ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 3, Issue 10, October 2016

"Advanced Combinatorial Interaction Testing System by Increasing Efficiency and Producing More Deterministic Results"

MD. Karishma,
M.Tech in Software Engineering,
Department of CSE,
JNTUA College of eEngineering, Ananthapuramu

Abstract: **-- Combinatorial Interaction Testing (CIT) is a black-box testing technique which tests the software with all required combinations to detect faults. The importance of CIT is to reduce the software testing cost and increasing the effectiveness. In the existing literature, there are primarily two tools they are Covering Arrays by Simulated Annealing (CASA) and Advanced Combinatorial Interaction Testing System (ACTS). Among them, ACTS is used to build the t-way test sets. It maintains the generation of a test set where 't' value ranging from 1 to 6 which allows the user to identify constraints and must be satisfied to be legal. It is used to generate covering array faster in which constraints play an important role to increase the efficiency. Constraints are limitations which should be satisfied by a set. By this it is easy to find a test suite which increases the fault detection rates. The drawback here is if constraints are increased, then it is difficult to apply Combinatorial Testing which is not efficient. So, in the proposed work to overcome this drawback and to increase the efficiency certain invalid combinations should be removed from the test set. Therefore, the efficiency is increased by decreasing the number of constraints.**

I. INTRODUCTION

The Combinatorial Testing (CT) is a very effective software testing strategy, which is applied to the combinatorial design. In CT, more faults are created by interactions and the source code access is not required. Assume a system having 'k' parameters where all combinations to be covered at least once. In t-way Combinatorial Testing, when the test parameters are designed correctly, all defects are caused by less than 't' parameters. To test a test suite its strength depends upon the level of interactions. Interactions at higher strengths can detect the faults which are not covered by lower strengths. To detect the faults earlier CIT makes a set of test cases to be prioritised.

ACTS generate a T-way test set up to 6-way coverage and the IPOG is the main algorithm for the generation of tests which supports features such as handling a constraint and generating a mixed strength test. ACTS has mainly two modes to generate a test set they are scratch and extend. If a test set is built from scratch, it is a scratch mode and if an existing test set is extended, it is an extend mode. In most systems, constraints may occur and the condition is that some combinations are invalid. So, ignoring constraints leads to invalid test. Invalid combinations should be removed from the test set. Assume a system having p1, p2, ..., p10 in which a default structure is defined for all parameters using strength 2. For instance, create a relation having four parameters p2, p4, p5, p7¬ using strength 3. If they are adjacent, then the 3-way interactions can prevent defects.

ACTS generate an existing test set, but it is incomplete for some recently included parameters and values or test strength is increased. To save the earlier effort in the testing process the existing test set should be extended. Based on the user's observation, if certain parameters are having a higher extent, then additional relations can be created. The IPOG strategy is generalised from pairwise to multi-way testing. First, as practical applications have arbitrary configurations it is not necessary to lay restrictions on the system configuration. Second, it must compromise with a large number of interactions.

II. RELATED WORK

Covering Arrays by Simulated Annealing (CASA) is freely available tool to handle constraints which are explicitly specified by the user. It is used to generate smaller covering arrays and also to avoid experimental bias. The Automatic Efficient Test Generator (AETG) can complete the partial test cases by

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 3, Issue 10, October 2016

filling the values for the missing fields. It is most widely used tool for generating a test suite for Combinatorial Testing. They construct a set of tests by which each test case can generate as many uncovered combinations as possible. It is a one-row-at-a-time variation where a single row of the array is constructed at each step until it covers all T-sets. Combinatorial Test services (CTS) is used for mathematical constructions to generate covering arrays by which the computations involved use typically lightweight and they are immune to any combinatorial effect. In some special cases, they are extremely fast and can produce optimal test sets. Orthogonal Array Test System (OATS) uses orthogonal array for PXM/StarMAIC system to generate test cases. The generated test cases can detect many errors that had never been detected previously. Advanced Combinatorial Testing System (ACTS) is used to generate test sets that ensure t-way coverage of input parameter values which can compute test for 2-way through 6-way interactions. It can produce large test suites but it is less effective at finding faults quickly. Hence, the efficiency is increased by decreasing the number of constraints.

III. METHODOLOGY

In Combinatorial Testing, the first step is to spread all parameter values of t-way combinations with strength ‘t’ is to discover a set of test cases. In testing, a huge test suite needs more testing cost and a higher coverage which extends the possibility of detecting failures. An ACTS model generates larger covering arrays. It generates the array that will consistently accept the test case which has larger number of uncovered t-tuples. From the model, covering arrays are generated by rejecting single-valued parameters. For completeness, after generating an array each test case with single-valued parameters is expanded. During the generation of a covering array it will also perform prioritisation. By this, the number of test suites will also be reduced. So that bugs can be detected earlier. The overall aim is to create a 2-dimensional array by covering all t-tuples.

In a direct way, the models of CIT are created from test suites which reject the parameters having single value. The number of combinations depends upon the number of combinations and it consumes more time and harder to count the effort between the constraints. In the modelling phase, some values which are provided by constraints will be rejected from CIT. For example, the suitable test cases do not require additional combinations.

After the covering array is generated, each of the test cases will be prioritised. It completes the iteration for test cases in which one test set will be retained to cover a large number of uncovered t-tuples and each time test suites are generated. Prioritisation can be calculated in two ways.

1) Based on prioritisation criteria, the existing test suite of CT should be reordered and

2) Considering the importance of combinations the ordering of the test should be generated for CT

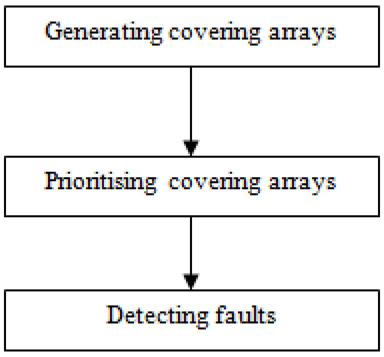

Figure 1: Procedure for detecting faults

When the total number of uncovered t-tuples is maximum all test cases should be gathered and one should be picked randomly. An array is added which makes changes to cover uncovered t-way interactions. For a given test set, a Boolean mapping is considered in which mapping is done between test cases and t-tuples. Here, first the uncovered t-way interactions should be noticed and then recorded. When a test case is newly added this mapping should be updated in order to reduce the recounting of uncovered t-tuples. In testing, according to a standard test, the prioritisation outcome is usually a list of test cases in which higher priority test cases are performed earlier. After performing prioritisation, first the test cases which are important should be tested in order to detect faults earlier.

The fault detection capacity is measured by each prioritised test suite. Here, the concentration is on finding faults by full test suites which not to be caused by single or error value assignments. The test suites which are not present need to be tested with any other parameters. In failure diagnosis, it is necessary to inquire where to locate the failure and then remove it. For covering arrays with lowest strength the percentage of fault detection is not equal to that of covering arrays with higher strength because it contains more test cases. If both the time and resources are plentiful, the more faults can be detected and covering arrays with higher strength are also achieved. In a particular time limit, covering arrays having lowest strength will allow valid fault detection.

A. Forbidden tuples method

The forbidden tuples is an efficient method for handling constraints. Faults are caused only by few combinations of parameters. Their outcome has a great impact on CT. If all faults are caused by combining less than ‘n’ parameters, then n-way combinatorial testing ensures that atmost all faults can be detected. It is used to check whether a complete test set is valid or not. However, forbidden tuples may be invalid for a partial test covering.

Due to more constraints the efficiency is not achieved. So, the number of constraints should be reduced in order to increase the efficiency. The number of constraints can be reduced by removing invalid combinations from the t-way test set. During generation of a test constraints should be identified by the user before they are handled. So, there is one method to specify constraints as a set of forbidden tuples. A forbidden tuple is a combination of values which must not present in any test. Constraints can also be identified as a set of logical expressions and it is a condition which should be satisfied by every test. Logical expressions are briefer when compared to explicit enumeration of forbidden tuples.

In constraint handling, a major step is to check for a validity i.e., checking whether all the constraints are satisfied by a test. This approach can be performed to check whether a test contains no forbidden tuples and needs to maintain the complete list of all the forbidden tuples. During the generation of test, his list should be first generated from the specified logic expressions and then used to perform validity check. A constraint solver is used to perform this check.

B. IPOG-C algorithm

For a constraint solver an IPOG-C algorithm is used which modifies an existing combinatorial test generation algorithm called IPOG [7] to handle constraints and tries to reduce the number of constraints of the constraint solver. In some cases such a constraint cannot be removed but this algorithm tries to solve this problem as much as possible.

This algorithm includes the following three optimizations:

1) A t-way test set must cover all the valid t-way combinations. A t-way combination is valid if a test is covered by at least once. Checking the validity of each t-way combination can be expensive and contains a large number of t-way combinations. So, if a test is found valid, then all the combinations correspond to this test will be valid, and those need not to be checked explicitly.

2) When a validity check is performed, some constraints may not be valid and those need not to be checked. We use a notion called constraint relation graph to identify groups of constraints that are related to each other, which are then used to identify relevant constraints in a validity check. It builds a test set incrementally, i.e., covering one parameter at a time. This incremental approach is leveraged to further reduce the number of invalid constraints.

3) Therefore, the number of invalid combinations will be reduced to the constraint solver by saving previous results.

For this purpose the IPOG-C algorithm is implemented in a combinatorial test generation tool called ACTS. To handle constraints IPOG algorithm is modified which is now known as IPOG-C algorithm to handle constraints. If no constraints are specified, the modified algorithm will generate the same test set as the original IPOG algorithm does.

IPOG-C algorithm makes sure that (1) all the valid t-way tests are covered and (2) all the generated tests are valid. First a validity check is performed on each t-way combination to identify all the valid t-way combinations that need to be covered. Then the invalid combinations will be rejected by which the number of constraints will also be reduced.

1) For each valid combination of values first, add a test for the first't' parameters and then for 't+1' parameters until it covers the t-way test set to the test set.

2) For each test in the test set choose a valid combination that covers a large number of combination of values.

3) Remove the test sets that have invalid combinations

4) For each combination in a set check whether it has been covered by the previous test sets

5) If it already exists then that combination can be removed.

6) Hence the resultant test set does not contain invalid combinations

IV. RESULTS

Practically an experiment is conducted by considering an organisation having projects and members to execute those projects. By taking these parameters a covering array is generated to accept which has large number of uncovered t-tuples. During the generation of a covering array it will also perform prioritisation the number of test suites will be reduced. Therefore, a 2-dimensional array is generated by covering all t-tuples.

After generating covering arrays, a priority is assigned for each test case like 1, 2, 3, 4 and priorities can be named as warning, severe. Thus, the faults can be detected earlier and if the test cases are invalid, then those will be deleted from the test set. Then the faults which are pending and resolved can be displayed in the report.

The graph represents the results between the existing and proposed system. The efficiency is based on the number of bugs that have been detected. In the existing system, both tools are used but ACTS is less effective at finding faults earlier. So, in the proposed system the efficiency is increased. Therefore, the efficiency of the system is achieved and in a graph the proposed results are shown.

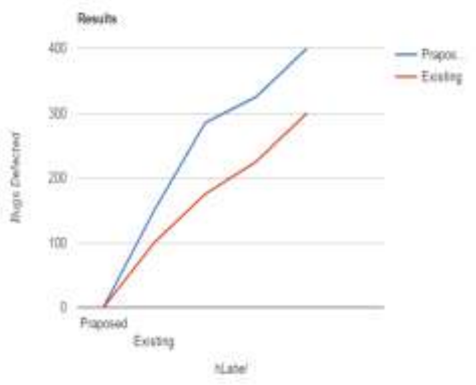

Figure 2: Result showing increased efficiency by detecting faults earlier in the proposed system

V. CONCLUSIONS AND FUTURE WORK

In this paper, a generalized forbidden tuple-based constraint handling method is performed for combinatorial test generation. It is based on the notion of minimizing combinations which in turn reduces the number of constraints. A complete test is valid if and only if it does not contain invalid combinations. The experimental results show that the enhanced constraint handling method is performed better by detecting the faults earlier.

Furthermore, if the number of forbidden tuples is large, then it is difficult for the user to enumerate them. This approach can be continued in the following two directions. First, the performance of this method can be still improved. Second, more experiments can be conducted to evalualte this performance. It can be further investigated by solving the problems like how to measure the complexity of a constraint and how to generate constraints of different complexity levels.

VI. REFERENCES

[1] Justyna Petke, Myra B. Cohen, Mark Harman, and Shin Yoo, "Practical Combinatorial Interaction Testing: Empirical Findings on Efficiency and Early Fault Detection," IEEE Trans. Softw. Eng., vol. 41, no. 9, Sep. 2015.

[2] J. Petke, M. B. Cohen, M. Harman, and S. Yoo, “Efficiency and early fault detection with lower and higher strength combinatorial interaction testing,” in Proc. Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng., Saint Petersburg, Russian Federation, Aug. 2013, pp. 26– 36.

[3] C. Nie and H. Leung, “A survey of combinatorial testing,” ACM Comput. Surv., vol. 43, no. 2, pp. 11:1–11:29, 2011.

[4] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, “IPOG/ IPOG-D: Efficient test generation for multi-way combinatorial testing,” Softw. Test., Verification Reliab., vol. 18, no. 3, pp. 125–148, 2008.

[5] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, “The AETG system: An approach to testing based on combinatorial design,” IEEE Trans. Softw. Eng., vol. 23, no. 7, pp. 437–444, Jul. 1997.

[6] R. C. Bryce and C. J. Colbourn, “Test prioritization for pairwise interaction coverage,” ACM SIGSOFT Softw. Eng. Notes, vol. 30, no. 4, pp. 1–7, 2005.

[7] M. B. Cohen, C. J. Colbourn, P. B. Gibbons, and W. B. Mugridge, “Constructing test suites for interaction testing,” in Proc. Int. Conf. Softw. Eng., May 2003, pp. 38–48.

[8] D. Kuhn, R. Kacker, and Y. Lei, “Automated combinatorial test methods: Beyond pairwise testing,” Crosstalk, J. Defense Softw. Eng., vol. 21, no. 6, pp. 22–26, 2008.

[9] T. Nanba, T. Tsuchiya, and T. Kikuno, “Using satisfiability solving for pairwise testing in the presence of constraints,” Inst. Electron., Inform. Commun. Eng. Trans., vol. 95-A, no. 9, pp. 1501–1505, 2012.

[10] S. Manchester, N. Samant, R. Bryce, S. Sampath, D. R. Kuhn, and R. Kacker, “Applying higher strength combinatorial criteria to test prioritization: A case study,” J. Combinatorial Math. Combinatorial Comput., vol. 86, pp. 51–72, Aug. 2013.

IFERP connecting engineers... developing research