Automated Test Data Generation Based On Matching Combination and Testing Matrix For Pairwise Testing

Tongsen Wang  
Department of Electronic Information & Electrical Engineering  
Fujian University of Technology,  
University Town, Fujian 350108 P.R. CHINA  
tongsen@fjut.edu.cn

Abstract

Based on the model of matrix, a new algorithm for original pairwise testing data set generation is first proposed, and then, based on which, a sufficient and necessary condition for the minimum pairwise testing data set generation of the testing system $T$ is given, where $T$ has $n$ different parameters, each parameter has $n$ different values, and $n$ is a prime number. Finally, according to the given original pairwise testing data set, a novel testing data adding method based on the idea of matching combination is designed to add extra testing data to cover all the pairwise combinations that are not covered by the original pairwise testing data set. Theoretical analysis and experimental results show that, the newly proposed algorithm is simple and effective, and has these good characteristics such as small test data set and less time consumption etc.

Keywords: Pairwise testing; Automated testing data generation; Matching combination; Software Testing; Testing matrix.

1. Introduction

With the development of software developing techniques, the scale of software systems is getting bigger and the complexity of software systems is getting higher at the same time, which lead to the exposure of damages caused by software faults. So, the quality of software is becoming one of the bottle necks that restrict the development of computer applications. In the process of software testing, the testing data generation is one of the key difficult tasks. According to statistics, designing suitable testing cases covers near to 40% of the total costs of software testing [1]. Combinatorial testing is one kind of important software testing methods, in which, all factors in the software system and the impacts possibly caused by the mutual interactions of those factors are considered sufficiently, and as less as possible testing data will be generated to cover as many as possible factors that may impact the software system according to the actual needs, and at the same time, these incomplete testing results can reflect the inner rules of the complete testing, that is to say, it is representative. So, combinatorial testing is useful for detection of the software faults caused by mutual interactions of some factors in the software system.

According to different covering degrees, combinatorial testing can be divided into different categories such as single factor covering testing, pairwise combinatorial testing etc [2-4]. Currently, empirical results show that pairwise combinatorial testing (or pairwise testing) is practical and effective for various types of software system, and can detect some software faults that are not easy to be found by traditional testing methods [5-7]. However, the question of automated testing data generation for pairwise testing is not well solved yet [8-12].
Pairwise testing is a specification based testing criterion, which requires that for each pair of input parameters of a software system, every combination of valid values of any two parameters can be covered by at least one testing case. As for the question of automated generation of pairwise testing data, Cohen and his colleagues \cite{9,10} proposed a kind of heuristic generation method of pairwise testing data, and developed the corresponding testing data automated generation system named AETG. And later, Tai and Lei \cite{5} proposed another kind of testing data automated generation method for pairwise testing based on the greedy algorithm and gradual expansion of parameters according to the order of parameters in the software system, and developed the corresponding testing data automated generation system named PAIRTEST too. In \cite{8}, R Mandl proposed a new approach to generate a pairwise testing set by using orthogonal arrays, which requires that all parameters have the same number of values and each pair of values can be covered by the same number of times.

As an important supplement to the heuristic methods, Kobayashi \cite{6} proposed a kind of algebra method to generate testing data automatically for pairwise testing, which can be more efficient in some cases than heuristic methods and greedy methods. At the same time, Williams \cite{11} proposed another kind of automated pairwise testing data generation algorithm based on the method of algebra, which can also generate testing data automatically for pairwise testing effectively. In 2001, Schroeder \cite{12} proposed a reduction way of pairwise testing data based on the accessional information of relations between inputs and outputs of the software system, which can reduce the number of testing data efficiently and will not reduce the faults detecting ability at the same time. Since the problem of reduction of testing data set is NP hard, and extra costs for obtaining accessional information is also needed, the method proposed by Schroeder is not universally effective, and even less effective than using combinatorial testing directly for some time.

On the basis of those above previous research results, this paper proposed a kind of new automated data generation algorithm for pairwise testing based on the method of matrix and combination matching, which includes two major steps. In the first step, an original testing data set will be generated based on the model of matrix. And based on which, a sufficient and necessary condition for the minimum pairwise testing data set generation of the testing system $T$ is given, where $T$ has $n$ different parameters, each parameter has $n$ different values, and $n$ is a prime number. In the second step, a novel testing data adding method based on the idea of matching combination is designed to add extra testing data to cover all the pairwise combinations that are not covered by the original pairwise testing data set. Theoretical analysis and experimental results show that, the newly proposed algorithm is simple and effective, and has these good characteristics such as small test data set and less time consumption etc.

The paper is organized as follows. Section 2 introduces basic theorems of combinatorial testing. In section 3, we presented the two major steps of our new algorithm. In Section 4 we conduct some simulation experiments to support our conclusions, and finally we conclude the paper in section 5.

2. Theorem of Combinatorial Testing

The theorem of combinatorial testing can be described as follows:

**Definition 1** (testing data set): Supposing that the testing system $T$ has $n$ different input parameters such as $p_1, p_2, \ldots, p_n$, and each input parameter $p_i$ has different input domain named $D_i$, $D_i = \{ c_{i1}, c_{i2}, \ldots, c_{im_i} \}$, that is to say, each input parameter $p_i$ has $m_i$ different values. Let the set $C = \{ c_1, c_2, \ldots \}$, and if for any $c_i \in C$, there is $c_i = \{ c_{i1}, c_{i2}, \ldots, c_{im_i} \}$, and $c_{ij} \in D_j (j \in [1,n])$, then
the set C is called a testing data set of the testing system T, and c_i is a piece of testing data of the testing system T.

Definition 2 (pairwise testing data set): Supposing that the testing system T has n different input parameters such as p_1,p_2,...,p_n, and each input parameter p_i has different input domain named D_i||D_i||=m_i, (n ≥ i ≥ 1), that is to say, each input parameter p_i has m_i different values. Let (x,y) represent any pairwise combination of values from two different input parameters of testing system T, if set C is a testing data set of the testing system T, and there exists at least one c_i∈C, which satisfies x∈c_i and y∈c_i, then the set C is called a pairwise testing data set of the testing system T.

In order to illustrate the concept of the above combinatorial testing method, a testing system T_1 with input parameters and different values for each parameter is given as an example, where T_1 has three input parameters named A,B,C respectively, and the values of A,B,C are shown as below:

1. Parameter A has two different values such as A_1 and A_2.
2. Parameter B has two different values such as B_1 and B_2.
3. Parameter C has three different values such as C_1, C_2 and C_3.

As for those above parameters A,B,C, it is obvious that the set \{(A_1,B_1,C_1), (A_1,B_1,C_2), (A_1,B_2,C_1), (A_1,B_2, C_2), (A_2,B_1,C_1), (A_2,B_1,C_2), (A_2,B_2,C_1), (A_2,B_2, C_3), (B_1,C_1), (B_1,C_2), (B_1,C_3), (B_2,C_1), (B_2,C_2),(B_2,C_3),(B_3,C_1), (B_3,C_2),(B_3,C_3),(C_1),(C_2),(C_3)\} has covered all 16 different pairwise combinations existed in the testing system T_1.

It is easy to prove by using enumeration method that these three testing data sets Ω_1, Ω_2, Ω_3 given below satisfy the definition2 and are all pairwise testing sets for testing system T_1, where Ω_1, Ω_2, Ω_3 are given as follows and include 6,7,8 different testing data separately:

1. Pairwise testing set 1 (Ω_1): \{(A_1,B_1,C_1),(A_1,B_2,C_2), (A_2,B_1, C_3), (A_2,B_2,C_1), (A_2, B_1,C_2), (A_1,B_3,C_1)\}.
2. Pairwise testing set 2 (Ω_2): \{(A_1,B_1,C_1),(A_1,B_2,C_2), (A_2,B_1, C_2), (A_2,B_2,C_1), (A_1,B_2, C_3), (A_1,B_3,C_1)\}.
3. Pairwise testing set 3 (Ω_3): \{(A_2,B_1,C_1),(A_2,B_2,C_1), (A_2,B_1, C_2), (A_2,B_2,C_1), (A_2, B_1,C_2), (A_1,B_3,C_1), (A_1,B_1,C_3), (A_2,B_2, C_3)\}.

Definition 3 (minimum pairwise testing data set): Supposing that the testing system T has n different input parameters such as p_1,p_2,...,p_n, and each input parameter p_i has different input domain named D_i||D_i||=m_i, (n ≥ i ≥ 1), that is to say, each input parameter p_i has m_i different values. Let Ω_1, Ω_2, ..., Ω_p be all pairwise testing data sets for testing system T_1, and the number of testing data included in Ω_1, Ω_2, ..., Ω_p be \|Ω_1\|, \|Ω_2\|, ..., \|Ω_p\| respectively, if \|Ω_x\|=min{\|Ω_1\|, \|Ω_2\|, ..., \|Ω_p\|}, then Ω_x is called a minimum pairwise testing data set for testing system T.

According to the above definition 3, it is easy to know that Ω_1 is a minimum pairwise testing data set for testing system T_1 by using enumeration method.

3. Automated Test Data Generation for pairwise testing

3.1 Automated Original Testing Data Set Generation Based on Testing Matrix

3.1.1 Automated Original Testing Data Set Generation Algorithm

Definition 4 (expanded domain): Supposing that the testing system T has n different input parameters such as p_1,p_2,...,p_n, and each input parameter p_i has different input domain named D_i||D_i||=m_i, (n ≥ i ≥ 1), that is to say, each input parameter p_i has m_i different values. Let
\(m_n \leq m_{n-1} \leq \ldots \leq m_1, D_i = \{p_{i1}, p_{i2}, \ldots, p_{im_i}\}\), and for each input parameter \(P_i (i \geq 3)\), let \(D_i' = \{p_{i1}, p_{i2}, \ldots, p_{im_i}\}\), where \(p_{ij} = p_{i(j-1) \mod m_i + 1}\), \(j \in [1,n]\), that is to say, the number of elements in \(D_i (i \geq 3)\) is circularly expanded into \(m_1\), then \(D_i'\) is called the expanded domain of \(P_i\).

**Theorem 1:** Supposing that the testing system \(T\) has \(n\) different input parameters such as \(p_1, p_2, \ldots, p_n\) and each input parameter \(p_i\) has different domain named \(D_i\), \(|D_i| = m_i (n \geq i \geq 1)\), that is to say, each input parameter \(p_i\) has \(m_i\) different values. Let \(m_n \leq m_{n-1} \leq \ldots \leq m_1, D_i = \{p_{i1}, p_{i2}, \ldots, p_{im_i}\}\), then the number of different testing data existed in any minimum pairwise testing data set of the testing system \(T\) is no less than \(m_1 \ast m_2\).

**Proof:** Considering the first tow input parameters \(P_1, P_2\) firstly, it is obvious that the number of different pairwise combinations of values between \(P_1, P_2\) is \(m_1 \ast m_2\). So, the number of testing data needed to cover all these \(m_1 \ast m_2\) different pairwise combinations of values between \(P_1, P_2\) shall be no less than \(m_1 \ast m_2\), since one testing data can cover one pairwise combination of values between \(P_1, P_2\) only \(\Rightarrow\) the number of different testing data existed in any pairwise testing data set of the testing system \(T\) is no less than \(m_1 \ast m_2\) \(\Rightarrow\) the number of different testing data existed in any minimum pairwise testing data set of the testing system \(T\) is no less than \(m_1 \ast m_2\).

According the conclusion given by the above theorem 1, we can present the detailed automated generation algorithm for the original testing data set as follows:

**Algorithm 1** (OTDS, automated generation algorithm for the Original Testing Data Set):

**Input:** The \(n\) different input parameters \(P_1, P_2, \ldots, P_n\) in the testing system \(T\), and the domain \(D_i\) of each \(P_i\), where \(|D_i| = m_i(n \geq i \geq 1)\), \(m_n \leq m_{n-1} \leq \ldots \leq m_1, D_i = \{p_{i1}, p_{i2}, \ldots, p_{im_i}\}\).

**Output:** The original testing data set \(\Psi\) for the testing system \(T\).

**Step 1:** Let \(m_i = p, m_2 = q\), and construct \(p\) groups of vectors \(V_1, V_2, \ldots, V_p\), and in any group of vectors \(V_i (i \in [1,p])\), there exists \(n-2\) different vectors with \(q\) different elements such as \(V_i = \{V_{i1}, V_{i2}, \ldots, V_{i(n-2)}\}\), where \(V_{ij} = (a_{ij}^1, a_{ij}^2, \ldots, a_{ij}^q), j \in [1,n-2]\).

**Step 2:** For any group of vectors \(V_i = \{V_{i1}, V_{i2}, \ldots, V_{i(n-2)}\}\), construct a \((n-2) \ast q\) matrix \(M_i\) \((i \in [1,p])\) as follows:

\[
M_i = \begin{bmatrix}
V_{i1} \\
V_{i2} \\
\vdots \\
V_{i(n-2)}
\end{bmatrix} = \\
\begin{bmatrix}
a_{11}^1 & a_{11}^2 & \ldots & a_{11}^q \\
a_{12}^1 & a_{12}^2 & \ldots & a_{12}^q \\
\vdots & \vdots & \ddots & \vdots \\
a_{i(n-2)}^1 & a_{i(n-2)}^2 & \ldots & a_{i(n-2)}^q
\end{bmatrix}
\]

For any \(M_i\) \((i \in [1,p])\), then:

**Step 2.1:** For \(V_{i1}\), let \(V_{i1} = [t, t+1, t+2, \ldots, q, 1, 2, \ldots, i-1]\).

**Step 2.2:** For \(V_j = (a_{ij}^1, a_{ij}^2, \ldots, a_{ij}^q)\), \(\forall j \in [2,n-2]\), let: \(a_{ij}^k = (a_{i(j-1)}^k + i-2) \mod q + 1, k \in [1,q]\).
Step 3: Construct the expanded domains for each input parameters $P_1, P_2, \ldots, P_n$ separately, and then based on the expanded domains of $P_1, P_2, \ldots, P_n$, construct the $m_1 \times m_2$ (i.e. $p \times q$) matrix

$$\Psi$$ as follows: $$\Psi = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_p \end{bmatrix} = \begin{bmatrix} P_{11}P_{21}P_{31,j,1}P_{41,j,1} \cdots P_{n1,f,j,1} \\ \vdots \\ P_{11}P_{21}P_{31,j,2}P_{41,j,2} \cdots P_{n1,f,j,2} \\ \vdots \\ P_{1p}P_{2p}P_{3p,f,j,p}P_{4p,f,j,p} \cdots P_{n1,f,j,p} \end{bmatrix}$$

Where $\forall j \in [1, p]$,

$$Y_j = \begin{bmatrix} P_{11}P_{21}P_{31,j,1}P_{41,j,1} \cdots P_{n1,f,j,1} \\ P_{12}P_{22}P_{32,j,2}P_{42,j,2} \cdots P_{n2,f,j,2} \\ \vdots \\ P_{1p}P_{2p}P_{3p,f,j,p}P_{4p,f,j,p} \cdots P_{n1,f,j,p} \end{bmatrix}.$$ 

Step 4: As for the above $p \times q$ matrix $\Psi = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_p \end{bmatrix}$, $\forall Y_j (j \in [1, p])$, constructs $(n-2) \times q$ coefficient matrices $C_j$ as follows:

$$C_j = \begin{bmatrix} f_1(j,1) & f_1(j,2) & \cdots & f_1(j,q) \\ f_2(j,1) & f_2(j,2) & \cdots & f_2(j,q) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(j,1) & f_n(j,2) & \cdots & f_n(j,q) \end{bmatrix},$$ and then let $C_j = M_j$.

Step 5: Replacing the coefficients in $\Psi$ with the values of corresponding coefficients in each $C_j (j \in [1, p])$, then algorithm exits, and finally, the final matrix $\Psi$ with real values of parameters has been obtained.

3.1.2 Algorithm Analyses

In order to visualize each step of the above algorithm 1, an example based on the testing system $T_2$ will be given below, where the testing system $T_2$ has four different input parameters $P_1, P_2, P_3, P_4$, and among them, one input parameter has five different values, and for the rest three parameters, each has three different values respectively. The more detailed information is presented as follows:

1. $P_1 = \{ p_{11}, p_{12}, p_{13}, p_{14}, p_{15} \}$,
2. $P_2 = \{ p_{21}, p_{22}, p_{23} \}$,
3. $P_3 = \{ p_{31}, p_{32}, p_{33} \}$,
4. $P_4 = \{ p_{41}, p_{42}, p_{43} \}$.

Step 1: Let $n=4, p=5, q=3$, constructs 5 groups of vectors $V_1, V_2, V_3, V_4, V_5$, where $V = \{ v_{11}, v_{12} \}$, $v_{11} = (a_1^1, a_1^2, a_1^3)$, $v_{12} = (a_1^4, a_1^5, a_1^6)$, $i \in [1, 5]$.

Step 2: For each $V_1, V_2, V_3, V_4, V_5$, constructs the $2 \times 3$ matrices $M_1, M_2, M_3, M_4, M_5$ as follows respectively:
Step 3: Constructs a 5*3 matrix $\Psi_{53}$ as follows:

$$\Psi_{53} = \begin{bmatrix}
Y_1 \\
Y_2 \\
... \\
Y_5
\end{bmatrix} = \begin{bmatrix}
P_1P_2P_3f_{(1,1)}P_{4f,(1,1)} & P_1P_2P_3f_{(1,2)}P_{4f,(1,2)} & P_1P_2P_3f_{(1,3)}P_{4f,(1,3)} \\
P_1P_2P_3f_{(2,1)}P_{4f,(2,1)} & P_1P_2P_3f_{(2,2)}P_{4f,(2,2)} & P_1P_2P_3f_{(2,3)}P_{4f,(2,3)} \\
P_1P_2P_3f_{(3,1)}P_{4f,(3,1)} & P_1P_2P_3f_{(3,2)}P_{4f,(3,2)} & P_1P_2P_3f_{(3,3)}P_{4f,(3,3)} \\
P_1P_2P_3f_{(4,1)}P_{4f,(4,1)} & P_1P_2P_3f_{(4,2)}P_{4f,(4,2)} & P_1P_2P_3f_{(4,3)}P_{4f,(4,3)} \\
P_1P_2P_3f_{(5,1)}P_{4f,(5,1)} & P_1P_2P_3f_{(5,2)}P_{4f,(5,2)} & P_1P_2P_3f_{(5,3)}P_{4f,(5,3)}
\end{bmatrix}$$

Step 4: For each $Y_1$, $Y_2$, $Y_3$, $Y_4$, $Y_5$, constructs the coefficient matrices $C_1$, $C_2$, $C_3$, $C_4$, $C_5$ as follows:

- $C_1 = \begin{bmatrix} f_3(1,1) & f_3(1,2) & f_3(1,3) \\ f_4(1,1) & f_4(1,2) & f_4(1,3) \end{bmatrix} = M_1 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$
- $C_2 = \begin{bmatrix} f_3(2,1) & f_3(2,2) & f_3(2,3) \\ f_4(2,1) & f_4(2,2) & f_4(2,3) \end{bmatrix} = M_2 = \begin{bmatrix} 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$
- $C_3 = \begin{bmatrix} f_3(3,1) & f_3(3,2) & f_3(3,3) \\ f_4(3,1) & f_4(3,2) & f_4(3,3) \end{bmatrix} = M_3 = \begin{bmatrix} 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}$
- $C_4 = \begin{bmatrix} f_3(4,1) & f_3(4,2) & f_3(4,3) \\ f_4(4,1) & f_4(4,2) & f_4(4,3) \end{bmatrix} = M_4 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$
- $C_5 = \begin{bmatrix} f_3(5,1) & f_3(5,2) & f_3(5,3) \\ f_4(5,1) & f_4(5,2) & f_4(5,3) \end{bmatrix} = M_5 = \begin{bmatrix} 2 & 3 & 1 \\ 2 & 3 & 1 \end{bmatrix}$

Step 5: Finally, Replacing the coefficients in $\Psi_{53}$ with the values of corresponding coefficients in each $C_1$, $C_2$, $C_3$, $C_4$, $C_5$, then we obtain:

$$\Psi_{53} = \begin{bmatrix}
P_1P_2P_3P_4 & P_1P_2P_3P_4 & P_1P_2P_3P_4 \\
P_1P_2P_3P_4 & P_1P_2P_3P_4 & P_1P_2P_3P_4 \\
P_1P_2P_3P_4 & P_1P_2P_3P_4 & P_1P_2P_3P_4 \\
P_1P_2P_3P_4 & P_1P_2P_3P_4 & P_1P_2P_3P_4 \\
P_1P_2P_3P_4 & P_1P_2P_3P_4 & P_1P_2P_3P_4
\end{bmatrix}$$

It is easy to verify that, the above matrix $\Psi_{53}$ is a minimum pairwise testing data set for the testing system $T_2$.

**Theorem 2:** Supposing that the testing system $T$ has $n$ different input parameters such as $p_1, p_2, ..., p_n$, and each parameter has $n$ different values, let $P_i = \{p_{i1}, p_{i2}, ..., p_{in} \}, i \in [1,n]$, and $\Psi_{nn}$ be the $n*n$ matrix obtained by using algorithm 1, then, when $n$ is a prime number, $\Psi_{nn}$ forms a minimum pairwise testing data set for the testing system $T$.

**Proof:**

1. First of all, we will prove that $\Psi_{nn}$ forms a pairwise testing data set for the testing system $T$.

By using permutation and combination, it is easy to know that there exists $C_n^2 \times n*n$ different pairwise combinations of values between any pair of parameters in the testing system $T$. But $\Psi_{nn}$ can cover at most $C_n^2 \times n*n$ different pairwise combinations of values between any pair of parameters in the testing system $T$. So, we need only to prove that there is at most one same bit between any two elements in $\Psi_{nn}$, then it is obvious that $\Psi_{nn}$ can cover all $C_n^2 \times n*n$ different pairwise combinations of values between any pair of parameters in the testing system $T$.
Let $A_1, B_1$ represent two different elements in $\nu_{nn}$, where

\[
A_1 = p_{11} P_{21} p_{3f,(j,i)} P_{4f,(j,i)} \cdots p_{nf,(j,i)} \quad \text{and} \quad B_1 = P_{1m} p_{21} P_{3f,(m,l)} P_{4f,(m,l)} \cdots p_{nf,(m,l)}, \quad i \neq l \text{ or } j \neq m,
\]

And there are two same bits between $A_1$ and $B_1$.

Let $A'_1 = p_{3f,(j,i)} P_{4f,(j,i)} \cdots p_{nf,(j,i)}$ and $B'_1 = P_{3f,(m,l)} P_{4f,(m,l)} \cdots p_{nf,(m,l)}$.

Since when $i = l$, then there is $p_{21} = p_{2l}$, which means that the second bit between $A_1$ and $B_1$ is the same, and when $j = m$, then there is $p_{1j} = p_{1m}$, which means that the first bit between $A_1$ and $B_1$ is the same, so, the following conclusions can be drawn:

(i) If $i = l$ and $j \neq m$, then there is one same bit between $A'_1$ and $B'_1$.

(ii) If $i \neq l$ and $j = m$, then there is one same bit between $A'_1$ and $B'_1$.

(iii) If $i \neq l$ and $j \neq m$, then there is two same bits between $A'_1$ and $B'_1$.

Next, we will analyze the above three cases separately as follows:

(i) If $i = l$, $j \neq m$ and there is one same bit between $A'_1$ and $B'_1$, then there are $A'_1 = p_{3f,(j,i)} P_{4f,(j,i)} \cdots p_{nf,(j,i)}$ and $B'_1 = P_{3f,(m,l)} P_{4f,(m,l)} \cdots p_{nf,(m,l)}$.

Since there is one same bit between $A'_1$ and $B'_1$, then we can suppose that there is $f_k(j,i) = f(k,m,i)$, where $j \neq m$, $k \geq 3$. From the step 4 of algorithm 1, we can know that $f_k(j,i)$ is the element at the $(k-2)$-th row and $i$-th column in the matrix $M$, and $f_k(m,i)$ is the element at the $(k-2)$-th row and $i$-th column in the matrix $M_n$. So, from the step 2 of algorithm 1, we can know that $f_k(j,i)$ is the $i$-th element of $\nu_{(k-2)}$, and $f_k(m,i)$ is the $i$-th element of $\nu_{(k-2)}$. \( \Rightarrow f_k(j,i) = d'_j(k-2), f_k(m,i) = d'_m(k-2) \).

Since $n$ is a prime number, so, from the step 2.1 of algorithm 1, we can know that there are $d'_j = (j+i-1) \mod n$, $d'_m = (m+i-1) \mod n$. In addition, since $j \neq m$, then we can assume that there is $j < m \Rightarrow d'_j = (d'_j + (m-j) \mod n) \mod n$, then from the step 2.2 of algorithm 1, we can know that there are $d'_j = (d'_j + j-2) \mod n + 1$, and $d'_m = (d'_m + m-2) \mod n + 1 \Rightarrow d'_j = (d'_j + 2(m-j) \mod n) \mod n \Rightarrow \cdots \Rightarrow d'_m(k-2) = (d'_j(k-2) + (k-2)(m-j) \mod n).$

So, from $f_k(j,i) = f_k(m,i) \Rightarrow d'_m(k-2) = d'_j(k-2) \Rightarrow (d'_j(k-2) + (k-2)(m-j) \mod n = d'_j(k-2) \Rightarrow (k-2)(m-j) \mod n = 0 \Rightarrow (k-2)(m-j) = tn$, where $t < k-2 \Rightarrow m-j = tn(k-2)$. Since $n$ is a prime number, so, $tn(k-2)$ will not be an integer \( \Rightarrow tn(k-2) \) is not a contradict with $m-j$ since $m-j$ is an integer but $tn(k-2)$ is not an integer. \( \Rightarrow \) The conclusion 1) is not correct.

(ii) If $i \neq l$ and $j = m$, then there is one same bit between $A'_1$ and $B'_1$. Just by simulating the above proof steps, it is easy to prove that the conclusion 2) is not correct too.

(iii) If $i \neq l$ and $j \neq m$, then there are two same bits between $A'_1$ and $B'_1$, then we can assume that there are $f_k(j,i) = f_k(m,l)$ and $f_s(j,i) = f_s(m,l)$, where $k \neq s$, $k \geq 3$ and $s \geq 3$.

Since $k \neq s$, $i \neq l$, $j \neq m$, then we can assume that there are $k < s$, $i < l$, $j < m$. Just by simulating the above proof steps, it is easy to prove that there are $f_k(j,i) = d'_j(k-2), f_k(m,l) = d'_m(k-2), f_s(j,i) = d'_j(k-2), f_s(m,l) = d'_m(k-2)$. From $f_k(j,i) = f_k(m,l) \Rightarrow$ There is $d'_m(k-2) = d'_j(k-2)$, where $d'_j(k-2) = (d'_j(k-2) + (k-2)(m-j) \mod n) \mod n$. From the step 2.1 of algorithm 1, we can know that there is $d'_j(k-2) = (d'_j(k-2) + l-i + (k-2)(m-j) \mod n) \mod n = l-i + (k-2)(m-j) = l-i$.
\[(s-2)(m-j)+n, \text{ where } t < k \Rightarrow (k-s)(m-j)=tn \Rightarrow m-j=tn/(k-s). \text{ Since } n \text{ is a prime number, so, } \]
\[tn/(k-s) \text{ will not be an integer } \Rightarrow tn/(k-s) \text{ is contradict with } m-j \text{ since } m-j \text{ is an integer but } \]
\[tn/(k-2) \text{ is not an integer. } \Rightarrow \text{ The conclusion 3) is not correct.} \]

So, we have proved that for any two different elements \(A_1, B_1\) in \(\Psi_{nn}\), there is at most one same bit between them. \(\Rightarrow \Psi_{nn}\) can cover \(C^2_n*n*n\) different pairwise combinations of values between any pair of parameters in the testing system \(T\). \(\Rightarrow \Psi_{nn}\) forms a pairwise testing data set for the testing system \(T\).

(2) Secondly, we will prove that \(\Psi_{nn}\) forms a minimum pairwise testing data set for the testing system \(T\).

Since the total number of different pairwise combinations of values between any pair of parameters in the testing system \(T\) is \(C^2_n*n*n\), let \(\Omega\) be a pairwise testing data set for the testing system \(T\), then \(\Omega\) shall cover all these \(C^2_n*n*n\) different pairwise combinations of values between any pair of parameters in the testing system \(T\). From definition 1, it is easy to know that, one testing data in \(\Omega\) can cover at most \(C^2_n\) different pairwise combinations of values between any pair of parameters in the testing system \(T\). \(\Rightarrow\) There shall be at least \(n*n\) (= \(C^2_n*n*n/ C^2_n\)) different testing data in \(\Omega\) to cover all these \(C^2_n*n*n\) different pairwise combinations of values between any pair of parameters in the testing system \(T\). \(\Rightarrow\|\Omega\| \geq n^2 \geq \|\Psi_{nn}\|\). From definition 3, it is easy to know that \(\Psi_{nn}\) forms a minimum pairwise testing data set for the testing system \(T\).

To visualize the theorem 2, we give an example based on the testing system \(T_3\) as follows, where \(T_3\) has five different input parameters \(P_1, P_2, P_3, P_4, P_5\), and each parameter has five different values. The detailed information is given below:

1. \(P_1=\{p_{11}, p_{12}, p_{13}, p_{14}, p_{15}\}\),
2. \(P_2=\{p_{21}, p_{22}, p_{23}, p_{24}, p_{25}\}\),
3. \(P_3=\{p_{31}, p_{32}, p_{33}, p_{34}, p_{35}\}\),
4. \(P_4=\{p_{41}, p_{42}, p_{43}, p_{44}, p_{45}\}\),
5. \(P_5=\{p_{51}, p_{52}, p_{53}, p_{54}, p_{55}\}\).

**Step1:** constructs 5 groups of vectors \(V_1, V_2, V_3, V_4, V_5\), where \(V_i=\{v_{i1}, v_{i2}, v_{i3}\}, v_{i1}=(a_1^i, a_2^i, a_3^i, a_4^i, a_5^i), v_{i2}=(a_1^i, a_2^i, a_3^i, a_4^i, a_5^i), v_{i3}=(a_1^i, a_2^i, a_3^i, a_4^i, a_5^i), i \in [1,5]\.\)

**Step2:** For each \(V_1, V_2, V_3, V_4, V_5\), constructs 3*5 matrixes \(M_1, M_2, M_3, M_4, M_5\) respectively as follows:

\[
M_1 = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5 \\
5 & 1 & 2 & 3 & 4
\end{bmatrix},
M_2 = \begin{bmatrix}
2 & 3 & 4 & 5 & 1 \\
3 & 4 & 5 & 1 & 2 \\
4 & 5 & 1 & 2 & 3
\end{bmatrix},
M_3 = \begin{bmatrix}
3 & 4 & 5 & 1 & 2 \\
5 & 1 & 2 & 3 & 4 \\
2 & 3 & 4 & 5 & 1
\end{bmatrix},
M_4 = \begin{bmatrix}
4 & 5 & 1 & 2 & 3 \\
2 & 3 & 4 & 5 & 1 \\
5 & 1 & 2 & 3 & 4
\end{bmatrix},
M_5 = \begin{bmatrix}
5 & 1 & 2 & 3 & 4 \\
4 & 5 & 1 & 2 & 3 \\
3 & 4 & 5 & 1 & 2
\end{bmatrix}.
\]

**Step3:** Constructs the 5*5 matrix \(\Psi_{55}\) as follows: \(\Psi_{55} = \begin{bmatrix} Y_1 \\ Y_2 \\ ... \\ Y_5 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}\).
For each $Y_1, Y_2, Y_3, Y_4, Y_5$, constructs coefficient matrices $C_1, C_2, C_3, C_4, C_5$ respectively as follows:

$$
C_1 = \begin{bmatrix}
  f_3(1,1) & f_3(1,2) & f_3(1,3) & f_3(1,4) & f_3(1,5) \\
  f_4(1,1) & f_4(1,2) & f_4(1,3) & f_4(1,4) & f_4(1,5) \\
  f_5(1,1) & f_5(1,2) & f_5(1,3) & f_5(1,4) & f_5(1,5)
\end{bmatrix}
= M_1 = \begin{bmatrix}
  1 & 2 & 3 & 4 & 5 \\
  1 & 2 & 3 & 4 & 5 \\
  1 & 2 & 3 & 4 & 5
\end{bmatrix}
$$

$$
C_2 = \begin{bmatrix}
  f_3(2,1) & f_3(2,2) & f_3(2,3) & f_3(2,4) & f_3(2,5) \\
  f_4(2,1) & f_4(2,2) & f_4(2,3) & f_4(2,4) & f_4(2,5) \\
  f_5(2,1) & f_5(2,2) & f_5(2,3) & f_5(2,4) & f_5(2,5)
\end{bmatrix}
= M_2 = \begin{bmatrix}
  2 & 3 & 4 & 5 & 1 \\
  3 & 4 & 5 & 1 & 2 \\
  4 & 5 & 1 & 2 & 3
\end{bmatrix}
$$

$$
C_3 = \begin{bmatrix}
  f_3(3,1) & f_3(3,2) & f_3(3,3) & f_3(3,4) & f_3(3,5) \\
  f_4(3,1) & f_4(3,2) & f_4(3,3) & f_4(3,4) & f_4(3,5) \\
  f_5(3,1) & f_5(3,2) & f_5(3,3) & f_5(3,4) & f_5(3,5)
\end{bmatrix}
= M_3 = \begin{bmatrix}
  3 & 4 & 5 & 1 & 2 \\
  5 & 1 & 2 & 3 & 4 \\
  2 & 3 & 4 & 5 & 1
\end{bmatrix}
$$

$$
C_4 = \begin{bmatrix}
  f_3(4,1) & f_3(4,2) & f_3(4,3) & f_3(4,4) & f_3(4,5) \\
  f_4(4,1) & f_4(4,2) & f_4(4,3) & f_4(4,4) & f_4(4,5) \\
  f_5(4,1) & f_5(4,2) & f_5(4,3) & f_5(4,4) & f_5(4,5)
\end{bmatrix}
= M_4 = \begin{bmatrix}
  4 & 5 & 1 & 2 & 3 \\
  2 & 3 & 4 & 5 & 1 \\
  5 & 1 & 2 & 3 & 4
\end{bmatrix}
$$

$$
C_5 = \begin{bmatrix}
  f_3(5,1) & f_3(5,2) & f_3(5,3) & f_3(5,4) & f_3(5,5) \\
  f_4(5,1) & f_4(5,2) & f_4(5,3) & f_4(5,4) & f_4(5,5) \\
  f_5(5,1) & f_5(5,2) & f_5(5,3) & f_5(5,4) & f_5(5,5)
\end{bmatrix}
= M_5 = \begin{bmatrix}
  5 & 1 & 2 & 3 & 4 \\
  4 & 5 & 1 & 2 & 3 \\
  3 & 4 & 5 & 1 & 2
\end{bmatrix}
$$

Step 5: Finally, replacing the coefficients in $\Psi_{5,5}$ with the values of corresponding coefficients in each $C_1, C_2, C_3, C_4, C_5$, then we obtain the $\Psi_{5,5}$ with real values as follows:

$$
\Psi_{5,5} = \begin{bmatrix}
  P_1P_2P_3P_4P_5 & P_1P_2P_3P_4P_5 & P_1P_2P_3P_4P_5 & P_1P_2P_3P_4P_5 & P_1P_2P_3P_4P_5 \\
  P_2P_3P_4P_5P_6 & P_2P_3P_4P_5P_6 & P_2P_3P_4P_5P_6 & P_2P_3P_4P_5P_6 & P_2P_3P_4P_5P_6 \\
  P_3P_4P_5P_6P_7 & P_3P_4P_5P_6P_7 & P_3P_4P_5P_6P_7 & P_3P_4P_5P_6P_7 & P_3P_4P_5P_6P_7 \\
  P_4P_5P_6P_7P_8 & P_4P_5P_6P_7P_8 & P_4P_5P_6P_7P_8 & P_4P_5P_6P_7P_8 & P_4P_5P_6P_7P_8 \\
  P_5P_6P_7P_8P_9 & P_5P_6P_7P_8P_9 & P_5P_6P_7P_8P_9 & P_5P_6P_7P_8P_9 & P_5P_6P_7P_8P_9
\end{bmatrix}
$$

It is easy to verify that $\Psi_{5,5}$ forms a minimum pairwise testing data set for the testing system $T$. From the theorem 1 and its proof steps, it is easy to prove that the following lemmas are correct.

**Lemma 1:** Supposing that the testing system $T$ has $n$ different input parameters such as $p_1, p_2, ..., p_n$, and each parameter has $n$ different values, then when $n$ is a prime number, $\Omega$ forms a minimum pairwise testing data set for the testing system $T \Rightarrow \| \Omega \| = n^2$.

**Proof:** From theorem 1, it is easy to know that $\Psi_{nm}$ forms a minimum pairwise testing data set for the testing system $T$ when $n$ is a prime number, since $\| \Psi_{nm} \| = n^2 \Rightarrow \| \Omega \| \leq n^2$. In addition, since the total number of different pairwise combinations of values between any pair of parameters in the testing system $T$ is $C_n^2 = n^2$, and $\Omega$ is a pairwise testing data set for the testing system $T \Rightarrow \Omega$ shall cover all these $C_n^2 = n^2$ different pairwise combinations of values between any pair of parameters in the testing system $T$. From definition 1, it is easy to
know that, one testing data in $\Omega$ can cover at most $C_n^2$ different pairwise combinations of values between any pair of parameters in the testing system $T$. \( \Rightarrow \) There shall be at least $n^*n = (C_n^2 * n^* n / C_n^2)$ different testing data in $\Omega$ to cover all these $C_n^2 * n^* n$ different pairwise combinations of values between any pair of parameters in the testing system $T$. \( \Rightarrow \) $||\Omega|| \geq n^2$.

So, we obtain that when $n$ is a prime number, $\Omega$ forms a minimum pairwise testing data set for the testing system $T \iff ||\Omega|| = n^2$.

**Lemma 2:** Supposing that the testing system $T$ has $n$ different input parameters such as $p_1, p_2, \ldots, p_n$ and each parameter has $n$ different values, if when $n$ is a prime number and $\Omega$ forms a minimum pairwise testing data set for the testing system $T$, then there are no redundant pairwise combinations existed in $\Omega$.

**Proof:** From Lemma 1, it is easy to know that, when $n$ is a prime number, if $\Omega$ forms a minimum pairwise testing data set for the testing system $T$, then there is $||\Omega|| = n^2$. From definition 1, we can know that one testing data in $\Omega$ can cover at most $C_n^2$ different pairwise combinations $\Rightarrow \Omega$ can cover at most $C_n^2 * n^* n$ different pairwise combinations. Since the total number of different pairwise combinations of values between any pair of parameters in the testing system $T$ is $C_n^2 * n^* n$, so, if $\Omega$ forms a minimum pairwise testing data set for the testing system $T \Rightarrow \Omega$ shall cover all these $C_n^2 * n^* n$ different pairwise combinations of values between any pair of parameters in the testing system $T$. \( \Rightarrow \) there are no redundant pairwise combinations existed in $\Omega$.

### 3.2 Testing Data Adding Based on Matching Combination

**Definition 5** (matching combination): For any combination $\{x_1, x_2, \ldots, x_n\}$, if there exists combination $\{x'_1, x'_2, \ldots, x'_n\}$, $\forall i \in [1, n]$, which satisfies one of the following conditions:

1. $x_i = x'_i$
2. $x_i = “*”$
3. $x'_i = “*”$

Then, $\{x'_1, x'_2, \ldots, x'_n\}$ is called a matching combination of $\{x_1, x_2, \ldots, x_n\}$.

**Definition 6** (combinatorial operator $\oplus$): For any combination $\{x_i, x_{i_2}, \ldots, x_{i_n}\}$, supposing that $\{x'_i, x'_{i_2}, \ldots, x'_{i_n}\}$ is a matching combination of $\{x_{i_1}, x_{i_2}, \ldots, x_{i_n}\}$, then we define that $\{x_i, x_{i_2}, \ldots, x_{i_n}\} \oplus \{x'_i, x'_{i_2}, \ldots, x'_{i_n}\} = \{x_1 \oplus x'_1, x_2 \oplus x'_{i_2}, \ldots, x_n \oplus x'_{i_n}\}$, where $\forall i \in [1, n]$, $x_i \oplus x'_i$ is defined as follows:

1. $x_1 \oplus x'_1 = x_i$, if $x_i = x'_i$
2. $x_1 \oplus x'_1 = x_i$, if $x'_1 = “*”$
3. $x_1 \oplus x'_1 = x_i$, if $x_i = “*”$

Where “*” in the $i$-th position of combination of $\{x_i, x_{i_2}, \ldots, x_{i_n}\}$ or $\{x'_i, x'_{i_2}, \ldots, x'_{i_n}\}$ represents an unfixed value that belongs to domain $D_i$ of the input parameter $P_i$.

**Definition 7** (contribution ratio): Supposing that $\Psi$ is an original testing data set, the pairwise combinations covered by which is $S$, and $||S||$ represents the number of elements in $S$. For a newly added testing data $d$, let set $D$ represent the pairwise combinations covered by $d$, and $H_d = \{h | h \in D, h \notin S\}$, then $P_d = ||H_d||/||S||$ is called contribution ratio of the testing data $d$.

Supposing that the testing system $T$ has $n$ different input parameters such as $p_1, p_2, \ldots, p_n$, and each input parameter $p_i$ has different input domain named $D_i$, $D_i = \{m_{i_1}, m_{i_2}, \ldots, m_{i_l}\}$, and each input parameter $p_i$ has $m_i$ different values. Let $m_n \leq m_{n-1} \leq \ldots \leq m_1$, $D_i = \{p_{i_1}, p_{i_2}, \ldots, p_{i_{m_i}}\}$, and $W$ represent the set consisted with all pairwise combinations of values between any pair of parameters in the testing system $T$, then the testing data adding algorithm based on the above combination matching method can be described as follows:
Algorithm 2 (TDAMC, Testing Data Adding based on Matching Combination):

Step1: Generates the original testing data set $\Psi$ for testing system $T$ by using algorithm 1.

Step2: Supposing that these pairwise combinations that are not covered by $\Psi$ form the set $S=\{(a,b) | a,b \in \{D_1,D_2,\ldots,D_n\}, \forall i \in [1,n], a,b \text{ will not belong to } D_i \text{ at the same time, and } (a,b) \in \Psi \}$, which S can be calculated by W and $\Psi$. $\forall (a,b) \in S$, supposing that there are $a \in P_i$, $b \in P_j$, $i \neq j$, then expands $(a,b)$ to $(x_{i1},x_{i2},\ldots,x_{il},a,x_{i+1},\ldots,x_{i+k},b,x_{j+1},\ldots,x_{j+k})$, where $\forall t \in [1,2,\ldots,i-1, i+1,\ldots,j-1,j+1,\ldots,n]$, there are $x_t^{âŠ“*\Psi}$, So, we can obtain that $S'=\{(x_{i1},x_{i2},\ldots,x_{il},a,x_{i+1},\ldots,x_{i+k},b,x_{j+1},\ldots,x_{j+k}) | a,b \in \{P_1,P_2,\ldots,P_n\}, \forall t \in [1,2,\ldots,i-1, i+1,\ldots,j-1,j+1,\ldots,n], \text{there is } x_t^{âŠ“*\Psi}\}$.

Step3: $\forall X \in S'$, obtains the number of matching combinations of $X$ in $S'$. Selects one of the combinations $Y$ with the biggest number of matching combinations (and the biggest contribution ratio), supposing that all of the matching combinations of $Y$ are $Y_1, Y_2,\ldots,Y_k$, and let $Y'=Y \oplus Y_1 \oplus Y_2 \oplus \ldots \oplus Y_k$, $\Psi = \Psi \setminus \{Y\}$, $S=S'\{\text{pairwise combinations covered by } Y\}$.

Step4: Turns back to Step 2, until $S=\Phi$ (empty set), then algorithm exits.

In order to visualize the steps in algorithm 2, we give an example based on the testing system $T_4$ as follows, where $T_4$ has six different input parameters $P_1, P_2, P_3, P_4, P_5, P_6$, and each parameter has six different values too. The detailed information is given below:

(i) Steps of algorithm 1:

Step1: constructs 6 groups of vectors $V_1,V_2,V_3,V_4,V_5,V_6$, where $V_i=\{v_{i1},v_{i2},v_{i3},v_{i4}\}$, $v_{i1}=\{a_{i1}^1,a_{i1}^2,a_{i1}^3,a_{i1}^4,a_{i1}^5,a_{i1}^6\}$, $v_{i2}=\{a_{i2}^1,a_{i2}^2,a_{i2}^3,a_{i2}^4,a_{i2}^5,a_{i2}^6\}$, $v_{i3}=\{a_{i3}^1,a_{i3}^2,a_{i3}^3,a_{i3}^4,a_{i3}^5,a_{i3}^6\}$, $v_{i4}=\{a_{i4}^1,a_{i4}^2,a_{i4}^3,a_{i4}^4,a_{i4}^5,a_{i4}^6\}$, $i \in [1,6]$.

Step2: For each $V_1, V_2, V_3, V_4, V_5, V_6$, constructs 4*6 matrixes $M_1, M_2, M_3, M_4, M_5, M_6$ respectively as follows:

$M_1=\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \end{bmatrix}$, $M_2=\begin{bmatrix} 2 & 3 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 5 & 6 & 1 & 2 & 3 & 4 \end{bmatrix}$, $M_3=\begin{bmatrix} 3 & 4 & 5 & 6 & 1 & 2 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 6 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$, $M_4=\begin{bmatrix} 4 & 5 & 6 & 1 & 2 & 3 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$, $M_5=\begin{bmatrix} 5 & 6 & 1 & 2 & 3 & 4 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 4 & 5 & 6 & 1 & 2 & 3 \end{bmatrix}$, $M_6=\begin{bmatrix} 6 & 1 & 2 & 3 & 4 & 5 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 6 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$.

Step3: Constructs the 6*6 matrix $\Psi_{66}$ as follows: $\Psi_{66}=\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_6 \end{bmatrix}$.
$$\begin{bmatrix}
P_1P_2P_3P_{f(1,1)}P_{f(1,2)}P_{f(1,3)}P_{f(1,4)}P_{f(1,5)} \quad \ldots \quad P_1P_2P_3P_{f(6,1)}P_{f(6,2)}P_{f(6,3)}P_{f(6,4)}P_{f(6,5)}P_{f(6,6)}
\end{bmatrix}$$

Step 4: For each $Y_1, Y_2, Y_3, Y_4, Y_5, Y_6$, constructs coefficient matrices $C_1, C_2, C_3, C_4, C_5, C_6$ respectively as follows:

$$C_1 = \begin{bmatrix}
f_3(1,1) & f_3(1,2) & f_3(1,3) & f_3(1,4) & f_3(1,5) & f_3(1,6) \\
f_4(1,1) & f_4(1,2) & f_4(1,3) & f_4(1,4) & f_4(1,5) & f_4(1,6) \\
f_5(1,1) & f_5(1,2) & f_5(1,3) & f_5(1,4) & f_5(1,5) & f_5(1,6)
\end{bmatrix} = M_1 = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6
\end{bmatrix},$$

$$C_2 = \begin{bmatrix}
f_3(2,1) & f_3(2,2) & f_3(2,3) & f_3(2,4) & f_3(2,5) & f_3(2,6) \\
f_4(2,1) & f_4(2,2) & f_4(2,3) & f_4(2,4) & f_4(2,5) & f_4(2,6) \\
f_5(2,1) & f_5(2,2) & f_5(2,3) & f_5(2,4) & f_5(2,5) & f_5(2,6)
\end{bmatrix} = M_2 = \begin{bmatrix}
2 & 3 & 4 & 5 & 6 & 1 \\
3 & 4 & 5 & 6 & 1 & 2 \\
4 & 5 & 6 & 1 & 2 & 3
\end{bmatrix},$$

$$C_3 = \begin{bmatrix}
f_3(3,1) & f_3(3,2) & f_3(3,3) & f_3(3,4) & f_3(3,5) & f_3(3,6) \\
f_4(3,1) & f_4(3,2) & f_4(3,3) & f_4(3,4) & f_4(3,5) & f_4(3,6) \\
f_5(3,1) & f_5(3,2) & f_5(3,3) & f_5(3,4) & f_5(3,5) & f_5(3,6)
\end{bmatrix} = M_3 = \begin{bmatrix}
3 & 4 & 5 & 6 & 1 & 2 \\
5 & 6 & 1 & 2 & 3 & 4 \\
2 & 3 & 4 & 5 & 6 & 1
\end{bmatrix},$$

$$C_4 = \begin{bmatrix}
f_3(4,1) & f_3(4,2) & f_3(4,3) & f_3(4,4) & f_3(4,5) & f_3(4,6) \\
f_4(4,1) & f_4(4,2) & f_4(4,3) & f_4(4,4) & f_4(4,5) & f_4(4,6) \\
f_5(4,1) & f_5(4,2) & f_5(4,3) & f_5(4,4) & f_5(4,5) & f_5(4,6)
\end{bmatrix} = M_4 = \begin{bmatrix}
4 & 5 & 6 & 1 & 2 & 3 \\
2 & 3 & 4 & 5 & 6 & 1 \\
5 & 6 & 1 & 2 & 3 & 4
\end{bmatrix},$$

$$C_5 = \begin{bmatrix}
f_3(5,1) & f_3(5,2) & f_3(5,3) & f_3(5,4) & f_3(5,5) & f_3(5,6) \\
f_4(5,1) & f_4(5,2) & f_4(5,3) & f_4(5,4) & f_4(5,5) & f_4(5,6) \\
f_5(5,1) & f_5(5,2) & f_5(5,3) & f_5(5,4) & f_5(5,5) & f_5(5,6)
\end{bmatrix} = M_5 = \begin{bmatrix}
5 & 6 & 1 & 2 & 3 & 4 \\
4 & 5 & 6 & 1 & 2 & 3 \\
3 & 4 & 5 & 6 & 1 & 2
\end{bmatrix},$$

$$C_6 = \begin{bmatrix}
f_3(6,1) & f_3(6,2) & f_3(6,3) & f_3(6,4) & f_3(6,5) & f_3(6,6) \\
f_4(6,1) & f_4(6,2) & f_4(6,3) & f_4(6,4) & f_4(6,5) & f_4(6,6) \\
f_5(6,1) & f_5(6,2) & f_5(6,3) & f_5(6,4) & f_5(6,5) & f_5(6,6)
\end{bmatrix} = M_6 = \begin{bmatrix}
6 & 1 & 2 & 3 & 4 & 5 \\
6 & 1 & 2 & 3 & 4 & 5 \\
6 & 1 & 2 & 3 & 4 & 5
\end{bmatrix}.$$
of algorithm 2:

**Step2:** Generates the original testing data set $\Psi_{66}$ for testing system $T_4$ by using algorithm 1.

**Step2:** Obtains the set $S$ consisted with these pairwise combinations that are not covered by $\Psi$, it is easy to verify that $S=\{(p_{31},p_{44}), (p_{32},p_{45}), (p_{33},p_{46}), (p_{34},p_{41}), (p_{35},p_{42}), (p_{36},p_{43})\} \cup \{(p_{31},p_{34}), (p_{32},p_{35}), (p_{33},p_{36}), (p_{34},p_{31}), (p_{35},p_{32}), (p_{36},p_{33})\} \cup \{(p_{31},p_{62}), (p_{31},p_{65}), (p_{32},p_{64}), (p_{32},p_{66}), (p_{33},p_{65}), (p_{33},p_{66}), (p_{34},p_{65}), (p_{34},p_{66}), (p_{35},p_{65}), (p_{35},p_{66}), (p_{36},p_{65}), (p_{36},p_{66})\} \cup \{(p_{11},p_{53}), (p_{11},p_{55}), (p_{12},p_{45}), (p_{13},p_{46}), (p_{14},p_{43}), (p_{15},p_{42}), (p_{16},p_{41})\} \cup \{(p_{11},p_{54}), (p_{11},p_{56}), (p_{12},p_{54}), (p_{12},p_{56}), (p_{13},p_{54}), (p_{13},p_{56}), (p_{14},p_{54}), (p_{14},p_{56}), (p_{15},p_{54}), (p_{15},p_{56}), (p_{16},p_{54}), (p_{16},p_{56})\}$.

In each of these above testing data, the first “*” can be any value that belongs to \(\{p_{11},p_{12},p_{13},p_{14},p_{15},p_{16}\}\), the second “*” can be any value that belongs to \(\{p_{21},p_{22},p_{23},p_{24},p_{25},p_{26}\}\).

So, till now after adding these above 18 extra testing data to the original testing data set $\Psi_{66}$, we finally obtain a testing data set with 54 different testing data that covers all the pairwise combinations in the testing system $T_4$. 

4. Experimental Result and Theoretical Analysis

**Theorem 3:** Supposing that the testing system \( T \) has \( n \) different input parameters such as \( p_1, p_2, \ldots, p_n \) and each input parameter \( p_i \) has different input domain named \( D_i \) such that, that is to say, each input parameter \( p_i \) has \( m_i \) different values. Let \( m_{i1} \leq m_{i2} \leq \ldots \leq m_{i1} \), \( D_i = \{ p_{i1}, p_{i2}, \ldots, p_{im_i} \} \), then, the time costs of the original testing data set generation by using algorithm 1 are \( O(n^* m_1^* m_2) \).

**Proof:**

(1) In the step 1 of algorithm 1, the time costs of constructing a group of \( n-2 \) different vectors with \( q \) different elements are \( O(n^* q) \), so the time costs of constructing \( p \) groups of such kind of vectors are \( O(n^* p^* q) \).

(2) In the step 2 of algorithm 1, the time costs of constructing a \( (n-2)^* q \) matrix \( M \) are \( O(n^* q) \), and the time costs of giving values to matrix \( M \) are \( O(n^* q) \) also, then, the time costs of constructing and giving values to \( p \) such kind of matrixes are \( O(n^* p^* q) \).

(3) In the step 3 of algorithm 1, the time costs of constructing a \( p^* q \) matrix \( \Psi \) are \( O(n^* p^* q) \), since the length of every element in \( \Psi \) is \( n \).

(4) In the step 4 of algorithm 1, the time costs of constructing a \( (n-2)^* q \) coefficient matrix \( C_i \) are \( O(n^* q) \), and the time costs of giving values to matrix \( C_i \) are \( O(n^* q) \) also, then, the time costs of constructing and giving values to \( p \) such kind of matrixes are \( O(n^* p^* q) \).

(5) In the step 5 of algorithm 1, the time costs of replacing the coefficients in \( \Psi \) with the values of corresponding coefficients in each \( C_i (j \in [1, p]) \) are \( O(n^* q) \), since there are \( p \) such kind of matrixes \( C_i \), then the total time costs are \( O(n^* p^* q) \).

Since there are \( p = m_1 \) and \( q = m_2 \), so, we proved that the time costs of the original testing data set generation by using algorithm 1 are \( O(n^* m_1^* m_2) \).

**Theorem 4:** Supposing that the testing system \( T \) has \( n \) different input parameters such as \( p_1, p_2, \ldots, p_n \) and each input parameter \( p_i \) has different input domain named \( D_i \) such that, that is to say, each input parameter \( p_i \) has \( m_i \) different values. Let \( m_{i1} \leq m_{i2} \leq \ldots \leq m_{i1} \), \( D_i = \{ p_{i1}, p_{i2}, \ldots, p_{im_i} \} \), then the time costs of the testing data addition by using algorithm 2 are no more than \( O(n^* m_1^2 * m_2^2) \).

**Proof:**

(1) In the step 1 of algorithm 2, from theorem 3, it is easy to know that the time costs of constructing the original testing data set are \( O(n^* m_1^* m_2) \).

(2) In the step 2 of algorithm 2, the time costs of constructing the pairwise combination between any pair of parameters \( P_i \) and \( P_j \) are \( O(m_i^* m_j) \) \( \leq O(m_1^* m_2) \). Since there are \( C_n^2 = \frac{n(n-1)}{2} \) different pairs of parameters in the testing system \( T \), then the total time costs of constructing all pairwise combinations in the testing system \( T \) are no more than \( O(n^2 m_1^* m_2) \). And similarly, we can prove that the total time costs of constructing all pairwise combinations in the original testing data set \( \Psi \) are \( O(n^2 m_1^* m_2) \) also. So, the time costs of obtaining the set \( S \) of pairwise combinations that are not covered by \( \Psi \) are no more than \( O(n^2 m_1^2 * m_2^2) \). In addition, since the number of elements in \( S \) will not exceed the number of pairwise combinations in the testing system \( T \), which means that the number of elements in \( S \) is no more than \( C_n^2 m_1^* m_2^2 \), so the time costs of constructing \( S' \) are no more than \( O(n^* m_1^* m_2) \).

(3) In the step 3 of algorithm 2, since the number of elements in \( S' \) is no more than \( C_n^2 m_1^* m_2^2 \), then the time costs of obtaining the pairwise combinations included by one element of \( S' \) are no more than \( O(n^2 m_1^* m_2) \), which means that the total time costs of obtaining the pairwise combinations included by all elements in \( S' \) are no more than
\(O(n^4 \cdot m_1^2 \cdot m_2^2)\). It is obvious that the time costs of selecting the biggest pairwise combinations from \(S'\) will be no more than \(O(n^4 \cdot m_1^2 \cdot m_2^2)\) also.

So, from the above analyses, we proved that the time costs of the testing data addition by using algorithm 2 are no more than \(O(n^4 \cdot m_1^2 \cdot m_2^2)\).

In order to verify the performances of our newly propose methods in this paper, some experiments are done and the experimental environments are as follows: Hardware platform: DELL, CPU: Celeron 2.40GHz, main memory space: 1GB, Operation system: Windows Xp, Coding language: Visual C++ 6.0.

In our first experiment, 8 different testing system \(\{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8\}\) are chosen, and the number of input parameters are \(\{4,5,7,11,13,17,19,23\}\) separately. The scales of different testing data sets obtained by our new methods are shown in table 1, and the time costs to obtain the testing data sets for each testing system are given in Fig.1 below:

**Table 1. Scales of testing data sets obtained by our new methods.**

<table>
<thead>
<tr>
<th>Testing system</th>
<th>(number of parameters, number of values of each parameter)</th>
<th>Obtained Minimum testing data set by our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_1)</td>
<td>(4,4)</td>
<td>16</td>
</tr>
<tr>
<td>(T_2)</td>
<td>(5,5)</td>
<td>25</td>
</tr>
<tr>
<td>(T_3)</td>
<td>(7,7)</td>
<td>49</td>
</tr>
<tr>
<td>(T_4)</td>
<td>(11,11)</td>
<td>121</td>
</tr>
<tr>
<td>(T_5)</td>
<td>(13,13)</td>
<td>169</td>
</tr>
<tr>
<td>(T_6)</td>
<td>(17,17)</td>
<td>289</td>
</tr>
<tr>
<td>(T_7)</td>
<td>(19,19)</td>
<td>361</td>
</tr>
<tr>
<td>(T_8)</td>
<td>(23,23)</td>
<td>529</td>
</tr>
</tbody>
</table>

**Fig. 1. Time costs to obtain the testing data sets for each testing system**

In the above Fig.1, the horizontal axis represents the scale of the testing systems \(\{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8\}\), i.e. the number of input parameters of \(\{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8\}\), and the vertical axis represents the time costs (with unit of second) of obtaining the minimum testing data set by our methods.

In our second experiment, 11 different testing systems \(\{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}\}\) are chosen to compare the performance of our methods with the methods proposed in [5] and [10]. The comparison result about the scale of testing data set obtained by different methods are shown in table 2, and the time costs of each methods to obtain the testing data

35
set are shown in Fig.2. And in Table 2, the 11 different testing systems \( \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}\} \) are defined as follows:

1. \( S_1 \): 3 different parameters (2 parameters have 2 different values each, and the rest 1 parameter has 3 different values)
2. \( S_2 \): 4 different parameters (each parameter has 3 different values)
3. \( S_3 \): 11 different parameters (each parameter has 11 different values)
4. \( S_4 \): 12 different parameters (each parameter has 12 different values)
5. \( S_5 \): 13 different parameters (each parameter has 13 different values)
6. \( S_6 \): 20 different parameters (each parameter has 12 different values)
7. \( S_7 \): 38 different parameters (16 parameters have 2 different values each, 12 parameters have 3 different values each, 10 parameters have 4 different values each)
8. \( S_8 \): 61 different parameters (15 parameters have 4 different values each, 17 parameters have 3 different values each, 29 parameters have 2 different values each)
9. \( S_9 \): 75 different parameters (1 parameter has 4 different values each, 39 parameters have 3 different values each, 35 parameters have 2 different values each)
10. \( S_{10} \): 80 different parameters (each parameter has 3 different values)
11. \( S_{11} \): 100 different parameters (each parameter has 2 different values)

<table>
<thead>
<tr>
<th>Testing system</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
<th>( S_4 )</th>
<th>( S_5 )</th>
<th>( S_6 )</th>
<th>( S_7 )</th>
<th>( S_8 )</th>
<th>( S_9 )</th>
<th>( S_{10} )</th>
<th>( S_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>AETG</td>
<td>7</td>
<td>11</td>
<td>153</td>
<td>182</td>
<td>217</td>
<td>293</td>
<td>32</td>
<td>35</td>
<td>25</td>
<td>28</td>
<td>12</td>
</tr>
<tr>
<td>PairTest</td>
<td>6</td>
<td>9</td>
<td>169</td>
<td>238</td>
<td>285</td>
<td>312</td>
<td>30</td>
<td>34</td>
<td>26</td>
<td>32</td>
<td>15</td>
</tr>
<tr>
<td>Without</td>
<td>6</td>
<td>9</td>
<td>121</td>
<td>177</td>
<td>169</td>
<td>286</td>
<td>29</td>
<td>31</td>
<td>24</td>
<td>27</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contribution ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>With contribution ratio</td>
</tr>
</tbody>
</table>

From the above table 2, it is easy to know that our methods are much better than the AETG and PairTest proposed in [5, 11], and generally, the performance of our methods when the contribution ratio is considered will be better than the performance when the contribution ratio is not considered. But in the above table 2, when the testing data are generated for the testing system \( S_8 \), the performance of our methods when the contribution ratio is considered is worse than the performance when the contribution ratio is not considered. After analyzing, we find the reason is that the length of latter combinations may be affected when we consider the contribution ratio to add the extra testing data. In addition, the number of input parameters and the number of different values for each parameter in \( S_4 \) are all smaller than that of \( S_5 \), but the scale of obtained testing data set for \( S_4 \) is bigger than that of \( S_5 \). After analyzing, we find the reason is that the testing system \( S_5 \) satisfies the conditions in theorem 2, needs not to use algorithm 2 to add extra testing data, and there are no redundant pairwise combinations in the obtained testing data set too. But the testing system \( S_4 \) does not satisfy the theorem 2, needs to use algorithm 2 to add extra testing data, and there are many redundant pairwise combinations in the obtained testing data set.

Fig.2 illustrates the time costs needed to generate the testing data set for \( \{S_1, S_2, S_3, S_4, S_5, S_6\} \) by using our methods and the methods proposed in [5,10].
From Fig. 2, it is obvious that our methods have less time costs than AETG. After analyzing, we find the reason is that our methods adopt a new way to generate the original testing data set and can reduce the number of pairwise combinations that are not covered by original testing data set efficiently, which leads to less time costs than AETG. Compared with PairTest, both our methods and PairTest have their own advantages and disadvantages in time costs. After analyzing, we find that our methods will be better than PairTest in time costs when the testing systems satisfy the conditions in theorem 2, and otherwise, PairTest will be better than our methods in time costs. And the major reason is that when a testing system satisfies the conditions in theorem 2, it needs not to use algorithm 2 to add extra testing data, so we can know that the time costs will be reduced sharply according to theorem 3 and theorem 4.

5. Conclusion and Future Works

Based on the model of matrix, a new algorithm for original pairwise testing data set generation is proposed firstly, and then, based on which, a sufficient and necessary condition for the minimum pairwise testing data set generation of the testing system T is given, where T has n different parameters, each parameter has n different values, and n is a prime number. Finally, according to the given original pairwise testing data set, a novel testing data adding method based on the idea of matching combination is designed to add extra testing data to cover all the pairwise combinations that are not covered by the original pairwise testing data set. Theoretical analysis and experimental results show that, the newly proposed algorithm is simple and effective, and has these good characteristics such as small test data set and time consumption etc.. For future works, we aim to improve the performance of our methods, especially the performance of the testing data addition methods.

Acknowledgments
This work was supported in part by the Natural Science Foundation of Fujian Province of China (No. 2008J0178) and the Research Fund of Fujian University of Technology (No.GY-Z10050).

References


