An Effective Test Suite Reduction Using Priority Cost Technique

Jyoti Prakash Rout
School of Computer Engineering
Kalinga Institute of Industrial Technology, KIIT University
Bhubaneswar, India
Jyotiprakash161@gmail.com

Ramshankar Mishra
School of Computer Engineering
Kalinga Institute of Industrial Technology, KIIT University
Bhubaneswar, India
Ramshankar12@gmail.com

Rajanikanta Malu
Professor, School of Computer Engineering
Kalinga Institute of Industrial Technology, KIIT University
Bhubaneswar, India
malubbsr@yahoo.co.in

Abstract—Effective testing can develop quality software with higher productivity at a lower cost. As the software is modified and new test cases are added to the test suite, the size of the test suite grows and the cost of testing is also increases. In order to reduce the cost of testing researcher have investigated the use of test suite reduction technique. Redundancy in test suite increases the execution cost. Weighted set covering Technique can be used to solve the test suite minimization. This paper adapts the cost and priority factor to reduce the test suite. We conduct a comparison between the different algorithm and design a formula for the test suite reduction.

Keywords-Software testing; Test suite reduction; Redundancy; Weighted Set Covering; Minimal test suite; Total cost; Priority

I. INTRODUCTION

Software testing is a critical phase of software development and crucial to software quality. Before testing a software, tester have to classify the objectives. Testing objective are the testing requirements. Since these test requirements are also satisfied by other test cases in the test suite that were added to the cover modifications in the latest versions of software, some test case in that test suite may be redundant. In software testing time and resource are always constants. Reduce the execution time in testing is a big challenging task. So, the technique of reducing time, cost and test cases is called test case minimization. Suppose T is the set of all covering requirements of test cases so, that T' is the reduced or minimized test suite. Testing requirement or the input requirement for to check the bug are called the test cases. A suitable test case that fit for the software and without any redundancy term or the minimized test cases are called test suite. That is the suitable test requirement for the software. The test case minimization is a NP-complete problem.

The test case reduction aims to finding a minimal subset of the test suite that can cover all requirements. It can be defined as follows [1]:

Given: A set of testing requirement R= { r_1, r_2,…..,r_n } that must satisfied the given test coverage of the program and a set of subsets { T_1, T_2,…..,T_n } of a test suite T= { t_1, t_1,…..,t_n }. They are associated thir exist t_i ἐ T_i.

Problem: Find the cardinality of the given test cases T by removing the redundancy term.

Chen and Lau have proposed a heuristics GE and GRE algorithm [2, 3]. Chen and Lau propose tow dividing strategies one is essential strategy and other one is 1-to-1 redundancy strategy [5]. Harrold et al. propose the heuristic algorithm H to reduce the size of the test cases [6]. This uses the essentialness and first group the test requirement then repeatedly reduces the test cases and finally remove the redundant test case in the test requirement [6]. Rui and Jianhui have proposed a Adapted Q-M(Quine-McCluskey) algorithm by using the Boolean function. In this method they simply follow the Q-M algorithm and the Karnaugh Map to solve the test cases minimization [7]. The Adapted Q-M algorithm is very efficient for testing they use the prime implicant factor and essential prime implicant factor in minimization [7]. S.Nachiyappan use an evolutionary algorithm i.e

GA(Genetic Algorithm) to reduce the redundancy test cases [8]. The GA is consist of combination of Block-based coverage and test execution cost [8].

This paper is mainly form two kind of ways for regression testing one is test case selection and other one is reduction by means of WSC(Weighted set coverage) [9][10]. We make a table for the test requirement according to the coverage criteria and then reduce it to a minimized form. Through the comparison with other algorithm, we can prove our technique generates a reduced test suite which has minimal cost at same time.

The rest paper organized as follows section II introduce the Background knowledge of WSC, the test coverage criteria and test suite reduction model. Section III proposed method and provide the example and section IV conclude the paper.

II. PRELIMINARIES

A. Test Case Selection

The test case selection procedure is in this way first gather the test coverage requirement. After that derive the test case from the test requirements[11]. The costs of the requirements are derived from the summation of coverage.

1) Coverage Criteria

There are many coverage criteria to satisfy the test coverage such as state coverage, edge coverage, branch coverage, definition use pair coverage etc. Coverage is represented as C and according to the coverage the test cases are generated represented as T. In coverage many redundant term are present so that minimization is needed to remove the redundant term. The minimized test cases represented as T**'**[11].

i) State coverage: It is the coverage criteria that cover all state at least once.

ii) Edge Coverage: It is the coverage criteria that cover all edges of the tested system and all edges visited at least once.

iii) Branch coverage: Every branch of a graph is at least visited once and it is a basic coverage method.

iv) Definition use pair coverage: Definition use pair is use in the program that coverage all program like variable name, Varchar etc.

2) Cost of the test suite

In a given test requirement the cost of the each test cases is equivalent to the summation of cost of its covered test requirements.

Definition: In a binary relation table the cost of the test requirement can be represented by the using the T and R. Cost(T_i) = summation of cost($t_{i)}$.

According to the test cases t in T and req(t) set consist of test requirement, which covered by test case t, and req(T)=R. If there is Req(T/t)≠R, t is called necessary test cases. If Req(T/t)=R, t is called redundant test case and for Req(t_i) is a subset of Req(t_j) then it is called a 1-1 redundancy.

B. Weighted set coverage

Finding minimal cardinality set coverage is the main goal of the coverage criteria. The minimal coverage is the sub set of the total universal coverage [9][10]. Let C is the cardinality of coverage and S is the total coverage then the weighted set coverage is C is a sub set of S. It based on the Greedy algorithm.

III. PROPOSED METHOD

In this paper we proposed a new mathematical formula to reduce the cost of regression testing by minimizing the size of the test suite using priority based cost. Based on Weighted set coverage [9][10], cost of the test requirement and test cases the priority factor will be calculated.

A. PROPOSED ALGORITHM

INPUT: Test-Requirement **R**

Test Cases **T**

Cost calculation **C**

Calculate the Probability factor **P**

OUTPUT: Reduced Test Suite **T'**

Begin

Read the test cases and their cost.

Calculate the Priority factor

Loop

Select the test cases according to their priority factor

Remove the Low Priority factor value

Else

Minimize the table and go to next priority factor

End

B. Priority Factor

- The priority Value will be set as Priority (t_i)=cost(t_i)/Req(t_i).
- According to the cost and requirement value the priority can be calculated and the low priority value in that requirement removed to reduce the redundancy and 1-to-1 redundancy.

C. Test Suite Reduction

Remove the unnecessary test cases and eliminates the 1-to-1 redundancy in the testing requirement is the main purpose. On the basic of WSC minimization [9][10] is will give a higher efficiency to the testing system. The test suite reduction is as follows

Step 1) Determine the test cases whether it can cover the test requirement if not execute step 2.

Step 2) Repeatedly execute and eliminate redundant tem. If requirement has only one test cases the select the test cases. Then update the test suite. After updating calculate the priority factor of each test requirement and remove the smallest priority value.

Step3) Last step is to determine the whether any uncovered test requirement have been covered if reaming then repeat the step 2 and step 3. Then the final output is minimized one.

D. Example

Consider the test cases T={t_1,t_2,t_3, ….,t_6} and let requirement of test cases are R={R_1,R_2,R_3…,R_{10}}. Requirement according to the test cases are t_1={R_1,R_2,R_3,R_4,R_5,R_6,R_{10}}, t_2={R_1,R_2,R_4,R_5,R_{10}}, t_3={R_6,R_8}, t_3={R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10}}, t_5={R_3 R_4 R_5 R_7 R_8 R_{10}}, t_6={R_3 R_4 R_5 R_6 R_9 R_{10} }.

Table 1

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
t_1	1	1	1	0	1	1	0	0	0	1
t_2	1	1	0	1	1	0	0	0	0	1
t_3	0	0	0	0	0	1	0	1	0	0
t_4	1	1	1	1	1	1	1	1	1	1
t_5	0	0	1	0	1	0	1	1	0	1
t_6	0	0	1	1	1	1	0	1	1	1
C	2	1	3	1	2	1	1	3	1	3

From the given Table 1 calculate the cost of the all test cases. Cost(t_1)=2+1+3+2+1+3=12, cost(t_2)=2+1+1+2+3=9, cost(t_3)=1+3=4, cost(t_4)=2+1+3+1+2+1+1+3+1+3=18, cost(t_5)=3+2+1+3+3=12, cost(t_6)=3+1+2+1+3+1+3=14.

Then next check the redundant term. If not present then calculate the priority factor. So first calculate the cardinality of the test cases. |req(t_1)|=6, |req(t_2)|=5, |req(t_3)|=2, |req(t_4)|=10, |req(t_5)|=5, |req(t_6)|=7. Thus the prority(t_i)=Cost(t_i)/|req(t_i)|. Priority(t_1)=12/6=2, priority(t_2)=9/5=1.8, priority(t_3)=4/2=2, priority(t_4)=18/10=1.8, priority(t_5)=12/5=2.4, priority(t_6)=14/7=2. Thus t_2 and t_6 have the lower priority factor so, that select t_2 and t_6.

Next repeat the same process after deleting two test cases the table will be.

Table 2

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
t_1	1	1	1	0	1	0	0	1	0	1
t_3	0	0	0	0	0	1	0	1	0	0
t_4	1	1	1	1	1	1	1	1	1	1
t_5	0	0	1	0	1	0	1	1	0	1
c	2	1	3	1	2	1	1	3	1	3

Calculate the cardinality factor |req(t_1)|=6, |req(t_3)|=2, |req(t_4)|=10, |req(t_5)|=5 and the priority(t_1)=12/6=2, priority(t_3)=4/2=2, priority(t_4)=18/10=1.8, priority(t_5)=12/5=2.4.Thus select and remove t_4. So all selected reduced output are {t_2, t_4, t_6}. The the cost of T' will be Cost(T')=cost(t_2)+cost(t_4)+ cost(t_6)=41.

E. Comparison

Table 3

Algorithm	Technique	Advantages	Limitation	Future Work
CBR Siripong et al [12]	Coverage, Reach ability, Complexity Filtering, Impact Factor, Path Coverage Filter	Remove the Redundancy, Removing the Unnecessary test cases, Controlling the growth of test cases	Path Coverage may not be an effective coverage for huge system	Apply other coverage factor to reduce the unnecessary test cases.
GE Chen & Lau [2]	Greedy Algorithm, using 1 to 1 redundancy	Construction of optimal representative set	Essential test case are not found	GRE was developed for the Essential test case.
GRE Chen & Lau [3]	Greedy Algorithm, using 1 to 1 redundancy	Construction of optimal representative set	Can not Guarantee the minimum set of test cases	H was Developed for the minimum set of test cases.
H Harrold et al [6]	Group the requirements select the essential test cases	Repeatedly selects the test case that can satisfy a maximum number of unsatisfied requirement	Do not Guarantee the obtained representative sets to be optimal	Enhance Zero – one optimal path set selection method to solve the optimal representative set selection problem
QM Zhang Rui et al [7]	Boolean Function, use the K-map	To find the essential prime implicants and prime implicants	Optimal path set selection problem	To solve Optimal path set selection problem developed a Adopted QM Algorithm.
Algorithm	Technique	Advantages	Limitation	Future Work
GA Ma Xueying et al.[8]	Block-based coverage and test execution cost	Prioritizing the test suite	Limited time is allotted for the testing test cases.	In order to promote these test cases use the mutation and crossover operation.
Evolutionary GA Nachiyappan etal [8]	Adaptive cross-over applied to the individual test cases, mutation, fitness function	Good optimal size test set and it is a highly adaptive method	Execution time is not consider	Fault finding using metric
QEA S.Nachiyappan	Quantum chromosomes	Better performance than	Test time will increase rapidly	Improve test efficiency

et al[8]		the classical optimization algorithm	as the scale of the problem increase	
WSC Shengwei Xa et al [9]	Use the Test coverage, edge coverage, branch coverage, U-method	Reduce test suite and total cost	Not valid for large scale test suite NP-complete problem	To developed better WSC
Priority Based	Use the test coverage, edge coverage, branch coverage , cost and the priority based cost	Reduce the redundant and unnecessary test cases	NP-complete problem	To reduce the redundancy and improved priority factor

In Table 3 it compare about the technique, advantages, limitation and future work. Hence the priority based technique is an efficient technique to minimize the test cases.

F. **Conclusion**

In this paper we introduce the priority based minimization to reduce the test cases. It is very efficient and capable to reduce the redundancy term. In this work we only consider some of the coverage but there are many coverage criteria are there. It may possible by taking other coverage criteria. However our method needs more research for optimistic result and analysis. The future work is to develop new factor like priority and develop efficient minimization.

REFERENCES

[1] M.J. Harrold, R. Gupta, and M.L. Soffa, "A methodology for controlling the size of a test suite", *ACM Transactions on Software Engineering and Methodology*, Vol. 2, No. 3, 1993, pp. 270−285.

[2] A.J. James and J.H. Mary, "Test-Suite Reduction and Prioritization for Modified Condition/Decision Coverage", *IEEE Transactions on Software Engineering*, Vol. 29, No. 3, 2003, pp. 195-209.

[3] T.Y. Chen and M.F. Lau, "A new heuristic for test suite reduction", *Information and Software Technology*, Vol. 40, Nos. 5-6, 1998, pp. 347–354

[4] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press. Zhong, H., Zhang. L., and Mei, H; "An experimenal comparison of four test suite reduction techniques ,"European Journal of Scientific Research, vol 49, pp.332-353, May 2006

[5] T.Y. Chen and M.F. Lau, "Heuristics towards the optimization of the size of a test suite", *Proceedings of the Third International Conference on Software Quality Management*, Seville, 1995, pp. 415–424.

[6] T.Y. Chen and M.F. Lau, "On the divide-and-conquer approach towards test suite reduction", *Information Science*, Vol. 152, 2003, pp. 89−119

[7] Rui Zhang, Jianhui Jiang, Jie Yin" A New Method for Test Suite Reduction" The 9th International Conference for Young Computer Scientists DOI 10.1109/ICYCS.2008.501

[8] S.Nachiyappan, A.Vimaladevi, C.B.SelvaLakshmi "An Evolutionary Algorithm for Regression Test Suite Reduction" Proceedings of the International Conference on Communication and Computational Intelligence – 2010, 2010.pp.503-508

[9] V. Chvatal. "A greedy Heuristic for the Set-Covering Problem," Math. Operations Research, vol.4, no.3, 233-235, Auguest 1979

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein "Introduction to Algorithms", MIT Press, Second Edition, September 2001.

[11] P.Amman and J.Offutt, Introduction to Software Testing, Cambridge University Press 32: Avenue of the Americas, New York, NY 10013-2473, USA. 2008

[12] Siripong Roongruangsuwan, Jirapun Daengdej "Test case reduction methods by using CBR" Autonomous System Research Laboratory Faculty of Science and Technology Assumption University, Thailand