Research on the Pairwise Test Case Generation of Two-Dimensional Expansion

Yuan Shuai
School of Computer Science and Engineering
Beihang University
Beijing, China
yuanshuai.deny@gmail.com

Ye Gang
School of Computer Science and Engineering
Beihang University
Beijing, China
gang.ye@cs2c.com.cn

Cui Jingyan
School of Computer Science and Engineering
Beihang University
Beijing, China
jingyan1949@126.com

Ma Shilong
School of Computer Science and Engineering
Beihang University
Beijing, China
slma@nlsde.buaa.edu.cn

***Abstract*—The recently widely-spreading usage of combinatorial interaction testing is dramatically improving the effectiveness of highly-configurable software. Conventional techniques based on greedy or heuristic algorithms can lead to suboptimal result in the size of the built test suite with unstability. In this paper, a strategy for the construction of pairwise covering test cases is presented on the basis of research on previous 2-dimensional expansion algorithm to eliminate randomness and optimize efficiency that caused by itself. The proposed approach IPO_S_R is supported by symmetry property and lower bound theory. In addition, experimental assessment is also presented.**

Keywords: combinatorial interaction testing; IPO_S_R; restriction; 2-dimensional expansion

I. INTRODUCTION

Program families are defined (analogously to hardware families) as sets of programs whose common properties are so extensive that it is advantageous to study the common properties of the programs before analyzing individual members[1]. This kind of situation puts forward a challenge for testing and validation. One function depends on all components working together. Test has been switched from testing in a single software configuration to testing in all possible different optional configurations. However, a set of possible inputs for any nontrivial piece of software is too large to be tested exhaustively[2][3]. Then pairwise testing has become an indispensable way in software testing[4].

Pairwise testing requires that, for each pair of input parameters of a system, every combination of valid values of these two parameters be covered by at least one test case [5]. Over the years, a number of combinatorial strategies have been devised to help testers choose subsets of input combinations that would maximize the probability of detecting defects[6]. A large part of these strategies are based on use of the traditional constraint solving, optimization methods or directly search covering arrays[7]. Due to the complexity of the problem is NP complete, most of the methods are local search algorithm, these methods can't guaranteed to get optimal solution, but the time is relatively less than other methods. One-test-at-a-time method and 2-dimensional expansion method are Mainstream methods. In this article, we will discuss two-dimensional expansion method and discover some problems existed in the in-parameter-order algorithm and put forward a solution for them.

II. TEST CASE GENERATION OF TWO-DIMENSIONAL EXPANSION

A. *Introduction of 2-dimensional expansion*

Lei proposes IPO algorithm, which is typical 2-dimensional expansion algorithm[8]. It firstly construct first two parameters all combinatorial, forming a minimal matrix and repeats the two steps 2-dimensional expansion. It will make vertical and horizontal expansions. In contrast with that algorithm, Calvagna and Garganitini proposed IPOs algorithm used symmetry property to expand coverage matrix[9]. In this article, we will use IPOs as prototype to analyze 2-dimensional expansion algorithm.

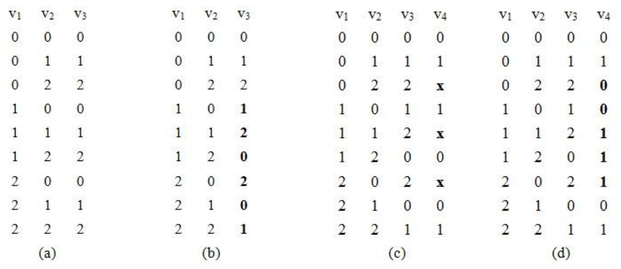
Firstly we will describe how the algorithm carries out. We assume a system with $3^3 \times 2^1$ parameters, which means 3 parameters with 3 values and 1 parameter with 2 values. According to IPOs, the parameters rank as big to small in line with the number of values and firstly covered by first two parameters v_1 and v_2. Expanded with the symmetry principle, we make the third parameter value column clone v_2 (see Fig.1-(a)). We then only need to complete pairing of v_3 and v_2. Note also that some of the pairs between v_3 and v_2 are already covered by construction:{(0,0),(1,1),(2,2)}, and that they are three times redundant. We only need to cover pairs {(0,1),(0,2),(1,0),(1,2),(2,0),(2,1)}. A simple heuristic applicable here is to edit the symbol in the row position of first redundant instance of a pair, that is, in this case, we choose to change assignment on fourth row so to form the missing pair $(v_2, v_3)=(0,1)$. Since this will delete the unique pair $(v_1,v_3)=(1,0)$, we need to restore it elsewhere, and precisely it can be done by

© 2012. The authors - Published by Atlantis Press

another change of assignment in sixth row from 2 to 0. This last induced change delete in turn the existing pair $(v_1,v_3)=(1,2)$, immediately restored by changing fifth row redundant assignment to 2, and also increased (v_2,v_3) coverage also by additional pairs (2,0) and (1,2). Similarity we continue to repair all the missing pair until no missing pair existed (see Fig. 1-(b)).

We expand all the parameters like that (see Fig. 1-(c) (d)). Details have already described by Calvagna and Garganitini.

However, 2-dimensional expansion has some issues to solve. We will state them in next part.



v_1	v_2	v_3
0	0	0
0	1	1
0	2	2
1	0	0
1	1	1
1	2	2
2	0	0
2	1	1
2	2	2

(a)

v_1	v_2	v_3
0	0	0
0	1	1
0	2	2
1	0	**1**
1	1	**2**
1	2	**0**
2	0	**2**
2	1	**0**
2	2	**1**

(b)

v_1	v_2	v_3	v_4
0	0	0	0
0	1	1	1
0	2	2	**x**
1	0	1	1
1	1	2	**x**
1	2	0	0
2	0	2	**x**
2	1	0	0
2	2	1	1

(c)

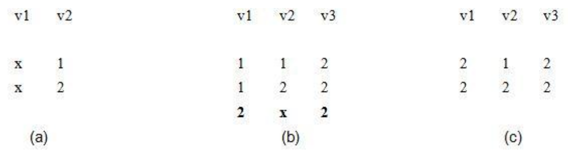
v_1	v_2	v_3	v_4
0	0	0	0
0	1	1	1
0	2	2	**0**
1	0	1	**0**
1	1	2	**1**
1	2	0	**1**
2	0	2	**1**
2	1	0	0
2	2	1	1

(d)

Figure 1. IPOs Example Task

B. *Issues with 2-dimensional expansion algorithm*

1) Expansion way: Expanding process mostly focuses on how to optimize horizonal expansion or repair process to eliminate the possibility of adding a row. Assume that there is a test case set S and ith parameter to cover pairwise, the total number of which is R_i. Now we expand the (i+1)th parameter, which means to search a combination Γ, that is, make R_i-V=(R_i,Γ) a best result for vertical expansion. If (i+1)th parameter range from 0 to (α-1), there are $|\alpha^{R_i}|$ possibilities to choose. In the research of the past, two ways are mainly refered to, one is IPO_H_EC, and other is IPO_H_IV[8]. Vertical expansion is low priority than horizontal. However, good vertical expansion can also make the next parameter expansion effective. Assume we extend (v1.v2)=(x,1) and (x,2), we assign arbitrary value to x(see Fig. 2-(a)). Now we expand the v3 parameter. Assume there is missing pair(2,2), the expansion result will better if x=2 than x=1.



v1	v2
x	1
x	2

(a)

v1	v2	v3
1	1	2
1	2	2
2	**x**	**2**

(b)

v1	v2	v3
2	1	2
2	2	2

(c)

Figure 2. Vertical Expansion Importance Example

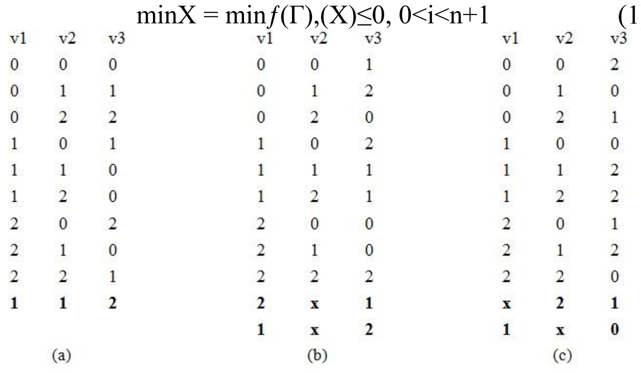
2) Random selection process: 2-dimensional expansion algorithm has uncertainty during the process. For instance, Assume there is a 3^3 system, after covering first two parameters, when we expand the 3rd parameter, there are 3 selections for us. It may be a good matter for solution but not for an algorithm because randomness is caused by that. The expansion result is on the basis of 0→1→2(see Fig. 3-(a)), 1→2→0(see Fig. 3-(b)), 2→0→1(see Fig. 3-(c)) by using IPO_H_IV. That explain why randomness interfere with generation result. In another instance, using IPO_H_EC, we are faces with the same issue that 12 best selections in 3^9=19683 selections when expand the 3rd parameter.

III. GLOBAL SUBSYSTEM

Definition 3.1: With input parameter set Γ, test case set X and number of parameters n, parameter set can be divided into $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$.

Definition 3.2(**Optimization Problem for Constraint**): So lution of the optimization solution set,

$$\min X = \min f(\Gamma),(X)\leq 0,\ 0<i<n+1 \qquad (1)$$



v1	v2	v3
0	0	0
0	1	1
0	2	2
1	0	1
1	1	0
1	2	0
2	0	2
2	1	0
2	2	1
1	**1**	**2**

(a)

v1	v2	v3
0	0	1
0	1	2
0	2	0
1	0	2
1	1	1
1	2	1
2	0	0
2	1	0
2	2	2
2	**x**	**1**
1	**x**	**2**

(b)

v1	v2	v3
0	0	2
0	1	0
0	2	1
1	0	0
1	1	2
1	2	2
2	0	1
2	1	2
2	2	0
x	**2**	**1**
1	**x**	**0**

(c)

Figure 3. Randomness Example Task

$f(\Gamma)$: Γ→X is objective function, $g_n(X)$ is constraint function which is meet pairwise condition. $g_n(X)$ is the final result set. We assume $g_i(X^*) \geq g_i(X)$ means X^* case set result is better than X, and X is best set if $g_i(X)=0$.

Feasible set is the set meet the constraint condition of X. All feasible set recorded as S is feasible region, which is as follows and $g_i(X) < 0$ means there are pairs redundant.

$$S= \{X|\ g_i(X)=0, i=[1,m]\ \};\ g_i(X)<0, i=[m,n]\} \qquad (2)$$

$I(X) = \{i|\ g_i(X)=0\}$ is binding constraint index set. If {m,n} is empty, then above problem has optimization solution; otherwise it is a Constraint optimization problem.

Definition 3.3(**Global Maximum Point**): Assume $X^* \in S$ and $\forall X \in S$, if

$$g_i(X^*) \geq g_i(X)$$

Then call X^* global maximum point.

Definition 3.4(**Local Maximum Value Point**): Assume $X^* \in S$ and $N_\varepsilon(X^*)=\{X|\ \|X-X^*\|<\varepsilon, \varepsilon=1,2,3,\ldots\}$, for $\forall X \in S \cap N_\varepsilon(X^*)$. If exist

$$g_i(X^*) \geq g_i(X)$$

Then call X^* local maximum value point.

It is important for optimization problem solution to ensure that algorithm has overall convergence. Therefore, there are some definitions as follows towards coverage issue.

Definition 3.5: If the feasible domain $g_i(X)=0$ and $g_j(X) = 0$, $0<j<i$, then $g_i(X)$ is the feasible stable point of the optimal solution.

Definition 3.6(**Lower Bound Theory**): Assume a system has n parameters, the number for parameter is Γ_i, i=1,2,…,n, there is a lower bound N of number of test case set as follows and $1 \leq i \neq j \leq n$:

$$N \geq \max(t_i \times t_j)$$

Proof: For the t_is value of i parameter and t_js value of j parameter, the completely combination number is $(t_i \times t_j)$. Therefore, test case set must have at least $(t_i \times t_j)$ cases in order to pairwise cover the parameters.

Finally, from above mentioned, we import lower bound theory to optimize the algorithm. That means during the 2-dimensional expansion, for each expansion, it is equivalent to expansion to its subsystem. Therefore, we define a global subsystem mechanism to optimize it. For instance, we want a generation result of 3^4 that contains a subsystem of 3^2 and 3^3. During the process, we have the generation of 3^2 and 3^3 at first. It is a feature of 2-dimensional expansion algorithm. We import global subsystem and use the lower bound theory to define it which will help us to eliminate the randomness.

IV. IPO_S_R

A. Algorithm

S is the set of cases generated. S[i][j] means value of ith col umn and jth row. P(a,b) means missing pairs of column a and b. N_i means total value of column i parameter. Parameter[i] means value of ith parameter. Let Established(m)=0 as row m which i s not considered as established row. Let Established(m)=1 as ro w m which is considered as established row. Besides, assume BestResult is subsystem best result case set of parameters unde r test.

Algorithm IPO_S_R(S,Parameter)

```
1.  P ← parameters
2.  Sort(P) as descend
3.  Empty(S)
4.  S ← Cover(1,2)
5.  for each column i from P(3) to P(n)
6.    if isBestResult(1,i) =1
7.      S ← continue and BestResult(1,i)
8.    else tempS ← AddColumn(i)
9.      if BetterResult(tempS, BestResult(1,i))>0
10.       BestResult(1,i) ← tempS
11.       S ← tempS
12.     end if
13.   end if
14. end for
```

AddColumn(i)

```
1.  for each row j in S do
2.    a ← S[i-1][j]
3.    if a ≤ N_i
4.      b:=a
5.    else b:=x
6.    end if
7.    S[i][j]:=b;
8.  end for
9.  while(P(i-1,i) is not empty)
10.   Randomly select (a,b) belonging to P(i-1,i)
11.   for each row j in S do
12.     if (a,b) = (S[i-1][j],S[i][j])
13.       Remove (a,b) from P(i-1,i)
14.       Established(j) ← 1
15.     else if (a,b) doesn't exist in column(i-1,i)
16.       Recover(a,i-1,b,i)
17.     end if
18.   end for
19. end while
```

Recover(a,i,b,j)

```
1.  Choose any row m where S[i][m]=x and S[j][m]=x or b
2.    S[i][m] :=a
3.  Choose any row m where S[i][m] = a and S[j][m] = x
    or where Established[m]=0 and S[j][m]=b
4.    Established[m]:=1
5.  if none
6.    insert a new row contained (a,b) and return;
7.  end if
8.  if S[j][m]=x S[j][m]:=b
9.  else if Established[m] = 0 and S[j][m] = b
10.   b_old:=S[j][m]
11.   S[j][m]:=b
12.   if(b_old=x) return
13.   else for each column h: 1..j-2 in S do
14.     if pair(S[h][m], b_old) is not covered
15.       Recover(S[h][m],h,b,j)
16.     end if
17.   end for
18.   end if
19. end if
```

B. Time and Space Complexity

In[10], the authors show that the number of tests for pairwise coverage grows at most logarithmically in n and quadratically in r with n the number of parameters and r the number of values. In Algorithm IPO_S_R, AddColumn function is $O(r^2)$, as it is dominated by the loop of calls to the Recover() function. The latter can at each execution either modify a row assignment or add a new test case to the test set. Recursion can be induced on when a row has been modified. This in turn can happen only N times overall, as any row can be modified at once. Moreover, since the recursive call is nested inside an O(n) loop, the total tine complexity of the function is O(nN), which is O(r2nlogn). Thus as the AddColumn() is called n times, the complexity is $O(r^4 n^2 \log n)$. The space complexity is $O(r^{n-1})$.

C. Analysis

We use global subsystem case set as auxiliary and expand x finally to ensure vertical expansion can give the next vertical expansion more selection. The whole generator structure is shown in Fig. 4.

V. Experiment

The proposed algorithm IPO_S_R aims at the issues of 2-dimensional expansion and tries to improve them. Now, we make IPOs algorithm as contrast and do some experiment. Here, a system contains n parameters and each parameter range for d is under test.

Firstly, we will compare the number if expansion. The times of Recover() function called will determine the efficiency of the algorithm.

Data in TABLE I and TABLE II show that our approach performance is better than IPOs. From that, it might be found that the modified algorithm can eliminate call times of Recover() function because it will assign x value in final not in every vertical expansion. In addition, this result doesn't import subsystem result.

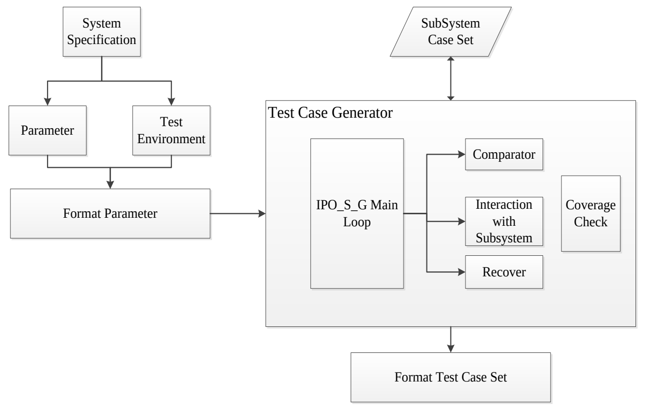


Figure 4. Test Case Generator

TABLE I. Comparative Recover times for 4^n System

n	5	10	20	30	40	50	60
IPOs	60	230	600	1057	1510	1931	2344
IPO_S_R	58	208	541	950	1321	1702	2021

TABLE II. Comparative Recover times for n3 System

n	4	5	6	7	8
IPOs	16	33	55	71	99
IPO_S_R	16	25	46	64	88

Secondly, we will see the advantages of the global subsystem. When the result imported, we make 500 times IPOs experiment and will show the average, minimum and variance just in TABLE III. With the complexity of the system under test increased, the randomness of the whole algorithm has become apparent, making test case generation outcome unstable. It is not conductive for test case generation. In addition, it makes generation instability.

TABLE III. Comparative set sizes

	5^{10}	10^{10}	15^{10}	20^{10}	25^{10}
Min	44	178	396	691	1090
μ	47.2	188.6	431.5	755.2	1168.5
σ^2	4.67	27.9	175.2	675.6	1584.3

Then we consider IPO_S_R algorithm to solve the problems with the experiment. We'll take a simple 5^{10} system as prototype. See Fig. 5 and to compare the performance between IPOs and IPO_S_R. It take about 30 times experiment of generating. IPO_S_R can avoid randomness and during generation it works stable in succeeding performance.

Another system under test is 10^{10} system, we will import subsystem of IPO 10^5 result in IPO_S_R and record the result as IPO_S_R_1, and import IPO 10^8 result as IPO_S_R_2. We do the experiment and result is shown in Fig. 6. Because IPO 10^{10} result is better than IPO_S_R, so if we import that subsystem of IPO result, the final result will be improved.

From the results it can be found that IPO_S_R algorithm has the following advantages:

1) High compatible with result of other case generation algorithm and easy to expand.

2) Eliminate the randomness of 2-dimension expansion and make uncertainty decrease.

3) With global subsystem results exist, it maintains case reusability and maintainability better.

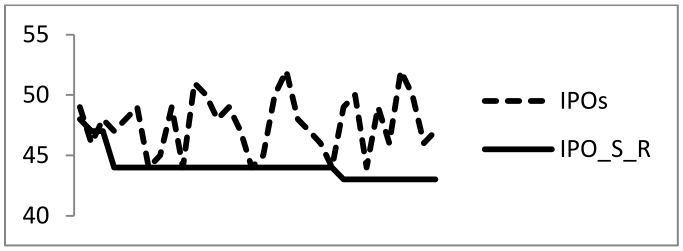


Figure 5. IPO_S_R Task Example Result 1

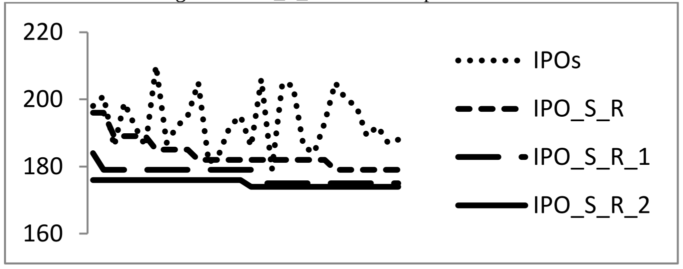


Figure 6. IPO_S_R Example Task Result 2

VI. Future Work and Conclusion

In this paper, we studied 2-dimensional expansion analysis in pairwise test case generation. We then propose a strategy to eliminate randomness and optimize some components. In addition, we do some experiment to prove our algorithm feasible and improve some places. As a final remark, work is undergoing to extend the algorithm to support t-wise testing and constrains over the inputs. In particular, we are working on applying the use of global subsystem to some other algorithm besides 2-dimensional expansion.

References

[1] D.L.Parnas, "On the Design and Development of Program Families" IEEE Trans. Software Eng. Vol.2, No.1, pp.1-9, 1976.

[2] G.J. Myers. The art of Software testing. John Wiley and Sons, 1979.

[3] B.Beizer. Software testing Techniques. Van Nostrand Reinhold, 1990.

[4] Tatsumi,K. "Test-Case-Design Support System." In Proceedings of the International Conference on Quality Control(ICQC), Tokyo, 1987, pages 615-620, 1987.

[5] Czerwonka J. Pairwise testing in real world: Practical extensions to test generators. In: Butt D. Gens C.eds. Proc. of the 24th Pacific Northwest Software Quality Conf. 2006.

[6] Bach, J., and P.Shroeder. "Pairwise testing: A Best Practic that Isn't." In Proceedings of the 22nd Pacific Northwest Software Quality Conference, pages 180-196, 2004.

[7] Czerwonka J. Pairwise Testing. 2009. http://www.pairwise.org/

[8] Lei Y, Tai KC. In-Parameter-Order: A test generation strategy for pairwise testing. In: Tsai J, Keefe T, Stewart D, eds. Proc. Of the IEEE Int'l. Symp. On high Assurance Systems Engineering. Los Alamitos: IEEE Press, 1998. 254-261.

[9] A.Calvaga, and A.Gargantini, "IPO-s: Incremental Generation of Combinatorial Interaction Test Data Based on Symmetrics of Covering Arrays," in Proc. of IEEE Int. Conf. on Software Testing Vertification and Validation Workshops, Denver, Colorado, USA, 2009, pp. 10-18.

[10] D.M.Cohen, S.R.Daral, M.L.Fredman, and G.C.Patton. The AETG system: An approach to testing based on combinatorial design. IEEE Transactions On Software Engineering, 23(7):437-444,1997.