TELKOMNIKA, Vol.11, No.1, January 2013, pp. 136~143
ISSN: 2302-4046

A Prioritized Test Generation Method for Pair-wise Testing

Yu Wang*, Hao Wu, Zhenyu Sheng
Computer and Information Engineering College of HOHAI University
Nanjing, China
*corresponding author, e-mail: won9805@hhu.edu.cn

Abstract

The most sufficient test methods are based on a test case set which covers all combinations of parameters. However, the scale of test cases is always too large and their cost cannot be accepted. People will first consider the implementation of critical test cases. Even if the test is terminated suddenly, the test cases of high importance will have been executed. It improves the testing efficiency while securing the detection rate of defect. The contribution of this paper is how to generate pair-wise testing cases with a priority. Firstly, We design formulas to compute the weights of priorities. Secondly, we adopt a greed algorithm to solve the combined testing problems. Furthermore, we integrate the greed strategy into a genetic algorithm which makes the most efficient testing in critical parameters and their sets, and ensures its detection rate of defect under limited resources.

Keywords*: Component; Software Test; Combinatorial Test; Genetic Algorithm.*

1. Introduction

Nowadays, testing a software tends to need more input parameters. And each parameter may have a number of different values or equivalence partitioning. The most sufficient test methods are based on a test case set which covers all combinations of parameters. However, the scale of produced test case is always too large and their cost cannot be accepted. In the observation of many applications, many procedural errors are caused by the interaction of a few parameters [1]. Therefore, researchers have proposed combinatorial test method.

Abroad researches mainly include three fields: combination test based on algebraic methods, e.g. Orthogonal Latin Squares and TCONFIG [2,3]; combination of test methods based on heuristic, e.g. Simulate Anneal Arithmetic methods, TABU Search, ant colony optimization[4,5]; combination of test methods based on greedy methods, e.g. AETG, IPO[6,7].

This article studies the classic combination of test methods: Orthogonal Test and AETG and then proposes two algorithms which make the most efficient testing in critical parameters and their combinations, and ensures its detection rate of defects under limited resources.

2. Classic Combinatorial Test Methods and Analysis

Combinatorial test is a common method among software test. Many experiments indicates that pair-wise testing of parameters for various systems is practical and effective. And the failure of many software systems is caused by the test parameters and their interactions. The common coverage standards of combinatorial test methods are as followed: Each-used Coverage, Pair-wise Coverage, T-Way, Variable Strength Coverage. The common combinatorial models are Covering Array Model and Variable Strength Covering Array Model. Combinatorial test generates a coverage matrix which meets a particular combination of coverage criteria. Theoretical studies have proved that the combination of test suite generation is an NP-the Complete Problem. Due to the minimum set of test cases can't be generated in polynomial time, approximation algorithm is generally preferred.

When Mandl implements the test of Ada Compiler, he uses orthogonal Latin squares to construct a test case set which meets the Pair-wise standard. As for a tested system with n

Received September 29, 2012; Revised November 23, 2012; Accepted November 23, 2012

parameters, it only needs n-2 mutually orthogonal Latin squares to construct the set of test cases under the Pair-wise criteria. The Orthogonal Latin squares based on the following assumptions:

- All parameters should have the same number of values;
- There are a sufficient number of orthogonal Latin squares;
- Each parameter is independent.

Only under these assumptions, can we use this method to generate the best collection of test cases. However, in practice, these constraint conditions can't be met quite often, which makes the orthogonal Latin method have a certain limitation.

D. M. Cohen from Bell Labs proposes the AETG. The method is integrated in a commercial software named AETG (Automatic Efficient Test Generator). The algorithm is as follows:

```
Algorithmic 1 Automatic Efficient Test Generator
Input:
test cases are selected, ti , ti-1;
test combinations have not been yet covered, Uncover.
Output:
pick up the test cases target which covers the most combination Uncover from M candidate.
while (Uncover ≠ ∅ ){
 for ( i = 0; i < M; i ++ ){ // to generator M candidate
        Identify the most frequently parameter from Uncover set, and set this
        parameter to the first argument;
        Make a random arrangement of the remaining parameters, determine the
        parameter values in turn accord to the Greedy Algorithm and generate
        candidate test cases;
 }
}
```

We find the traditional combination of test methods e.g. Orthogonal Test and AETG, are committed to generate the optimum combination of test case collection. However, in a practical application, due to the limitation of time and cost, people cannot run the entire test suite. In order to improve the efficiency of testing, the test cases can be sorted in a descending order of priority so that the test cases of high priority can be implemented as soon as possible. At present, WDA (Weighted Density Algorithm) has made an attempt.

3. Pair-Wise test methods about Priority

3.1. Priority & Combination model

We assign the value 0-1 to each parameter as priority weights. The weight represents its priority information, the greater the weight that the higher the priority. The standards of how to set weights are based on the code coverage, the cost of test cases, recently modified code domain and the testers' inclination. After the assignment of each parameter, the weight of any binary tuple is computed from the product of two weights inside the binary tuple; The weight of a test case is the sum of a binary group weights when it first covers.

Priority combination model (and also called Biased Covering Array), is defined as followed. A Biased Covering Array which meets the Pair-Wise criteria is a CA(m; 2, P), and it also satisfy the below conditions:

- The test cases are sorted in a descending order based on their priority weights;
- The weight sum of the first N test cases should be the maximum as far as possible. That means even if we can't find another CA(m; 2, P), the sum of its first N test cases should be bigger;
- The goal of our algorithm is to generate a Biased Covering Array and make it meet pair-wise criteria.

3.2. Computation of Priority Value

This article takes a decimal value from 0 to 1 as the priority weight, which measures the significance of a test case. Because of the huge number of test cases, artificial assignment of weights will inevitably make the metric inaccurate and small discrimination. But the influencing factors about priority of test cases are definite, and the influence factors can be evaluated easily. We compute this priority weight with specific formula. The formula has four

factors, which are code coverage, cost of test cases, recently modified code domain and the testers' inclination. And Eq.1 is presented below:

$$w = C \times \eta_1 + \frac{p_{\max} - p}{p_{\max} - p_{\min}} \times \eta_2 + q \times \eta_3 + f \times \eta_4 \quad (1)$$

In this formula, η_1, η_2, η_3, η_4 represent the weight of each influence factors. And $\eta_1 + \eta_2 + \eta_3 + \eta_4 = 1$. w is weight of a test case. C means the code coverage rate, a decimal value from 0 to 1; p is the cost of a test case; p_{max} is the maximum cost of test case. p_{min} is the minimum cost of a test case; f means the inclination degree of a test, a decimal value from 0 to 1. q is a degree of last modified code domain ,and it also meets Eq.2:

$$q = e^{-\frac{t}{\alpha}} \quad (2)$$

t is a time value since the last modified; α is a constant value. This formula meets the Hermann Ebbinghaus curve. As for all the test cases, testers should assign value to each parameters according to the above four criterions, and then compute each weight of test cases with Eq.1. This weight is the final priority value of a specific test.

3.3. Greedy Algorithm for Test Set

Inspired by the AETG Algorithm, this article takes one-test-at-a-time one-dimensional expansion strategy. According to the Greed Algorithm, every time select one test case complies with the greed strategy and put it into test case collection. When the algorithm is initializing, the valued parameter sets are inputted. Then, calculate binary groups and their weights with all parameters. And set them into the uncovered binary group *Uncover*. Each time, select a test case with possibly large weight and add it to the test collection *TestSuite*, until the *Uncover* set becomes empty. The framework of the algorithm is as follows:

Algorithm 2 Greedy Algorithm for Test Set
Input : Factor Set *F*.
Output : TestSuite
Put all the pairwise interaction with weights of *F* into uncovered interaction set *Uncover*.
while (*Uncover* ≠ ∅){
  Initialize a new test TestCandidate with all factors not fixed and an empty test BestTest;
  for (*i* = 0; *i* < *M*; *i* ++){
    Select interaction *t* with maximum weights in *Uncover*;
    Fix factors of *TestCandidate* according to *t*, other factors are free;
    while (free factors remain){
      Randomly select a factor *f*;
      Select a value *v* of f that is maximal the total weights with fixed factors;
    }
    if (*i* == 0) *BestTest* = *TestCandidate*;
    else if (weights of *TestCandidate* > weights of *BestTest*)
      BestTest = *TestCandidate*;
  }
  Add *BestTest* to *TestSuite*;
  Remove the interactions covered by *BestTest* from *Uncover*;
}
return *TestSuite*;

The priority combination of test case generation is a typical combinatorial optimization problem, which can be solved by parameterized and heuristic search algorithm. Greedy strategy has a higher ability of local search, but it is easy to fall into local optimal solution. And Genetic algorithm has strong global search capability. Combine the greedy strategy and genetic

algorithm can solve the problem of combinatorial test with priority. And it can develop a better global search algorithm.

3.4. Greedy Strategy Combined with Genetic Algorithm for Test Set

The proposed algorithm uses one-test-at-a-time and one-dimensional expansion strategy. Every time, we use this algorithm to get a test case, and then add it to the test case collection *TestSuite*. Our algorithm can be divided into two parts: Greedy Strategy and Genetic Algorithm. The first part generates *M* candidates as the input of second part. In Genetic Algorithm, the input will be encoded to get a initialized population. It is evaluated through a certain number of generations. Finally, when the genetic algorithm stops, the best individual will be selected to add in a test collection *TestSuite*.

The section of Greedy Algorithm constructs *M* test cases, the procedures are as follows:

- Selected the big ith binary group from *Uncover* set, as t_i, i=1,2,..., M;
- Identify the two parameters' value of the candidate test case testi based on t_i, assign values to the rest unfixed parameters in the order of Greed Strategy. The Greed Strategy here means pick up a parameter and ensure it can combine with the fixed parameters to generate a binary group, whose multiplex weight should be the largest, finally we get $test_i$;
- The basic operations of Genetic Algorithm are as follows:

1) Encoding Method: It uses binary encoding method, the number of possible value of *f* may be *t*, if $2^{n-1} < t \leq 2^n$, the encoding length of this parameter will be *n*. Fig.1 shows that assuming the system has 4 parameters: (f_1, f_2, f_3, f_4), their value is 2,2,3,4 respectively. They can be encoded with six bits: f_1 occupies b_0, f_2 occupies b_1, f_3 occupies b_2 and b_3, f_4 occupies b_4 and b_5. The first three codes of f_3 are 00, 01, 10, and they can be represented by $v_{3,0}$, $v_{3,1}$, $v_{3,2}$. The maximum weight of them is selected as the encoding number "11".

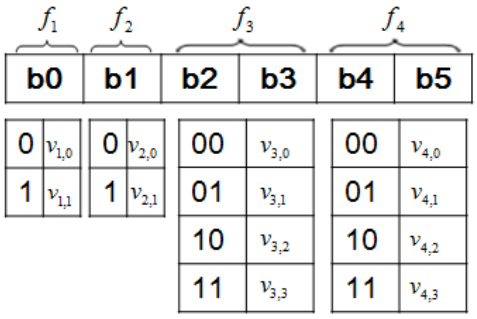

Figure 1. the effect of genetic encode

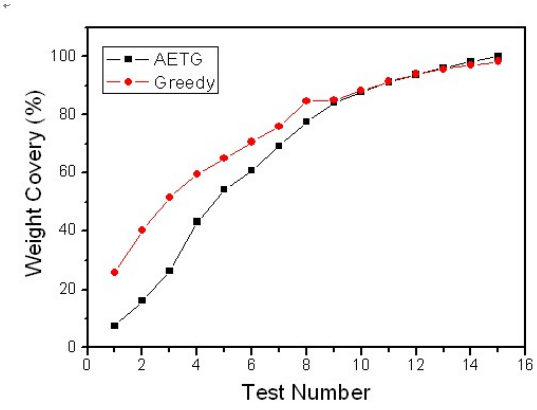

Figure 2. the effect of Comparison between AETG and this Greedy

2) Fitness Function: The Genetic Algorithm assesses the level of the pros and the cons of each individual from individual fitness so that the algorithm can decide the genetic opportunities of each individual. Under the criterion of pair-wise coverage test, the composite weights of combinatorial test cases set generated by Greedy Strategy combined with genetic algorithm should be as large as possible. Therefore, in the evolution of each generation, test cases with largest weight will be picked up. Because of the composite weights of test cases always taken non-negative, their composite weights can be choosen directly as the fitness.

3) Selection Operation: The selection operation is a strategy of combination of proportional selection and optimal conservation strategy.

4) Cross Operation: Given the crossover probability *Pc*, takes single-point crossover strategy.

5) Mutation Operation: Given the mutation probability *Pm*, negate the variance binary bit.

4. Experimental result & Analysis

4.1. Application model and case introduction

In the large water conservancy and hydropower engineering, safety and emergency management information systems is aimed at real-time monitoring of major hazard, timely response, scientific decision and dynamic management. They can also help improve the institutionalization, standardization, informatization of safety production. Three Gorges company develops the MIS to enhance the monitoring of safety production, prevent and reduce the safety accidents, guarantee the safety of employees and construction. The safety and emergency management information system is also a support for decision, management and execution of Three Gorges enterprise.

This system is developed by the engineering institution of Huazhong University of Sciences and Technology and tested by water resources utilization and engineering safety national engineering research center of HoHai University.

The obstacles of the safety and emergency management information systems are that we can't find proper test suites to execute the combination test of system service. Instead, what we can utilize are the known defects and faults, the hidden software defects are difficult to be found. Usually, the use case needs too much input information as well as the output results. In addition, the realize ways of test software are various which there is no object specification to constraint. So combination test suites fail to implement complete test of the system to discover the hidden defect.

The scale of the tested system: the size of source files including modules, scripts, database check records is 200.5MB, the amount of all files is 40260, and the file folders is 4500, quantity of code is 43750000.

About traversing the dependency sequence of use cases generated by activity diagram, the method proposed in references is representative. Firstly, process the activity diagram in a " fragment" way to generate the Formal Activity Diagram(FAD). Secondly, part the FAD into fragments and generate a sequence covering every control flow for each fragment. Lastly, merge all fragments according to the fragment equation (the Logical relationship among fragments) to generate the satisfactory dependency sequences set of use cases. However, this method has evident defects:

- Specific partition method is not provided;
- Too much work is required to generate FAD, and it is also complex;
- The activities caused by invalid input are not be considered, so it can't generate complete dependency sequences, and additional use cases are required.

Different from the known method, this paper introduces a method which build weighed control flow tree(CFT) which every tree node has priority weight. First, we mark the tree nodes while building the CFT. Then we traverse the CFT to generate the dependency sequences of use cases.

Dfinition: Control Flow Tree(CFT) is binary group (V,E), V is the vertex set and E is the directed edge set. The concrete method of building the CFT is as follows:

- The start state of the activity diagram is the root node which priority weight is highest. Given that a CFT has only one root node;
- Leaf nodes are marked with FN,ER. Other nodes are as non-leaf nodes;
- Each control flow of activity diagram is as a directed edge between two nodes accordingly.

For marking the control flow with number is to confirm the child nodes of corresponding node and the type of child node also indicates the state of parent node after execution process. For example, nodes ER indicate the abnormal results after execution process, nodes FN and other non-leaf nodes indicate the normal state of parent nodes. So the CFT we define satisfies the model introduced in section 3.1.

When setting the priority weight of root node, we can accord to four factors including code coverage, cost of test cases, recently modified code domain and the testers' inclination. Then, assign the priority weight of relevant data object in turn along the execution path. At the same time, concrete realization form(amount of parameters, type of parameters and their domains) and relevant activities should be settled down.

Example: Fig.2.1 and Fig.2.2 show the corresponding WCFTs of the created activity diagram. Fig.2.1 is the CFT without loops and Fig.2.2 is the CFT with one loop. Every leaf node is "FN" or "ER", indicates the normal end and abnormal end respectively. In the aspect of data, it indicates the valid I/O data set and invalid I/O data set respectively. In the design stage, this category and partition is just based on logical analysis, and all data doesn't involve specific platform information. Table 2 shows the comparsion table of the description of platform-indenpendent and platform-dependent data. The specific case of parameter is showed in Table 5 in the appendix section.

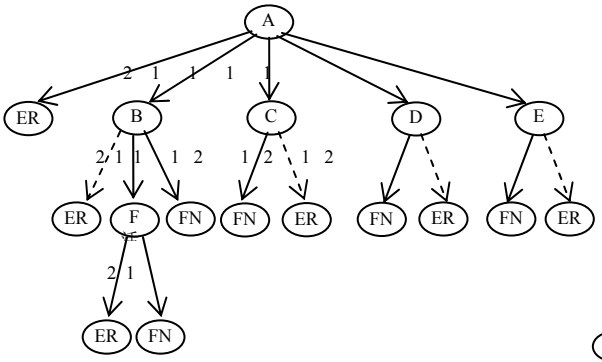

Figure 3. CFT of the MIS(without loops)

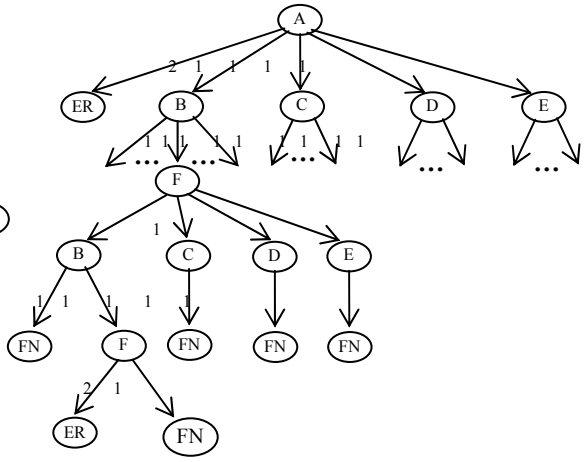

Figure 4. CFT of the MIS(with one loop)

Table 2. Comparsion table of the description of platform-indenpendent and platform-dependent data

Object descripttion	Platform-independent data		Platform-dependent data			Related activities
	design parameters	State category	actual parameters	domain	valid state data	

However, if we only part the execution state of use case by traversing and sampling combination, it will possibly lead to invalid control flow paths appearing in the CFT. Therefore, we need to exclude these invalid flow paths and then execute the combination test. The combination test with priority can decrease the scale of CFT and increase the traverse efficiency.

4.2. Comparison between AETG and our Greedy Strategy

There are 13 parameters and the amount of each parameter's value is 3, we assign the weight with one decimal value from 0 to 1 randomly. As illustrated from the result Fig.2, comparing with the proportion of the weights of previous test case accounting for the total weights, our method covers more priority values much earlier.

4.3. Comparison among WDA, Greedy, Greedy + GA

Table 3 and Table 4 are the test results of production and safety management information system of water conservancy and hydropower project, and emergency management information system for water resources and hydropower engineering, respectively. The first column is the number of test cases, the rest three columns are the sum of weights of test case generated by three methods. As the tables show, greedy strategy combined with genetic algorithm makes the test cases cover more weight as soon as possible.

Table 3. results of production and safety MIS of wate conservancy and hydropower project

Test NO	WDA	Greedy	Greedy + GA
5	52.197	52.284	54.471
10	87.450	87.239	88.025
20	142.588	142.872	145.259
30	187.975	188.223	192.473
50	242.734	242.767	248.127
60	254.542	253.609	259.312
70	262.896	261.744	267.459

Table 4. results of emergency MIS for water resources and hydropower project

Test NO	WDA	Greedy	Greedy + GA
5	80.992	81.052	82.343
10	139.421	140.647	143.249
15	183.324	184.958	189.185
20	207.758	207.571	213.235
25	226.032	225.736	232.558
30	239.592	238.473	245.943
35	251.048	250.254	256.524
40	259.983	259.736	264.939

5. Conclusion

This paper described the research status of the combinatorial test, explored the classic combination of test methods: Orthogonal Latin squares algorithm, AETG methods in detail. We introduce a priority weight to indicate the priority of test case, and use the weighted parameters to construct a composite binary group. We add all the value of binary groups covered by test cases to get the weights of test cases. In addition, we adopt Biased Covering Array to construct a model of pair-wise test cases collection with priority. And enlightened by Greedy Strategy and the thought of AETG, we have designed an approximation algorithm to generate Biased Covering Array. This paper in previous sections has given the description and the framework of our algorithm; In order to further improve the weights of Biased Covering Array, we introduced the Genetic Algorithm and design another method to get the array. At last, we verify the correctness and validity of these two methods through 2 typical combinatorial test instances. In conclusion, what we have done has laid a foundation for further better algorithms.

However, this paper is just an attempt which also has defects.Greedy algorithm can generate the Biased Covering Array quickly, but it is a little far from the optimal result.Greedy algorithm integrated with GA help to get a better result, but the efficiency is remain to be improved. Therefore, we will put emphasis on how to solve the combination test problem involving priority in a better and quicker way.

Acknowledgement

The work is supported by the NSF No. 61103017.and Su Jingxin comprehensive (No[2011]1178).

References

[1] Kuhn DR, Reilly MJ. An investigation of the applicability of design of experiments to software testing. *In: Caulfield M, ed. Proc.of the Annual NASA/IEEE (SEW)*. Los Alamitos: IEEE Press, 2002.
[2] Jun YAN, Jian ZHANG. *Journal of Software*. 2009; 20(6): 1393-1405.(In Chinese)
[3] Jianfeng LIAO, Xiantao CAI. *Computer Engineering and Applications*, 2012; 48(11):65-70. (In Chinese)
[4] Nurmela KJ, Upper bounds for covering arrays by tabu search. *Discrete Applied Mathematics*. 2004; 138(9):143-152.
[5] Xiang CHEN, Qing GU, Xinping WANG, Daoxu CHEN. *Computer Science*. 2010; 37(3): 1-5. (In Chinese)
[6] Mats Grindal, Birgitta Lindstrom, Jeff Offutt, Sten Fandler. An evaluation of combination strategies for test case selection. *Empirical Software Engineering*. 2006; 583-611.
[7] Lei Y, Kacker R, Kuhn DR, Okun V, Lawrence J. IPOG: A general strategy for t-way software testing. In: Leaney J, O'Neill T, Peng J, eds. *Proc. of the Annual IEEE Int'l Conf. and Workshops on the Engineering of Computer-Based Systems (ECBS)*. LosAlamitos: IEEE Press, 2007; 549-556.

Appendix

The description of related data object for super user in water adjustment system is as Table 5.

Table 5 comparsion table of a use case in management system

Object description	Platform-independent data		Platform-dependent data			Related activities
	design parameters	State category	actual parameters	domain	valid state data	
User login interface	page	-	JSP page	-	index.jsp	A
User account	user name, password, type	valid/invalid	(text，text，text)	user name:[0-9]&[a-z]&[A-Z]，6-15 character password：[0-9]&[a-z]&[A-Z]，6-15 characters Type：{“General” | ”Admin” }	valid：（ ABcd2012,123456,Admin ） invalid：{（#123as$, 12345,Admin ）（ ABcd2012,@1234%,Admin ）（ ABcd2012,123456,General ）}	A
Login Action	button	click	submit button	-	click login button	A
System Management Interface	page (the item of system management function)	-	JSP page	-	system management main page	B/C/D/E/FN
“Water consumption summary” Action	function item	click	*hyperlink*	-	click on *water consumption summary* button	C
Water summary of each unit output interface (Y/M)	page (unit name, the water consumption, major period)	-	JSP page	-	water summary of each unit output interface(Y/M)	C
“Water inquiry” Action	function item	click	hyperlink	-	click on *water inquiry* button	D
Water query of each unit output inerface	page（report、approvel,actual water consumption）	-	JSP page	-	each unit’s water (Y/M) information display page	D
“Time inquiry ” Action	function item	click	hyperlink	-	click on *time inquiry* button	E
Water information in specific time	page (unit name, specific time, water consumption)	-	JSP page	-	display page of water information in specific	E
“Plan and approval”Action	function item	click	hyperlink	-	click on *plan and approval* button	B
New plan display	page (unit name, year’s and plan, the examination and approval link labeled)	-	JSP page	-	new plan of water consumption display page	B
“Approval” Action	function item	click	hyperlink	-	click approval button	B
Input approval data	page	-	JSP page（FORM）	-	approval data to input page	F
Approval of Water consumption	number (approved water consumption)	valid/invalid	text	water consumption volume：[0-9]，0-3 characters	valid：50 Invalid：1000,-1	F
“Invalid data” prompt	instant prompt	-	real-time prompt	-	“invalid data”real-time prompt	F
Submit Action	submit button	click	submit button	-	click submit button	F
Loging out	function item	click	hyperlink	-	click log out button	FN