there exist any potential errors which can be caused by input $\langle p_1, p_2, p_3\rangle = \langle 0, 1, 1\rangle$ or $\langle p_1, p_2, p_3\rangle = \langle 0, 0, 0\rangle$, they will not be exposed.

As we see from the above example, test cases based on the previous test model of MCA are not appropriate for t-way testing on SUT's with shielding parameters. More experimental results are given in Section 6. To the best of our knowledge, no researchers have studied shielding parameters in the literature, and no available tools were designed to generate satisfactory test cases which can expose all potential errors triggered by no more than t parameters for this kind of problems. We give effective solutions in the following sections.

Table 3. MCA

p_0	p_1	p_2	p_3	p_4
0	~~0~~	~~0~~	~~0~~	0
1	0	0	1	1
0	~~1~~	~~0~~	~~1~~	0
1	1	0	0	1
0	~~0~~	~~1~~	~~1~~	1
1	0	1	0	0
0	~~1~~	~~1~~	~~0~~	1
1	1	1	1	0
1	1	0	0	0
0	~~0~~	~~1~~	~~1~~	0
0	~~0~~	~~0~~	~~0~~	1
1	1	1	1	1

Table 4. Valid Data

p_0	p_1	p_2	p_3	p_4
0	#	#	#	0
1	0	0	1	1
1	1	0	0	1
0	#	#	#	1
1	0	1	0	0
1	1	1	1	0
1	1	0	0	0
1	1	1	1	1

4 Our Test Model

In this section, we propose a new test model called *Mixed Covering Array with Shielding parameters (MCAS)*. Compared with MCA, MCAS can express shielding relationship and thus generate proper test cases in order to cover all the needed value combinations.

Table 5. The Whole Input Space

p_0	p_1	p_2	p_3
0	0	#	#
0	1	#	#
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

4.1 Preliminaries

Whenever a parameter p is effective, it takes values from its *effective domain*. For example, the effective domain of each switch in the ATP system is {ON, OFF}. In an SUT with shielding parameters, if p is a dependent parameter, it may be ineffective (shielded) in certain circumstances. When p is ineffective, it does not take part in that test case, $e.g.$, p_1, p_2 and p_3 do not appear in the first test case in Table 4. For the sake of convenience, an ineffective parameter is still regarded as part of the test case, which takes the vain value ("#").

Definition 3. *The vain value is a special value denoted by "#" such that, if and only if a dependent parameter is shielded, it takes "#".*

For example, the first test case in Table 4 is expressed by $\langle p_0, p_1, p_2, p_3, p_4\rangle = \langle 0, \#, \#, \#, 0\rangle$ rather than $\langle p_0, p_4\rangle = \langle 0, 0\rangle$. Thus, for an SUT with k parameters, no matter whether there are some shielding parameters or not, each test cas has the same size k.

Definition 4. *Among the whole space of input data for an SUT with k parameters, all value combinations of t ($t \leq k$) parameters that do not include vain values are called complete t-tuples, and all value combinations that include at least one vain value are called incomplete t-tuples.*

For example, Table 5 lists the input space for an SUT with four boolean parameters p_0, p_1, p_2 and p_3, and $S(p_0|\{0\}) = \{p_2, p_3\}$. There are only ten rather than 2^4 different inputs for this system due to the existence of shielding parameter p_0. As we can see from Table 5, there are three 2-tuples for p_0 and p_2 including two complete 2-tuples ($\langle 1, 0\rangle$ and $\langle 1, 1\rangle$) and one incomplete 2-tuple ($\langle 0, \#\rangle$).

4.2 Mixed Covering Array with Shielding parameters (MCAS)

Test cases of t-way testing for an SUT with shielding parameters should satisfy the following properties.

- Take the shielding relationship into account.
- Cover all possible complete t'-tuples for $t' \leq t$.

In our model, test cases are represented by *Mixed Covering Array with Shielding parameters (MCAS)*, where each column represents a parameter, and each row represents a test case. In MCAS, shielding relationship is expressed by a relationship set R, which consists of r triples $\langle c_i, \phi_i, \psi_i\rangle$ $(0 \leq i < r)$. Each triple expresses some shielding relationship, *e.g.*, $\langle 1, \{0\}, \{2, 3, 4\}\rangle$ means that the parameter corresponding to the array's first column is a shielding parameter, and when it is assigned 0, it will disable three parameters represented by the second, third and fourth columns. Note the integer $j \in \{c_i\} \cup \psi_i$ means the j-th attribute.

Definition 5. *Mixed Covering Array with Shielding parameters MCAS$(N; t, k, v_1 v_2 \cdots v_k; R)$, where $R = \{\langle c_i, \phi_i, \psi_i\rangle | 0 \leq i < r\}$, is an $N \times k$ matrix having the following properties.*

1. *Regardless of "#", symbols in the j-th column ($1 \leq j \leq k$) compose a set of size v_i.*
2. *For each row, if its c_i-th value belongs to ϕ_i, for all $j \in \psi_i$, the j-th value is "#".*
3. *For all $t' \leq t$, any $N \times t'$ sub-matrix covers all complete t'-tuples of these corresponding columns.*

Similar to terminologies used in MCA's definition, t is called the strength. Table 6 shows *MCAS*$(10; 3, 5, 2^5; R)$ where $R = \{\langle 1, \{0\}, \{2, 3, 4\}\rangle\}$. As we can see from Table 6, all of the legal complete $3/2/1$-tuples are covered by the array, which assures the generated test cases can expose any potential errors triggered by no more than three parameters.

Notice that the motivation of t-way testing is to expose all possible errors that can be triggered by *no more than* t parameters rather than exactly t parameters. The generated test cases will not be sound if we only require all of the complete t-tuples to be covered at least once. For instance, although the last eight test cases of Table 6 include all complete 3-tuples, they exclude two complete 2-tuples for p_0 and p_4, namely $\langle 0, 0\rangle$ and $\langle 0, 1\rangle$. This is the reason why we stress in the defintion for all $t' \leq t$, all complete t'-tuples should be covered at least once. Someone may find that in MCA's definition (refer to Definition 1 in Section 2), they do not explicitly require all complete t'-tuples where $t' < t$

Table 6. $MCAS(10; 3, 5, 2^5; \{\langle 1, \{0\}, \{2, 3, 4\}\rangle\})$

p_0	p_1	p_2	p_3	p_4
0	#	#	#	0
0	#	#	#	1
1	0	0	0	0
1	1	0	0	1
1	0	1	0	1
1	1	1	0	0
1	0	0	1	1
1	1	0	1	0
1	0	1	1	0
1	1	1	1	1

to be covered by the test cases. This is because in the previous model, no shielding parameters are taken into account, so all tuples in that model are complete. And thus, all of the complete t'-tuples are destined to be subsumed by the t-tuples implicitly.

In fact, we can also define MCAS in an equivalent way that only pays attention to t-tuples.

Definition 6. *Mixed Covering Array with Shielding parameters MCAS*$(N; t, k, v_1v_2\cdots v_k; R)$ *, where* $R = \{\langle c_i, \phi_i, \psi_i\rangle | 0 \leq i < r\}$*, is an* $N \times k$ *matrix having the following properties.*

1. *Regardless of "#", symbols in the* j*-th column (*$1 \leq j \leq k$*) compose a set of size* v_i*.*
2. *For each row, if its* c_i*-th value belongs to* ϕ_i*, for all* $j \in \psi_i$*, the* j*-th value is "#".*
3. *Any* $N \times t$ *sub-matrix covers all* t*-tuples (both complete and incomplete) of the corresponding* t *parameters.*

Theorem 1. *Definition 5 and Definition 6 are equivalent.*

Proof. Since Property (1) and Property (2) are the same, we need only prove the third properties are equivalent.

(1) Definition 6 implies Definition 5.

Since all t-tuples of t parameters are covered, then for any $t' \leq t$, t' parameters' complete t'-tuples are also covered because they are subsets of complete t-tuples.

(2) Definition 5 implies Definition 6.

Observe that in Definition 5, t' can be any number that is not greater than t, so all complete t-tuples must have been covered. The theorem will be proved if we can show that the incomplete t-tuples are also covered. Without any loss of generality, let us consider an incomplete t-tuple $\vec{v} = \langle a_1, a_2, \ldots, a_t\rangle$. Besides, let X denote a set that includes parameters which are assigned "#" in $\vec{v}$; let Y denote a set that stores all parameters which shield the parameters belonging to X. We have $|X| \leq t$ and $|Y| \leq |X|$. Let us consider a complete tuple $\vec{u}$ of size $(t - |X| + |Y|)$, which consists of $|Y|$ shielding parameters' shielding values, and $t - |X|$ values in $\vec{v}$ (except for $|X|$ "#"s). Since $t - |X| + |Y| \leq t$, $\vec{u}$ must be covered by the array. Since all shielding parameters in $\vec{u}$ takes shielding values , $\vec{v}$ will also appear at the same row. We have thus proved the theorem. □

Due to the equivalence of Definition 5 and Definition 6, we can flexibly choose either to prove conclusions in the future.

4.3 Assumption

According to our observations, shielding parameters in realistic applications often have the following features.

- If a parameter is a shielding parameter, it can not be a dependent parameter, and *vice versa*;
- A shielding parameter may have different shielding values, but it always shields the same set of parameters.

Thus for simplicity, in the following sections, the default relationship set $R = \{\langle c_1, \phi_1, \psi_1\rangle, \langle c_2, \phi_2, \psi_2\rangle, \ldots, \langle c_r, \phi_r, \psi_r\rangle\}$ is restricted to the following assumption.

For all $i, j \in \{1, 2, \ldots, r\}$ and $i \neq j$, $c_i \neq c_j$ and $c_i \notin \psi_j$.

5 Test Generation

Up to now, we have established the test model of MCAS. For an SUT with shielding parameters, test cases of t-way testing are generated by constructing a corresponding MCAS of strength t. In this section, we give three approaches to constructing MCAS's.

5.1 Conversion Method

The conversion method means to convert problems of constructing MCAS's into problems of constructing MCA's. It is based on the following relationship between MCAS's and MCA's.

Theorem 2. *For an* $N \times k$ *MCA of strength* $t + r$ *(*$t + r \leq k$*), if we designate* r *parameters as shielding parameters, after replacing the invalid values with "#"s, it becomes an MCAS of strength* t*.*

Proof. Properties (1) and (2) hold obviously.

According to Definition 6, the theorem will be proved if we can show all t-tuples of t parameters are covered by the array. Before designating r shielding parameters, $(t +$

r)-tuples of any $t + r$ parameters are covered in the array because it is an MCA of strength $t + r$. Without loss of generality, let $\vec{P}$ denote a set of t parameters.

(a). If $\vec{P}$ has no dependent parameters of the designated shielding parameters, then all t-tuples of these t parameters are kept in the array.

(b). If $\vec{P}$ has dependent parameters of the designated shielding parameters, let $\vec{P'}$ denote the union of $\vec{P}$ and these r designated shielding parameters. All $(t + r)$-tuples of parameters in $\vec{P'}$ are covered in the original MCA. Because all parameters that can bring vain values to value combinations of parameters in $\vec{P}$ are included in the r shielding parameters, all t-tuples of parameters in $\vec{P}$ must be covered by the array after replacing shielded values with "#"s.

Thus the theorem is proved by summarizing (a) and (b). □

According to Theorem 2, an MCAS of strength t with r shielding parameters can be constructed as follows.

Algorithm 1 Conversion Method

1: Compute the MCA of strength $r+t$ without considering the shielding parameters;
2: Replace invalid entries of the array by "#"s according to relationship set R;
3: Remove redundant data from the array;

In fact, we have already used the conversion method and constructed an MCAS in Section 3.3. According to Theorem 2, Table 4 is an instance of $MCAS(8; 2, 5, 2^5; R)$, where $R = \{\langle 1, \{0\}, \{2, 3, 4\}\rangle\}$. The conversion method is appropriate for occasions in which only traditional tools are available. The disadvantage is that it increases the global strength in the first step, which may give rise to a large number of redundant test cases because interactions between many parameters (like interactions between ordinary parameters) are tested in a higher strength ($t + r$ rather than t). Experimental results are given in Section 6.

5.2 Branching Method

Any shielding parameter p_i can take either shielding values or non-shielding values, and only if p_i takes shielding values will its dependent parameters be disabled. For an SUT with r shielding parameters, the branching method (see Algorithm 2) partitions the whole problem into 2^r different branches (subproblems), each of which has a different set of effective parameters, and then it solves each branch as a separate MCA problem.

For example, let p_i^+ denote that p_i's domain is restricted to its non-shielding values, and let p_i^- denote that p_i's domain is restricted to its shielding values, then $MCAS(30; 2, 5, 2^2 3^3; R)$ where $R = \{\langle 1, \{0\}, \{4\}\rangle, \langle 3, \{1, 2\}, \{2\}\rangle\}$ is constructed in the following four branches.

- (1) Parameter set: $\{p_0^+, p_1, p_2^+, p_3, p_4\}$
  - $MCA(6; 2, 5, 1^2 2^1 3^1)$
- (2) Parameter set: $\{p_0^-, p_1, p_2^+, p_4\}$
  - $MCA(9; 2, 4, 1^1 2^1 3^2)$
- (3) Parameter set: $\{p_0^+, p_2^-, p_3, p_4\}$
  - $MCA(9; 2, 4, 2^2 3^2)$
- (4) Parameter set: $\{p_0^-, p_2^-, p_4\}$
  - $MCA(6; 2, 3, 1^1 2^1 3)$

Finally, merge the above four MCA's into an array parameter to parameter, fill all blank entries with "#"s, and then we get the required MCAS.

Algorithm 2 Branching Method

1: **for** $i = 0$ to 2^r-1 **do**
2: establish an empty parameter set *PSet*.
3: **for** $j = 1$ to r **do**
4: **if** the j-th bit of i equals 1 **then**
5: add the j-th shielding parameter to *PSet*, and delete non-shielding values from its domain;
6: **else**
7: add all dependent parameters of the j-th shielding parameter p to *PSet*; add p to to *PSet*, and delete shielding values from its domain.
8: **end if**
9: **end for**
10: add ordinary parameters to *PSet*.
11: compute MCA_i of strength t for *PSet*.
12: **end for**
13: merge MCA_i ($i = 0, 1, \ldots, 2^r$-1) to an array.
14: **return** the array

Theorem 3. *The array returned by the branching method (described in Algorithm 2) is an MCAS of strength t.*

Proof. Here, we choose to use Definition 5. Clearly, property (1) and (2) hold for the result of the branching method. The solution will be proved if we can show that the result covers all complete t'-tuples, where $t' \leq t$.

Without loss of generality, let $\vec{v} = \langle a_0, a_1, \ldots, a_{t'-1}\rangle$ denote an complete t'-tuple. Because these 2^r branches include all possible effective parameter sets for the system, the set of these t' parameters should belong to at least one of the branches. Since the intermediate MCA in each branch is of strength t (which is not smaller than t'), it must have covered $\vec{v}$. For instance, suppose p_0 is an shielding parameter with shielding value 0 and 1, p_1 is an ordinary parameter, and p_3 is an dependant parameter of p_4, the complete 3-tuple $\langle 1, 0, 1\rangle$ of p_0, p_1 and p_2 is covered in branches where p_0 takes shielding values and p_4 takes non-shielding values.

Consequently, we are led to the conclusion. □

5.3 Conjunctive Branching Method

In the branching method, each branch is deemed as an independent MCA instance. This makes it convenient to use off-the-shelf tools, but it will bring redundant test cases because combinations of many parameters are considered more than once. For instance, in the above example, $\langle p_3, p_4\rangle$ are considered in both branch (1) and (3). Thus, we propose the conjunctive branching algorithms by eliminating such duplicated considerations to construct the MCAS with fewer test cases.

Algorithm 3 *Branch(PSet, visited, t)*

```
1: establish an empty set TestSet
2: add all value combinations of the first t parameters in
   PSet to TestSet
3: for i = t + 1 to |PSet| do
4:    for all parameter combinations P⃗ such that t − 1 pa-
      rameters are among the first i − 1 parameters of PSet,
      and the other is the i-th one do
5:       if P⃗ ∉ visited then
6:          add all value combinations of P⃗ to VSet
7:          add P⃗ to visited
8:       end if
9:    end for
10:   for all ⟨v1, v2, ..., vi−1⟩ ∈ TestSet do
11:      select vi from Di such that ⟨v1, v2, ..., vi−1, vi⟩
         covers most elements of VSet
12:      change ⟨v1, v2, ..., vi−1⟩ to ⟨v1, v2, ..., vi−1, vi⟩
13:      remove all elements covered by ⟨v1, v2, ..., vi⟩
         from VSet
14:   end for
15:   for all V⃗ ∈VSet do
16:      if there exists test in TestSet which can cover V⃗ by
         instantiating its unassigned parameters then
17:         instantiate test to cover V⃗
18:         remove newly covered value combinations by
            test from VSet
19:      end if
20:   end for
21: end for
22: return TestSet
```

The main procedure is the same as Algorithm 2 except for line 11. In the branching method, this line returns an MCA. While in the conjunctive branching method, it constructs an array for each branch by Algorithm 3 instead, which uses a quasi-IPOG method with horizontal extensions (line 4 to 14) and vertical extensions (line 15 to 20). Branches are dealt with successively, and a global set denoted by $visited$ is maintained as a bridge of these branches. Compared with the standard IPOG algorithm (refer to Lei's work [10, 9]), the main difference of Algorithm 3 is from line 5 to line 7, $i.e.$, for any parameter combination $\vec{P}$, we need to check out whether it is already put to $visited$. Only if $\vec{P}$ is not in $visited$, should its value combinations be put into the uncovered *VSet*. Whenever we have covered value combinations of $\vec{P}$, it is put into the $visited$, and thus, it will never be re-considered in following branches. Additionally, using the same idea of a $visited$ set, we can also implement the conjunctive branching method based on other algorithms like meta-heuristic algorithms.

6 Experimental Results

In our experiments, three kinds of shielding parameters are considered, among which "A" denotes a shielding parameter that has only one dependent parameter, "B" denotes a shielding parameter that has two dependent parameters, and "C" denotes a shielding parameter that has three dependent parameters.

6.1 Coverage of MCA's

The motivation of t-way combinatorial testing is to cover all possible complete t'-tuples ($t' \leq t$) by at least one test case. In this subsection, we focus on how shielding parameters affect coverage of test cases generated by the previous test model MCA. Inspired by the ATP system, we choose an SUT with 17 boolean parameters[2]. We generate 11 test cases by constructing $MCA(11; 2, 17, 2^{17})$ for the 2-way testing. Then, assuming there exist 1 to 4 shielding parameters (twelve conditions: A/B/C×1/2/3/4) in the SUT, we collect the corresponding coverage data for all legal complete 2-tuples of the whole input space in Figure 1. Meanwhile, we collect more details of coverage data of $MCA(24; 3, 17, 2^{17})$ for 3-way testing in Table 7, where columns of "all" and "cved." show the exact number of complete 2/3-tuples that should be covered and have been covered, respectively.

As we can see from Figure 1 and Table 7, if there exist any shielding parameters in the SUT, test cases generated by MCA of strength t can not cover all of the complete t'-tuples ($t' \leq t$). *In other words, the previous model fails in carrying out complete t-way testing for SUT's with shielding parameters*. Moreover, as the number of shielding parameters increases, the coverage decreases, and shielding parameters with more dependent paremeters can reduce the coverage more dramatically.

[2]Conclusions are also applicable to SUT's with different kinds of parameters.

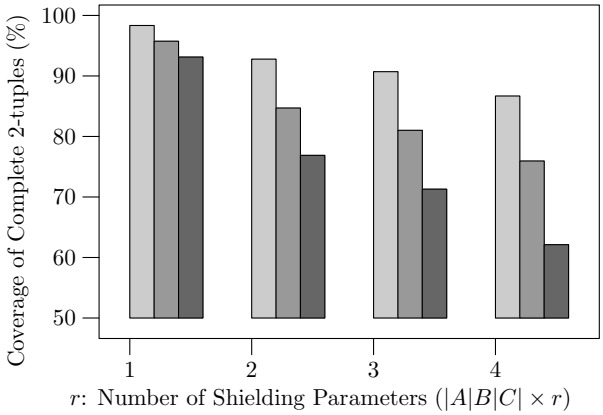

Figure 1. Coverage of $MCA(11; 2, 17, 2^{17})$

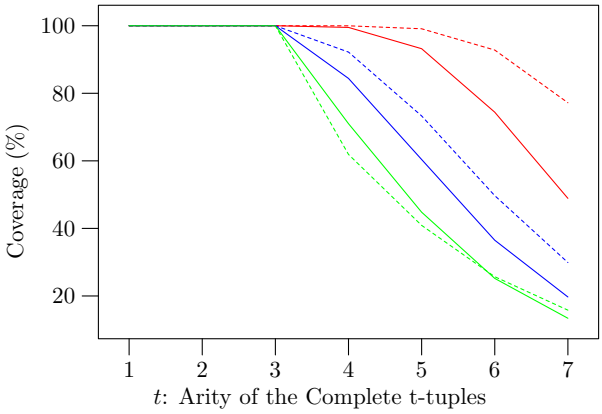

Figure 2. Coverage of MCAS

Table 7. Data of $MCA(24; 3, 17, 2^{17})$

R	Complete 2-tuples			Complete 3-tuples		
	cved.	all	rate	cved.	all	rate
A×1	542	542	100%	5294	5380	98.4%
A×2	540	540	100%	5136	5320	96.5%
A×3	537	538	99.8%	4942	5260	94.0%
A×4	533	536	99.4%	4721	5200	90.8%
B×1	540	540	100%	5170	5324	97.1%
B×2	535	536	99.8%	4821	5208	92.6%
B×3	529	532	99.4%	4430	5092	87.0%
B×4	515	528	95.2%	3939	4976	79.1%
C×1	538	538	100%	5073	5272	96.2%
C×2	531	532	99.8%	4577	5104	89.7%
C×3	514	526	97.7%	3937	4936	79.8%
C×4	499	520	96.0%	3191	4768	66.9%

6.2 Improvement by MCAS's

We use the new test model of MCAS to carry out t-way testing for the above SUT's. The conversion method (CM), branching method (BM) and conjunctive branching method (CBM) proposed in Section 5 are integrated into a prototype tool named CTES (Combinatorial TEsting with Shielding parameters). In CTES, all intermediate MCA's (for CM and BM) are constructed using the IPOG algorithm. We use CTES to generate corresponding MCAS's for the above SUT assuming there exist twelve different sets of shielding parameters as listed in the first column of Table 7. For each condition, we generate three different test suites by constructing MCAS's using CM, BM and CBM, respectively.

Experimental results illustrate that test cases generated by constructing MCAS's of strength t, in any of the three approaches, can cover every legal complete t'-tuples for $t' \leq t$. In Figure 2, we list the coverage of MCAS's with strength three whose relationship set is $R_1(B \times 2)$ or $R_2(B \times 4)$. From top to bottom, these six curves illustrate coverage of MCAS's generated by CM with R_1, CM with R_2, BM with R_1, BM with R_2, CBM with R_2 and CBM with R_1, respectively. As we can see from the figure, for complete t'-tuples where $t' \leq 3$, coverage of MCAS's generated by these algorithms is 100% (the left part of the figure). *That is to say, test cases generated by our model can successfully carry out complete t-way testing for SUT's with shielding parameters.*

Table 8. Sizes of MCAS's

R	2-way testing			3-way testing		
	CM	BM	CBM	CM	BM	CBM
A×1	24	20	13	60	49	34
B×1	24	20	13	60	48	34
C×1	24	20	13	60	47	34
A×2	60	36	14	147	94	44
B×2	60	36	14	146	89	44
C×2	60	36	14	146	85	44
A×3	146	72	18	330	177	58
B×3	146	72	18	326	163	57
C×3	144	70	18	308	148	60
A×4	330	144	23	700	334	74
B×4	316	134	23	666	287	77
C×4	275	117	23	525	233	86

For the same application, sizes of MCAS's are greater than MCA's. This is because MCA's don't take shielding parameters into account, and they are not complete test suites. Meanwhile, sizes of MCAS's generated by different algorithms are also quite different. As we can see from Table 8, since CBM eliminates many re-considerations of parameter combinations, it reduces a large number of test cases compared with BM (*esp.* if there exist many shield-

ing parameters). In most conditions, test suites generated by CM have much more test cases than BM and CBM. Besides, since CM requires to compute intermediate MCA's of a high strength $(t+r)$, it consumes more time than the other two. In our experiments, both BM and CBM cost fewer than 1s to generate each test suite. Whereas, when $t = 3$, CM costs 2s, 13s, 65s and 240s for $r = 1, 2, 3, 4$, respectively.

Furthermore, we collect coverage for complete t^*-tuples where $t^* > t$. As we can see from the right part of Figure 2, although MCAS's constructed by CM require more test cases, they can achieve higher coverage for complete t^*-tuples. Especially when $t < t^* \leq t + r$, the coverage is quite high. In contrast, MCAS's generated by CBM require fewer test cases, but they have lower coverage for complete t^*-tuples. Sizes and coverage of BM are between the two. Based on realistic test scenarios, one may choose the most suitable algorithm to generate the test cases.

7 Related Work

In the test model of MCA, values for different parameters are supposed to be independent from each other. In realistic applications, they are often required to satisfy some explicit or implicit constraints like

$$const_0 : (p_0 = 1 \wedge p_1 = 0) \rightarrow p_2 \neq 0.$$

This kind of constraints are discussed in the literature, *e.g.*, AETG [1], PICT [4] and jenny [5] provide extensions to handle or partially handle such constraints. Recently, Cohen *et al.* [2] proposed an approach to dealing with "full constraints". They compile constraints into a boolean satisfiability (SAT) problem and integrate constraints checking with both greedy and simulated annealing algorithms.

The previously studied constraints are between values of some parameters (just like $const_0$), while the shielding relationship we disscuss in this paper is another kind of constraint between some values (the shielding values) and the effectiveness of some parameters (the dependent parameters). They are different kinds of constraints in different levels. In a sense, the shielding relationship can also be written in the form of the constraints between values. For instance, given two boolean parameters p_0 and p_1, $S(p_0|\{0\}) = \{p_1\}$ can be regarded as constraints

$$(p_0 = 0 \rightarrow p_1 \neq 0) \wedge (p_0 = 0 \rightarrow p_1 \neq 1).$$

But they can not be handled by the existing tools including the claimed "full constraint". As far as we know, all of the constraints they can solve are required not to disable any parameters. For example, according to $const_0$, if p_0 is assigned 1 and p_1 is assigned 0, then p_2 can not be assigned 0. Additionally, they require there must be at least one legal value for p_2, or else, no test cases will be generated. In other words, once they generate a test suite, all of the test cases are complete tuples. Therefore, these tools and algorithms can not generate test cases for SUT's with shielding parameters, either.

8 Conclusions and Future work

Combinatorial testing is an efficient approach to detecting interaction errors for systems with several parameters. However, existing lines of research on this topic assume all parameters of the SUT are effective throughout. Motivated by practical applications, we introduce a new paradigm in combinatorial testing, where some parameters can cause other parameters to be ineffective in some conditions. We call these parameters shielding parameters. If there exist some shielding parameters, test cases generated by previous test model MCA can not expose some interaction errors that should be detected. Therefore, we propose a new model, the Mixed Covering Array with Shielding parameters (MCAS), to generate test cases for SUT's with shielding parameters, and give three approaches to constructing MCAS's, namely the conversion method, the branching method and the conjunctive branching method. According to the experimental results, our test model is appropriate for combinatorial testing with shielding parameters.

In this paper, we assume that parameters are caused to be ineffective by one shielding parameter each time. For even more complicated applications, such consequences may be caused by interactions of more than one parameters ($e.g.$, only if $p_0 = 1$ and $p_1 = 0$ is p_2 disabled). Besides, in a realistic software application, there may exist both regular constraints (discussed in Section 7) and shielding parameters. Dealing with combination of these two kinds of relationship is more difficult. We leave such problems as our future work.

Acknowledgement

This work is motivated by the application in the Signalling and Communications Research Institute of China Academy of Railway Science (CARS). The authors would like to thank M.E. Xunzheng Yin and other engineers in CARS for helping us to understand the background of SUT's used in railway industry. The authors also would like to thank Dr. Feifei Ma and the anonymous referees for their helpful comments and suggestions. This work is supported in part by the National Natural Science Foundation of China (Grant No. 60633010, 60903049) and the High-Tech (863) program of China (Grant No. 2009AA01Z148).

References

[1] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton. The AETG system: An approach to testing based on combinatorial design. *IEEE Trans. Softw. Eng.*, 23(7):437–444, 1997.

[2] M. B. Cohen, M. B. Dwyer, and J. Shi. Interaction testing of highly-configurable systems in the presence of constraints. In *ISSTA '07: Proceedings of the 2007 International Symposium on Software Testing and Analysis*, pages 129–139, New York, NY, USA, 2007. ACM.

[3] J. Czerwonka. Pairwise testing. http://www.pairwise.org.

[4] J. Czerwonka. Pairwise testing in real world: Practical extensions to test case generator. In *PNSQC '06: Proceedings of 24th Pacific Northwest Software Quality Conference*, pages 419–430, Portland, Oregon, USA, 2006.

[5] B. Jenkins. Jenny: a pairwise testing tool. http://burtleburtle.net/bob/math/jenny.html.

[6] D. R. Kuhn and M. J. Reilly. An investigation of the applicability of design of experiments to software testing. In *SEW '02: Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop (SEW-27'02)*, pages 91–95, Washington, DC, USA, 2002. IEEE Computer Society.

[7] D. R. Kuhn, D. R. Wallace, and A. M. Gallo. Software fault interactions and implications for software testing. *IEEE Trans. Softw. Eng.*, 30(6):418–421, 2004.

[8] R. Kuhn, Y. Lei, and R. Kacker. Practical combinatorial testing: Beyond pairwise. *IT Professional*, 10:19–23, 2008.

[9] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence. IPOG: A general strategy for t-way software testing. In *ECBS '07: Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 549–556, Washington, DC, USA, 2007. IEEE Computer Society.

[10] Y. Lei and K.-C. Tai. In-parameter-order: A test generation strategy for pairwise testing. In *HASE '98: The 3rd IEEE International Symposium on High-Assurance Systems Engineering*, pages 254–261, Washington, DC, USA, 1998. IEEE Computer Society.

[11] C. Nie and H. Leung. A survey of combinatorial testing. *ACM Computing Surveys*, to appear.