International Journal of Advance Research in Science and Engineering
Volume No.06, Issue No. 11, November 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

Survey on Cost attentive of test suite reduction for software testing

T.Mahesh Kumar[1] , Dr.K.Subba Rao[2]

[1]CSE,Research Scholar,KL University
[2]CSE,Professor,BVRIT,Narsapur

ABSTRACT

Software testing is a combination of verification and validation. To produce a quality product to the real customers or to satisfy the specific customer companies have to concentrate on both process to be used and outcome of the process. a test case is a perception of a test engineer which are designed to exercise a specific test requirement. During Testing It is not possible to test all of the specified test requirements with a single test case. Combination of test cases belongs to one functionality is called as test suite so, if test suite size increases it leads to increase the number of test cases. Researchers have investigated two approaches for addressing the test-suite size problem that maintain the same coverage as the original test suite test-suite reduction and test-suite prioritization. Test suite prioritization algorithms identify an ordering of the test suite according to some criteria. Test suit reduction is the most imperative approach in which the numbers of the test cases in the test suite are minimized, at the same time covering all the requirements. The problem of test suite optimization has been also formulated as a combination of multiple often contrasting criteria. Many of the test suite reduction approach using the optimization algorithms have been concentrated as an active are of research. The efficient performance of the optimization based test suite reduction depends on the tester with the requirements like choosing some testing criteria to be satisfied, and using an optimization technique to select/order the test cases on the basis of the chosen criteria.

Keywords: Test Case, Test Suite, Test suit reduction, optimization algorithms.

I. INTRODUCTION

Software testing ensures the quality and reliability of a System under Test (SUT) by revealing maximum possible defect. Software testing and retesting occurs continuously during the software development lifecycle. Even though the assurance practice is expensive, it provides putative applications to the development organizations regarding the SUT. Testing is the primary method that is widely adopted to ensure the superiority of the software under development. According to the IEEE definition, a test case is a set of input data and expected output results which are designed to exercise a specific software function or test requirement. During testing, the testers, or the test harnesses, will execute the underlying software system to either examine the associated program path or to determine the correctness of a software function. It is difficult for a single test case to satisfy all of the specified test requirements. Hence, a considerable number of test cases are usually generated and collected in a test suite. Software systems evolve constantly to provide the required functionalities

and to adapt to ever-changing customer needs. However, modifying software can break the previously verified functionalities of the system, causing regression faults. Software regression testing is therefore required in order to detect such faults. The dominant strategy is to rerun test cases that are available from an earlier version of the product. As software grows and evolves, so too do the accompanying test suites. Over time, some test cases in a test suite may become redundant as the requirements executed by them are also executed by other test cases in the test suite. Due to time and resource constraints for re-testing the software every time it is modified, it is important to develop techniques that keep the test suite size manageable by removing those test cases that may have become redundant with respect to the coverage of program

Requirements.

Test-suite reduction and test-suite prioritization. Test-suite reduction is also known as test-suite minimization algorithms. Test suite reduction algorithms identify a reduced test suite that provides the same coverage of the software as the original test suite. Test suite prioritization algorithms identify an ordering of the test suite according to some criteria. Test suit minimization is the most imperative approach in which the numbers of the test cases in the test suite are minimized, at the same time covering all the requirements.

Since the cost of the testing increased because of testing the software or the test suite with every possible test case to full fill the test requirements, automation seems to be the key solution for improving the efficiency and effectiveness of the testing process. And so the test minimization problem is reflected as the NP hard optimization problem and it is equivalent to the cover set problem. The problem of test suite optimization has been also formulated as a combination of multiple often contrasting criteria. Many of the test suite reduction approach using the optimization algorithms have been concentrated as an active are of research. The efficient performance of the optimization based test suite reduction depends on the tester with the following requirements; (i) choosing some testing criteria to be satisfied, and (ii) using an optimization technique (e.g., greedy or search-based algorithm) to select/order the test cases on the basis of the chosen criteria. The criteria used for the requirement are code coverage, program modification, execution cost, past fault information. Search-based optimization techniques can be useful for regression test selection; usually such techniques only try to find (near) optimal solution with respect to some fitness functions. They do not suite reduction some time leads to the stagnation. The optimization of the test suite is performed using the multiple criteria as well as the single criteria. In some approaches, the different criteria are combined to the single objective function.

Even though the optimization based procedure is found to be the ideal for the test suite reduction, the reduction procedure with the proper criteria selection for the suite reduction selecting the test cases covering all the requirements of the test is necessary which is lacking in the existing works. And also the regression testing by the generated test case must reduce the cost of testing, with reduced complexity .In addition the reduced test suite must also result without any loss in the fault detection effectiveness.

International Journal of Advance Research in Science and Engineering
Volume No.06, Issue No. 11, November 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

II LITERATURE SURVEY OF RELATED WORKS

Table 1 list the literature of the existing works related to the test suite reduction in the regression testing. The advantages and the disadvantages of the literatures are also highlighted in the survey table.\

Author	Adopted Methodology	Advantages	Disadvantages
Dennis Jeffrey *et al.* [15]	Heuristic approach – HGS algorithm	Test suite reduction with selective redundancy is achieved	Less fault detection loss leads to the increase in the size of the test suit, increasing the cost and complexity of thetesting
SivasjMirarab*et al.* [16]	Programming IP based technique, Greedy RTP technique, Hybrid IP and RTP based technique	Optimize the test suite and select the test case capable of revealing the faults	The running time of reduced test suite is not considered in the test minimization, Limitation in consideration of the test requirement
Luciano S. de Souza *et al.* [17]	Search based technique	Test case is selected with respective execution time reducing the overall regression testing time	Requirement coverage considered in the optimization is limited
Gordon fraser*et al.* [18]	Memetic Algorithm	Hybrid search based algorithm incorporating the local and the global search easing the test suite minimization with requirement coverage	Advancement in the parameter control technique degrades the performance
Chu-Ti Lin *et al.* [19]	Irreplaceable test	Represent and	Application domain

	procedure	generate the reduced test suite with low execution cost	change affects theperformance of reduction, test case prioritization problem is left out
SrividhyaJeyaprakahsh*et al.* [20]	genetic approach	Test suite reduction considering multi objective criteria	The overall cost of the execution is increased
Annibalepanichella*et al*, [21]	Diversity genetic Algorithm	test suite reduction preserving the diversity, independent of number of test criteria	Difficulty in customization of the diversity in the search based approaches
SumitDahiya*et al.* [22]	test selection using class sequence and activity diagrams	selection by considering the semantic changes in the operations	Increase in the regression time

III. PROPOSED METHODOLOGY:

The primary intention of this research is to design and develop a technique for test suite reduction using binary fractional firefly algorithm. This work aims to bring an new optimization algorithm called, BEfirefly algorithm to select test cases optimally with two constraints, i) It should satisfy all the test requirement ii) Cost metrics should be minimum. Based on these two constraints, BEfirefly algorithm will be developed by modifying the popular optimization algorithm called, Firefly algorithm. At first, initial solutions are generated randomly with the constraint that selected test cases in each and every solution should satisfy the entire test requirement. Then, fitness will be evaluated using the total cost which is the aggregated execution time of all the selected test cases. The solution set which have the minimum aggregated Cost measure is selected as the best solution set. The generation of the new solution set and its evaluation is done with the help of the proposed BEfirefly algorithm, where the generation of new solution will be modified with the help of the new formulae.

IV. CONCLUSION

According to the literature review, we have come to know about some of the contributions ade in the test suite reduction. The proposed methodology will be evaluated and acts like bridge for those who want to work on test suite reduction algorithms. By implementing BEfirefly algorithm can be evaluate the performance metrics like, cost and execution time.

The quantitative analysis of the proposed methods will be done using five metrics namely, Suite Cost Reduction, Suite Size Reduction, Improvement in Cost and Size.

REFERENCES

[1] Myers, G. J., Sandler, C., &Badgett, T., "The art of software testing", John Wiley & Sons, 2011.

[2] P. Ammann, J. Offutt, "Introduction to Software Testing", Cambridge University Press, 2008.

[3] D. Binkley, "Semantics guided regression test cost reduction", IEEE Transaction on Software Engineering, vol. 23, no. 8, pp. 498–516, 1997.

[4] H. Zhong, L. Zhang, H. Mei, "An experimental study of four typical test suite reduction techniques", Inform. Softw.Technol., vol. 50, no.6, pp. 534–546, 2008.

[5] Harrold, M.J., "Testing evolving software", J. Syst. Softw., vol. 47, pp. 173–181, 1999.

[6] M.J. Harrold, R. Gupta, and M.L. Soffa, "A Methodology for Controlling the Size of a Test Suite," ACM Trans. Software Eng. and Methods, vol. 2, no. 3, pp. 270-285, July 1993.

[7] S. Elbaum, A. Malishevsky, and G. Rothermel, "Prioritizing Test Cases for Regression Testing," Proc. ACM Int'l Symp. Software Testing and Analysis, pp. 102-112, Aug. 2000.

[8] A. De Lucia, M. Di Penta, R. Oliveto, and A. Panichella, "Estimating the evolution direction of populations to improve genetic algorithms," in Proc. 14th Int. Conf. Genetic Evol. Comput. Conf., pp. 617–624, 2012.

[9] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms. New York, NY, USA: Wiley, 2001.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: NSGA-II," IEEE Trans. Evol.Comput., vol. 6, no. 2, pp. 182–197, Apr. 2002.

[11] M. J. Harrold, R. Gupta, and M. L. Soffa, "A methodology for controlling the size of a test suite," ACM Trans. Softw. Eng. Methodol., vol. 2, pp. 270–285, 1993.

[12] S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in Proc. ACM/SIGSOFT Int. Symp.Softw.Testing Anal., pp. 140–150, 2007.

[13] A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum, "Cost-cognizant test case prioritization," Dept. Comput. Sci. Eng., Univ. Nebraska-Lincoln, Lincoln, Nebraska, USA, Tech. Rep. TR-UNL-CSE-2006-0004, Mar. 2006.

[14] W. E.Wong, J. R. Horgan, S. London, and A. P. Mathur. Effect of test set minimization on fault detection effectiveness. Software - Practice and Experience, vol. 28, no. 4, pp. 347–369, April 1998.

[15] D. Jeffrey, Neelam Gupta, "Test Suite Reduction with Selective Redundancy", In proceedings of IEEE International Conference on Software maintainance, pp. 549-558, 2005.

[16] SiavashMirarab, SoroushAkhlaghi, and LadanTahvildari, "Size-Constrained Regression Test Case Selection Using Multicriteria Optimization", IEEE Transaction on Software Engineering, vol. 38, no. 4, pp. 936-956, 2012.

[17] Luciano S. de Souza, Ricardo B.C.Prudêncio, Flavia de A.Barros, Eduardo H.daS.Aranha, "Search based constrained test case selection using execution effort", Expert System with Applications, vol. 40, pp. 4887–4896, 2013.

[18] Gordon Fraser, Andrea Arcuri, Phil McMinn, "A Memetic Algorithm for whole test suite generation", The journal of Systems and Software, 2014.

[19] Chu-Ti Lin, Kai-Wei Tang, Gregory M. Kapfhammer, "Test suite reduction methods that decrease regression testing costs by identifying irreplaceable tests", Information and Software Technology, vol. 56, pp. 1322–1344, 2014.

[20] SrividhyaJeyaprakash, K. Alagarsamy, "A Dsitinctive Approach for test Suit optimization", Procedia Computer Science, vol 62, pp. 427-434, 2015.

[21] AnnibalePanichella, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia, "Improving Multi-Objective Test Case Selection by Injecting Diversity in Genetic Algorithms", IEEE Transactions on Software Engineering, vol. 41, no. 4, pp. 358-383, 2015.

[22] SumitDahiya, Rajesh K. Bhatia, Dhavleesh Rattan, "Regression test selection using class, sequence and activity diagrams", IET Software, pp. 1-9, 2016.

[23] T.Y. Chen, M.F. Lau, "A new heuristic for test suite reduction", Information and Software technology, vol. 40, 347-354, 1998.

[24] James A. Jones and Mary Jean Harrold, "Test-Suite Reduction and Prioritization for Modified Condition/Decision Coverage", IEEE Transactions on Software engineering, vol. 29, no. 3, pp. 195-209, 2003.

[25] Saif Ur RehmanKhana, Sai Peck Lee, Raja Wasim Ahmad, Adnan Akhunzada, Victor Chang, "A survey on Test Suite Reduction frameworks and tools", International Journal of Information Management, vol. 36, pp. 963-975, 2016.

[26] Xin-She Yang, "Firefly Algorithm, Stochastic Test Functions and Design Optimisation ", International Journal of Bio-Inspired Computation, Vol. 2, no. 2, pp. 78-84, March 2010.