International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 2, Issue 5, May 2015, PP 38-44
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

A Heuristic Technique for Generating Pairwise Test Cases for Web Applications

M.LakshmiPrasad[1], M.Seshadri[2], A.SumaTejaswini[3], G.RupeshKumar[4]

[1]NBKR Institute of Science & Technology
[1]Department of Computer Science & Engineering
[1]Nellore, AP, India
[2, 3, 4] NBKR Institute of Science & Technology
[2, 3, 4] Department of Computer Science & Engineering
[2, 3, 4] Nellore, AP, India
[1]prasad.hinduniv@gmail.com, [2]funwithsheshu@gmail.com,[3]teju.asam@gmail.com,
[4]rupeshkumar.gudi@gmail.com

Abstract: *Software testing/Programming testing is the procedure of breaking down a product thing to distinguish the contrasts in the middle of existing and obliged conditions (that is, bugs) and to assess the highlights of the product things. By and large, efficient testing is the best way to survey the event of disappointments in frameworks comprising in an arrangement of segments, however given the measure of genuine frameworks, it would be exceptionally lavish to develop and test all the conceivable mixes of the conditions of parts. Combinatorial cooperation testing is a current method that properly diminishes the quantity of experiments by picking sets, triplets, and so forth., i.e. t-tuples, of data qualities. Obviously, the viability of a test suite is higher when picking e.g. triplets of inputs as opposed to combines. Since high estimations of t are ideal, a substantial number of experiments could in any case be created. This methodology is roused by the perception that in numerous applications a noteworthy number of flaws are brought on by cooperations of a littler number of parameters. We propose a strategy called HTPTG-Web for building the littlest conceivable test suite of size t. This method comprises in decreasing the quantity of experiments via painstakingly picking non repetitive t-tuples. This venture demonstrates that for acquiring the littlest conceivable arrangement of tests, it is best to produce a vast "adaptable" arrangement of t-tuples and afterward diminish such a set until the littlest one is acquired. The handiness of these calculations is shown through investigations. For the situation t = 3 specifically, our calculations displayed noteworthy results.*

Keywords:*Pairwise testing,AETG,Ant Colony Optimization,Simulated Bee Colony*

1. INTRODUCTION

Verification and validation of highly-configurable software systems, such as those supporting many optional or customizable features, is a challenging activity. In fact, due to its intrinsic complexity, formal specification of the whole system often requires a great effort. Hence, modelling activities may become extremely expensive and time consuming, and the tester usually decides to model (at least initially) only the inputs and require they are sufficiently covered by tests. On the other hand, traditional approaches to systematic testing based on structural coverage criteria are unsuitable to detect incorrect behaviours or failures caused by unintended interaction between optional features. Since most of the faults in a software system are triggered by unintended interaction of a relatively low number of input parameters, testing all the combinations of input configurations can be very effective in revealing software defects.

Combinatorial Interaction Testing (CIT) systematically explores pairwise feature interactions inside a given system, by effectively combining all pair-tuples of parameter assignments in the smallest possible number of test cases. In particular, *pairwise* testing ($t = 2$), consists in testing all *pairs* of input values at least once. It has been empirically confirmed that pairwise testing can detect a significantly large part (typically 50% to 75%) of faults in a system], thus many CIT approaches have been developed and are currently applied in practice. CIT can be applied to a wide variety of problems: highly-configurable software systems, software product lines, hardware systems, etc.

2. LITERATURE SURVEY

S.A Ghazi and M.A Ahmed[1] had proposed the pair-wise test coverage using genetic algorithms.There has been an emerging trend to develop software using different components. In this way the cost of the software reduces and the developer is able to complete the system efficiently. The components' code may or may not be visible to the developer. Testing, in this case, requires the development of a set of test configurations that can be applied on the software. However, for software that comprises a large number of components, it is infeasible to test each and every test configuration within the limited testing budget and time. A GA-based technique was proposed that identifies a set of test Configurations that are expected to maximize pairwise coverage, with the constraint that the number of test configurations is predefined. Although the paper primarily focuses on the interaction between software components, the idea can be applied to single code component testing. They had performed some experiments using their proposed approach.

J.D. McCaffrey [2] had presented a new technique for generating the pairwise test sets using a genetic algorithm.Pairwise testing is a combinatorial technique used to reduce the number of test case inputs to a system in situations where exhaustive testing with all possible inputs is not possible or prohibitively expensive. Given a set of input parameters where each parameter can take on one of a discrete set of values, a pairwise test set consists of a collection of vectors which captures all possible combinations of pair of parameter values. The generation of minimal pair-wise test sets has been shown to be an NP-complete problem and there have been several deterministic algorithms published. Compared with published results for deterministic pairwise test set generation algorithms, the genetic algorithm approach produced test sets which were comparable or better in terms of test set size in 39 out of 40 cases. However, the genetic algorithm approach required longer processing time than deterministic approaches in all cases. The results demonstrate that the generation of pairwise test sets using a genetic algorithm is possible, and suggest that the approach may be practical and useful in certain testing scenarios.

J.D., McCaffrey [3] had proposed an empirical study of pairwise test set generation using a genetic algorithm.Pairwise test set generation is the process of producing a subset of all possible test case inputs to a system in situations where exhaustive testing is not possible or is prohibitively expensive. For a given system under test with a set of input parameters where each parameter can take on one of a discrete set of values, a pairwise test set consists of a collection of vectors which capture all possible combinations of pair of parameter values. Generating pairwise test sets with a minimal size has been shown to be an NP-complete problem, and several deterministic generation algorithms have been published. The genetic algorithm approach produced pair-wise test sets with comparable or smaller (better) size compared with published results for deterministic algorithms for 39 out of 40 benchmark problems. However, the genetic algorithm test set generation technique required significantly longer processing time in all cases. The results illustrate that generation of pair-wise test sets using a genetic algorithm is possible, and suggest that the technique may be both practical and useful in certain software testing situations.

S Maity and A Nayak [4] had proposed an improved test generation algorithms for pair-wise testing. Software testing is expensive and time consuming. Given the different input parameters with multiple possible values for each parameter, performing exhaustive testing which tests all possible combinations is practically impossible. Generating an optimal test set which will effectively test the software system is therefore desired. Pair-wise testing is known for its effectiveness in different types of software testing. Pair-wise testing requires that for a given numbers of input parameters to the system, each possible combination of values for any pair of parameters be covered by at least one test case Pair-wise testing is known for its effectiveness in different types of software testing. The problem of generating a minimum size test set for pair-wise testing is NP-complete. They also presented new techniques for reducing the number of test cases for pair-wise testing. An algorithm to generate test cases for 2-valued parameters was showed and how orthogonal arrays and ordered designs may be used for deriving test cases for parameters with more than two values. Moreover, using mixed-level or asymmetric orthogonal array as tool and also presented a test set generation strategy for parameters with different number of values. A comparison of empirical results with previous test generation strategies "AETG" and "IPO" shows that the number of test cases generated with the proposed methodology is never higher and in some cases significantly lower than using AETG or IPO.

Kewen Li and Zhixia Yang [5] had proposed a new generating method for pair-wise covering test data based on ACO. Optimizing test suite can reduce the cost of time and resources, and improve the efficiency of regression test when test cases are generated. The generation of pair-wise covering test data is an NP question, which can be solved by heuristic method, greedy arithmetic and algebra method at present. Ant colony arithmetic was adopted, which is a new way to solve the pair-wise test data generating question. It can generate fewer test cases which can cover more pair combinations, and can solve questions with fast calculate speed. The method can get the goal of optimizing in the process of regression test. The result shows that the method is feasible.

A Calvagna, A Gargantini, and E Tramontana, [6] had built a t-wise combinatorial interaction test suites by means of grid computing. Generally, systematic testing is the only way to assess the occurrence of failures in systems consisting in a set of components, however given the size of real-world systems, it would be very expensive to construct and test all the possible combinations of the states of components. Combinatorial interaction testing is an existing technique that appropriately reduces the number of test cases by choosing pairs, triplets, etc., i.e. t-tuples, of input values. Of course, the effectiveness of a test suite is higher when choosing e.g. triplets of inputs rather than pairs. Since high values of t are preferable, a large number of test cases could still be generated. A technique for building the smallest possible test suite of size pair was proposed. This technique consists in reducing the number of test cases by carefully choosing non redundant pair-tuples. They have shown that for obtaining the smallest possible set of tests, it is best to generate a large 'flexible' set of t-tuples and then reduce such a set until the smallest one is obtained. Reduction is a computationally expensive operation and therefore it is worth performing it by parallelising its execution. This paper proposes a solution for executing the reduction algorithm over a set of grid resources.

Xiaoying Pan, and Hao Chen, [7] had used organizational evolutionary particle swarm techniques to generate test cases for combinatorial testing. Based on the analysis of the characteristics of combinatorial testing, an organizational evolutionary particle swarm algorithm (OEPST) to generate test cases for combinatorial testing is proposed. This algorithm is used to select the test cases of local optimal coverage in current environment based on these test cases, and then a test suite satisfying the pair-wise coverage criterion is built. The empirical results show that their approach can effectively reduce the number of test case.

Wang Jianfeng, and Jiang Shouda[8] had improved algorithm for test data generation based on particle swarm optimization. The test case generation is one of key issues of combinatorial testing. In this paper, a new algorithm for test data generation based on Particle Swarm Optimization PSO is presented. Based on Particle Swarm Optimization, the optimization base and extended parameters are introduced. The number of the current output test data is adjusted dynamically according to the data generated before. The efficiency of the test data generation is improved effectively on the premise of ensuring the optimization of the data generated

Xiang Chen, Qing Gu, Jingxian Qi, and Daoxu Chen [9] had applied particle swarm optimization to pair wise testing. Combinatorial testing (also called interaction testing) is an effective specification-based test input generation technique. By now most of research works in combinatorial testing aims to propose novel approaches trying to generate test suites with minimum size that still cover all the pairwise, triple, or n-way combinations of factors. Since the difficulty of solving this problem is demonstrated to be NP-hard, existing approaches have been designed to generate optimal or near optimal combinatorial test suites in polynomial time. In this paper, we try to apply particle swarm optimization (PSO), a kind of meta-heuristic search technique, to pairwise testing (i.e. a special case of combinatorial testing aiming to cover all the pairwise combinations). To systematically build pairwise test suites, they had proposed two different PSO based algorithms. One algorithm is based on one-test-at-a-time strategy and the other is based on IPO-like strategy. In these two different algorithms, they had use PSO to complete the construction of a single test. To successfully apply PSO to cover more uncovered pairwise combinations in this construction process, we provide a detailed description on how to formulate the search space, define the fitness function and set some heuristic settings. To verify the effectiveness of their approach, their algorithms were implemented and choose some typical inputs. In our empirical study, the impact factors of their approach were analyzed and compared their approach to other well-known approaches. Final empirical results show the effectiveness and efficiency of our approach

3. EXISTING SYSTEM

Combination strategies are a class of test-case selection methods where test cases are identified by choosing "interesting" values1, and then combining those values of test object parameters. The values are selected based on some combinatorial strategy. Some combination strategies are based on techniques from experimental design.

This section first explains the different coverage criteria, normally associated with combination strategies and then briefly describes the combination strategies that were identified in the literature. The combination strategies have been organized into different classes based on the amount of randomness of the algorithm and according to how the test suites are created. Figure 2.1 shows an overview of the classification scheme. The combination strategies labeled non-deterministic all depend to some degree on randomness. A property of these combination strategies is that the same input parameter model may lead to different test suites. The simplest non-deterministic combination strategy is pure random selection of test cases. The group of non-deterministic combination strategies also includes two heuristic methods, CATS and AETG.

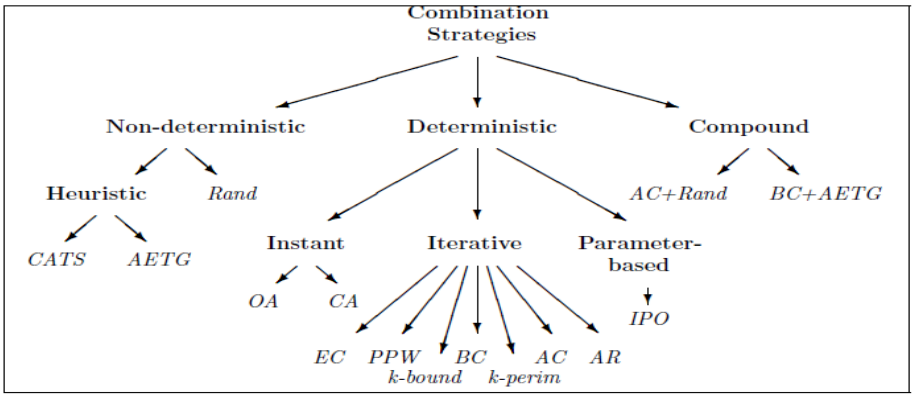

Figure1. *Classification Scheme for Combination Strategies*

The deterministic combination strategies group is further divided into three subgroups, instant, iterative, and parameter-based. All of these combination strategies will always produce the same result from a specific input parameter model. The two instant combination strategies, Orthogonal Arrays (OA) and Covering Arrays (CA), produce the complete test suite directly. The largest group of combination strategies is iterative. They share the property that the algorithms generate one test case at a time and add it to the test suite.

Each Choice (EC), Partly Pair-Wise (PPW), Base Choice (BC), All Combinations (AC), and Anti-random (AR) all belong to the iterative combination strategies. The parameter-based combination strategy, In Parameter Order (IPO), starts by creating a test suite for a subset of the parameters in the input parameter model. Then one parameter at a time is added and the test cases in the test suite are modified to cover the new parameter. Completely new test cases may also need to be added.

Like many test-case selection methods, combination strategies are based on coverage.. The following subsections define the coverage criteria satisfied by combination strategies are included. Each-used (also known as 1-wise) coverage is the simplest coverage criterion. 100% each-used coverage requires that every interesting value of every parameter is included in at least one test case in the test suite. 100% Pair-wise (also known as 2-wise) coverage requires that every possible pair of interesting values of any two parameters are included in some test case. Note that the same test case may cover more than one unique pair of values.

A natural extension of pair-wise (2-wise) coverage is t-wise coverage, which requires every possible combination of interesting values of t parameters be included in some test case in the test suit, t-wise coverage is formally defined.A special case of t-wise coverage is N -wise coverage, where N is the number of parameters of the test object. N -wise coverage requires all possible combinations of all interesting values of the N parameters be included in the test suite.

The each-used, pair-wise, t-wise, and N -wise coverage criteria are purely combinatorial and do not use any semantic information. More coverage criteria can be defined by using semantic information. Cohen et al. indicate that valid and error parameter values should be treated differently with respect to

coverage. Normal values lie within the bounds of normal operation of the test object, and error values lie outside of the normal operating range. Often, an error value will result in some kind of error message and the termination of the execution. To avoid one error value masking another author suggests that only one error value of any parameter should be included in each test case. This observation was also made and explained in an experiment also.

4. PROPOSED SYSTEM

The design of the proposed system is as shown below. The system after careful analysis has been identified to be presented with the following modules: Input module it takes the inputs as number of parameters and the number of values for those parameters for generating the test cases. Internal Module presents a heuristic technique that is being implemented for generating the optimized pairwise tests cases. Initially this module, the heuristic technique takes the number of parameters and their values takes as input and sends the optimized result to the output module and finally the output module displays the optimal pairwise test suite as output.

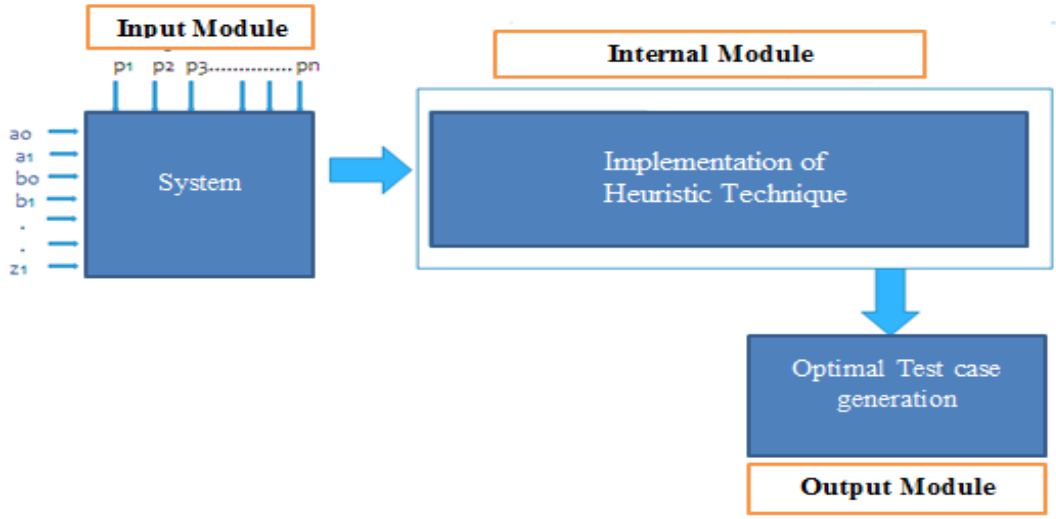

Figure2. *Design of the Implementation System*

Algorithm

Step 0: Begin

Step 1: Read the number of parameters

Step 2: Read the number of values for each parameter

Step 3: Read the values for those parameters

Step 4: See the initial parameter and domain values matrix as shown in table 1

Step 5: Construct the Minimal Test suite matrix

Step 5.1: (a) Consider the initial combination of parameter input pairs ie. AB

(b) Then calculate step value = AB pair values /Number of domain values in parameter A

(c) Fill the initial column in such way that by considering step value.

Step 5.2 For the second column repeat the second parameter values consecutivelyuntil the all the pair is to be filled.

Step 6: Repeat the Step 6 until final test suite matrix is gets filled.

Step 7: Print the Optimized test suite Matrix

Step 8: Stop

The optimized pairwise test cases generated by using our strategy are as shown in the table 2.

Table2. *Test Case Vectors*

Test vector Number	Test Vector elements			
	Element-A	Element-B	Element-C	Element-D
t1.	a0	b0	c0	d0
t2.	a0	b1	c0	d1
t3.	a1	b0	c1	d0
t4.	a1	b1	c1	d1
t5.	a2	b0	c2	d0
t6.	a2	b1	c2	d1
t7.	a0	b1	c3	d0
t8.	a1	b0	c3	d1

5. RESULTS

The numbers of test cases generated for different systems which can be defined using various parametric combinations are shown in the table 1

Table1. *Parametric Configurations of Systems Considered For Testing*

Serial Number	System	Number of input variable	Number of selected input variables	Number of domain values considered
1	S1	4	4	3
2	S2	4	4	3
3	S3	13	13	3
4	S4	61	15	4
			17	3
			29	2
5	S5	75	1	4
			39	3
			35	2
6	S6	100	100	2
7	S7	20	20	10

From the above table, it is seen that the number of test cases generated by our technique is quiet minimal for considering any of the system configurations

Table2. *Pair Wise Test Set Sizes For New Strategy And Other Algorithms.*

System	S1	S2	S3	S4	S5	S6	S7
All Pairs	12	10	22	41	30	16	664
New Strategy	8	9	9	12	12	4	100

6. CONCLUSION

Hence we concluded that a heuristic technique was implemented to generate the optimized pairwise test cases for testing WEB based Applications. We have implemented this test generation algorithm and have shown some empirical results. When used properly, pair wise test set generation is an important technique that cans helps to produce minimal test cases for testing WEB applications. The experiment results illustrated that our proposed heuristic technique will yields better results when comparing to other deterministic techniques like All pairs, IPO etc.. As mentioned earlier, pairwise testing (or 2-way testing) is a special case of n-way testing. Our proposed strategy presented in this project can be easily extended for n-way testing. We are investigating possible improvements of our algorithm without increasing time complexity.

REFERENCES

[1] . S.A Ghazi, M.A Ahmed, "Pair-wise test coverage using genetic algorithms," CEC '03. The 2003 Congress on Evolutionary Computation, vol.2, no., pp.1420,1424 Vol.2, 8-12 Dec. 2003.

[2] . J.D., McCaffrey, "Generation of Pairwise Test Sets Using a Genetic Algorithm," 33rd Annual IEEE International Computer Software and Applications Conference," COMPSAC '09. vol.1, no., pp.626-631, 20-24 July 2009.

[3] . J.D., McCaffrey, "An Empirical Study of Pairwise Test Set Generation Using a Genetic Algorithm," 2010 Seventh International Conference on Information Technology: New Generations (ITNG), vol., no., pp.992-997, 12-14 April 2010

[4] . S Maity and A Nayak, "Improved test generation algorithms for pair-wise testing,". 16th IEEE International Symposium on Software Reliability Engineering, 2005. ISSRE 2005, vol., no., pp.10 pp., 244, 1-1 Nov. 2005.

[5] . Kewen Li, and Zhixia Yang, "Generating Method of Pair-Wise Covering Test Data Based on ACO," International Workshop on Education Technology and Training, 2008, and 2008 International Workshop on Geoscience and Remote Sensing. ETT and GRS 2008. vol.2, no., pp.776-779, 21-22 Dec. 2008

[6] . A Calvagna, A Gargantini, and E Tramontana, "Building T-wise Combinatorial Interaction Test Suites by Means of Grid Computing," 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009. WETICE '09. vol., no., pp.213-218, June 29 2009-July 1 2009.

[7] . Xiaoying Pan, and Hao Chen, "Using Organizational Evolutionary Particle Swarm Techniques to Generate Test Cases for Combinatorial Testing," 2011 Seventh International Conference on Computational Intelligence and Security (CIS), vol., no., pp.1580,1583, 3-4 Dec. 2011.

[8] . Wang Jianfeng, and Jiang Shouda, "An Improved Algorithm for Test Data Generation Based on Particle Swarm Optimization," 2011 First International Conference Instrumentation, Measurement, Computer, Communication and Control, vol., no., pp.404-407, 21-23 Oct. 2011.

[9] . Xiang Chen, Qing Gu, Jingxian Qi, and Daoxu Chen, "Applying Particle Swarm Optimization to Pairwise Testing," 2010 IEEE 34th Annual Computer Software and Applications Conference (COMPSAC), , vol., no., pp.107,116, 19-23 July 2010.