Different computational perspective of test suite minimization in software testing

Neeru Ahuja, Pradeep Kumar Bhatia

Abstract : Software has become a need in the present context of society Depending upon the requirements of user new features are being continuously added in software which may lead to increase the test suite size. The reduction of test suite size becomes necessary to get rid out of this problem. Test suite minimization has become an effective method for this problem that efficiently reduce the cost as well as computation time. In the present paper an attempt has been made to describe various such techniques that reduce the test suite size. A number of things like test adequacy criteria, various dataset used in literature and further challenges have been discussed.

Index Term : Software testing, Test Suite, Test adequacy criteria, Test suite minimization, Soft Computing, Test case

1. INTRODUCTION:

Testing of software is essential for software industry. The aim of software engineer is to develop software with least possible error and cost. Therefore, whole effort of software engineering activities is to design method and tools to eliminate error at source. Therefore, software testing is highly significant in SDLC. As per the IEEE standard software testing may be defined as "the process of analyzing a software item to detect difference between existing required condition and to evaluate the feature of software item". Testing is most expensive process and consumes as most as fifty percent of total software development cost .Now question arise what is important for minimizing cost i.e. test suite size. A collection of a number of test cases is called as test suite. As per the IEEE standard test suite may be defined as ''A set of test inputs, execution, and expected results developed for a particular objective such as to exercise a particular program path or to verify compliance with a specific requirement''. The quality of a test suite can be measured through fault coverage, code coverage, size and no. of faults that one being detected by it [17]. As the time passes new test cases increases due to a lot of changes that include system functionality, technology and business demand . Therefore, it has been the main focus of the researchers to find an effective and small set of test case which satisfy test requirements, minimize testing cost and maximize profit in order to test a software system [18]. To achieve this objective, test suite minimization techniques are being used. We can identify two main approaches which are being discussed in literature. These include structural and functional testing. The structural approach helps to study the internal structure of a program. Black box testing involves the outputs which are being produced by certain definite inputs. All the details of a program can be checked by applying the black box testing. The composition of our paper is given below : section II gives a some description on the test suite minimization techniques. Section III describes state –of- the art and Section IV through some light on the test adequacy criteria and dataset. And Section V presents some future research direction with conclusion.

- *Research Scholar ,Deptt. Of Computer Science and Engineering Guru Jambheshwar University Hisar* *Neeru902@gmail.com*
- *Professor ,Deptt. Of Computer Science and Engineering Guru Jambheshwar University Hisar* *Pkbhatia.gju@gmail.com*

II TEST SUITE MINIMIZATION TECHNIQUES

These techniques design to search a reduced test suite which still assures software quality.The reduced test suite size should be smaller than original test suite. It gives the same coverage with negligible effect on the capability of fault detection as well as test suite can significantly reduce the cost of execution. Harrold et al. defined test suite minimization as:"Given a test suite T, a set of test case requirements r1,r2,…,rn that must be satisfied to provide the desired test coverage of the program, and subsets of T,T1,…,Tn one associated with each of the ri such that any one of the test cases tj belonging to Ti can be used to test ri .Test suite minimization is to find a representative set of test cases from T that satisfies all of the ri[15]." Although, a plethora of techniques are defined for minimizing test suite. In every technique researchers choose test adequacy criteria to achieve its objective and use available resources. Initially basic techniques were classic greedy, Heuristic H, Heuristic GRE and then after a decade the literature was extensible moved to look forward for ILP and computational techniques which are explained below. Heuristic H[15]: This technique was proposed by Harrold with an objective is to find the reduce size of representative set of the test suite. Algorithm consists initialization and preprocessing of all the input sets cardinality and check maximum cardinality of input sets. First include those test cases Ti that occur in the cardinality one in input sets and marked all Tj cardinality any of these input sets. Secondly marked sets are not reprocessed and unmarked Tj of cardinality two are consider and added to representing input sets. Subsequently, all remaining unmarked Tj's containing these test cases are marked. This process repeats for Tj= 3, 4 ------,m. This recursive function applies until either a single test case is found or final random choice is made and if there is a tie among test cases when examing Tj's of size m then unmarked Tj with max cardinality m+1 is chosen and random choice is made. Heuristic GRE[19]: This algorithm was proposed by Chen and Lau. Heuristic GRE algorithm which is based on three strategies: greedy, essential, one to one reduction strategy. Out of a number of requirements some one not yet satisfied greedy strategy mainly focus on that . Selects all essential test cases is done by essential strategy and 1to 1 strategy is defined as technique of minimizing satisfiability relation S(T,R) by removing 1 to 1 redundant test cases. In this algorithm firstly select all essential test cases and then apply greedy heuristic G. 1to 1 repeatedly apply because of removing

redundant test case so that to reduce problem. The greedy strategy is applied when neither the essentials strategy nor 1-to-1 strategy is applicable. ILP[20] : Black et. al. proposed strategy in which two Binary ILP model were evaluated, one a single objective model and second one for multiobjective. In single objective test suite minimization the ILP selected test cases only.But in test suite minimization process a number of different cases arises out of which some can reveal error where as some other don't. To avoid this problem error detection has been introduced as second objective in optimized process of multiobjective model i.e. weighted- sums bi-criteria model. Soft Computing Techniques: Soft computing, opposite to traditional one acknowledges with imprecise models and gives solution to complicated real life problems. Soft computing is tolerate of unpredictability, exaggeration, imprecision, and approximation. Soft computing techniques are fuzzy logic, GA, PSO, machine learning and expert system.

3 State- of-the-Art :

To remove redundant test cases we have to choose an optimization problem whether single objective or multiobjective. Both types cover different source, type and criteria. Single objective optimization focus either on effectiveness or cost but not both. Multiobjective optimization problem focus on both effectiveness and cost associated challenges. These challenges becomes more and more complex by the participation of a number of conflicting objective and restriction defined by users for test suite reduction optimization[8]. Extension of different techniques are described in the state- of – art and table 1 describe year wise detail of literature with test adequacy criteria, dataset and result.

3.1 Extension of Greedy Algorithm

Gupta et al. [6] proposed delayed greedy algorithm in which author have reduce the table's size on the application of owner, object and attribute reduction.Then author remove intervention by selecting a test using greedy heuristic. Paras et al.[11] presented BOG algorithm which reduce test case w.r.t to branch coverage and compared result to H. But algorithm selects test suites in a particular order only. A reversal in such orders lead to the adverse results. Sugave et al.[12] proposed a newly designed algorithm (GTAP) using TAP measure and greedy search algorithm, which focused on reducing cost and in a further extended ATAP consisting of nine metrics. Liu et al.[2] proposed a techniques to select the faults that occur at the boundaries by extending one step in HGS algorithm. Lin et al.[4] used three techniques namely greedy, greedyIrreplacebility, greedyratio reduce cost, time, fault detection capability, total regression cost in case of small sized test suites. With the increases in test suite size the effectiveness of these techniques become less and less.

3.2 Extension of GA

Mala et al.[1] presented Hybrid Genetic algorithm which is combination of GA, BA and Local search. They developed a tool called Hybrid Tester. Wang et al.[7] performed an experiment with an objective to minimize test suite and increase capability of fault detection in software product line engineering using weight based GA, Random weight GA, WBGA-Multiobjective. The weighted technique may or may not applicable on every product line testing. Ali et al.[3] used a coupled multiobjective ga and greedy and found that GA is much better than greedy.

3.3 Extension of ILP

Some of researchers selected the ILP find solution. Hsu et al.[18] used ILP-based multiobjective optimization technique with the objective to make a balance between both effectiveness measure and cost. Author proposed a tool called MINTS to solve multiobjective TSR problem. This tool is freely available. Lin et al.[17] implemented tool NEMO, which is used to solve nonlinearity of problem and convert it into linear problem. They developed technique that programmatically transform nonlinear into a linear form. Multi-objective technique being more informative and user friendly is most preferred by the users. As it gives various solutions of the objective being considered and also helps in decision making. It's main limitation in comparison to single objective is its large computation time.

Table1. *Yearwise details of few techniques used in test suite minimization*

Year	Author	Technique/Tool	Test adequacy criteria	Dataset	Result
2005	Gupta et al.	Delayed Greedy algorithm	Coverage Requirement	Sir	Delayed-Greedy always not produced optimum result
2009	Hsu et al.	MINTS	Maximal coverage & Fault Detection capability	Logic blox & eclipse core	As efficient as Heuristic &maintain scalability
2010	Mala et al.	Hybrid Genetic algorithm	Mutation score, path coverage	Academic &industrial problem	HGA produce optimal solution than GA & BA Provide high fault revealing capability
2013	Wang et al.	Weight based genetic algorithm	Pairwise coverage,fault detection capability	Industrial problem, Literature Case study	WBGA find better when compared with random search algorithm
2014	Mohapatra et al.	Genetic algorithm	Coverage	Java program	50 % Reduce size than original
2015	Jeyaparkash et al.	NSGA-II	MAX branch coverage	Online ticket booking application	NSGA-II provides 15% more efficient than original test suite
2016	Srivasatav	BA with requirement based mapping approach	Code coverage, fault detection capability	binary search program	New approach gives better result than genetic and ba
2017	Indumathi et al.	Genetic Algorithm	Requirement coverage,fault detection capability		Algorithm reduce size little but not fault loss
2017	Sugave et al.	GTAP	Suite cost, suite cost reduction, improvement cost, improvement(%)	Sir	Proposed technique gives 93.07 result which is higher than other

4. Test adequacy Criteria And Datasets Used for Test suite Minimization

4.1 Test Adequacy criteria

A set of rules that is applied to the test suites for determination of treated or untreated software bugs. The black box testing is based upon requirements and models and in white box it is based on code. Researchers adopted different test adequacy criteria to achieve their objective as shown in figure 1. It can be coverage criteria, fault detection capability, fault detection effectiveness etc.

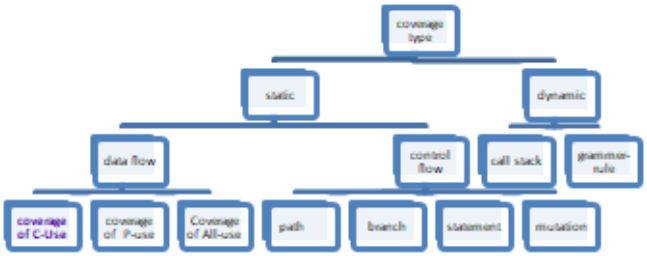

Fig. 1*. Different coverage type criteria*

Mutation score = (Dead Mutants / (totalmutants – equivalent mutants))*100 [1]
Requirement coverage = (No. fulfill requirement/ Total no.of requirement) *100
Branch coverage = (No. of decision outcome tested/ Total no of decision outcome) *100
Fault Detection Capability[9] : Measures the FDC of selected test solution for product.
SucR = NumSuc / NumSuc+NumFail
Here execution of test case success rate is Suci , number of success execution is NumSuc and number of fail execution is represented by NumFail.
Path Coverage[1]:is ratio of number of paths covered to total number of paths.

Figure 2 describe which criteria is used in how many no. of publications.

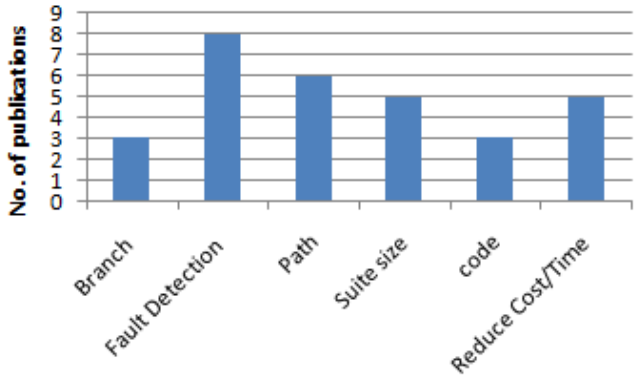

Fig. 2 *Test adequacy criteria used in publications*

4.2 Datasets

SIR1(Software-artifact Infrastructure Repository) : SIR is repository of software related artifacts such as test suites, fault data and scripts which is used for experimentation[18].The repository contains many software written in c, c++, c# and java.

TABLE 2 *Different Objects from sir repository used in literature*

SIR Repository Subject Program			
Programs	Description	LOC	Used by
Flex	Fast lexical analyzer	10459	[20,19]
Sed	Command line text editor	14427	[19]
Make	Executable builders and generator	35545	[19]
Space	Interpreter	6199	[14,12,18]
Schedule	Priority scheduler	412	[14,20,12,18]
Schdule2	Priority scheduler	374	[14,20,12]
Replace	String pattern match and replace	564	[20,12,18]
Print_token	Lexical analyzer	726	[14,20,12,18]
Print_token2	Lexical analyzer	570	[14,20,12,18]

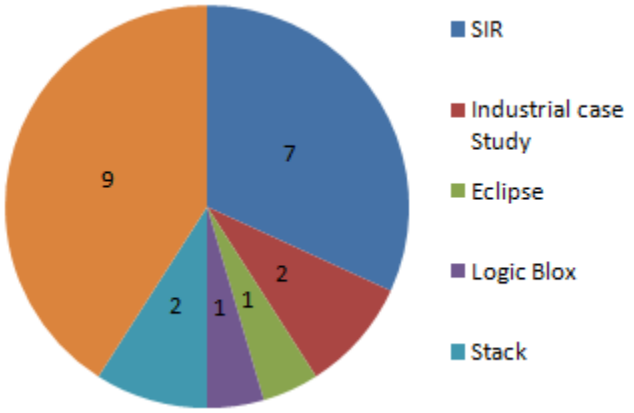

Fig. 3 *No. of publications where considered dataset is used*

Logic Blox2: is sales prediction system that contains a lot of lines nearly one million in different languages. [18]
Eclispe3 : Plug-ins are provided by eclipse that are most widely used. [18]
Others: It contain different programs or algorithm of different languages like c or java. e.g GCD no, prime number etc.

CONCLUSSION AND FUTURE CHALLENGES

In this study , we represented a survey of test suite minimization techniques and discuss trends of future research. We have discussed test adequacy criteria used to minimize test case . Test suite reduction is intrinsic for effectiveness of software so its grabbing the attention of many researchers .There are many research challenges , which are yet to be addressed in this area. some of them are briefly listed as follows:

- Need to focus on different domain like web service, model based testing, software product lines.
- For testing large system scalability is always be an issue.
- To focus on industrial data set
- A focus has to be made on multiobjective optimization problem.

REFERENCE

[1] D. J. Mala, V. Mohan, “Quality Improvement And Optimization Of Test Cases – A Hybrid Genetic Algorithm Based Approach”, ACM SIGSOFT Software Engineering Notes, vol. 35, no. 3, pp. 1-14, 2010 .
[2] P. Liu, “ An Efficient Reduction Approach To Test Suite”, IEEE, 2014.
[3] A. Yamuç, M. Özgür Cingiz, G. Biricik, O. Kalıpsız, “Solving Test Suite Reduction Problem Using Greedy and Genetic Algorithms”, International Conference – 9th Edition Electronics, Computers and Artificial Intelligence, 2017.
[4] Chu-Ti Lin, Kai-Wei Tang, Jiun-Shiang Wang, Gregory M. Kapfhammer, “Empirically Evaluating Greedy-Based Test Suite Reduction Methods at Different Levels of Test Suite Complexity” Sci. Comput. Program. (2017)
[5] S. R. Khan, S. P. Lee, R. W.Ahmad, A. Akhunzada, V. Chang, “A Survey On Test Suite Reduction Frameworks And Tools”, International Journal of Information Management ,pp. 963–975, 2016.
[6] S. Tallam, N. Gupta, “ A Concept Analysis Inspired Grredy Algorithm For Test Suite Minimization”,in ACM SIGSOFT Software Engineering Notes, 2005, pp. 35-42.
[7] S. Wang, S. Ali, A. Gotlieb, “Minimizing Test Suites In Software Product Lines Using Weight-Based Genetic Algorithms”, ACM, pp. 1493-1500,2013.
[8] S.U. Khan, S. P. Lee, N. Javaid, W. Abdul, “A Systematic Review On Test Suite Reduction: Approaches, Experiment’s Quality, Evaluation, And Guidelines ” , IEEE Access, pp. 1-26, 2018.
[9] P. R. Srivastava, “Test Case Optimization A Nature Inspired Approach Using Bacteriologic Algorithm”, Int. J. Bio-Inspired Computation, vol.8, no. 2, pp. 122-131, 2016.
[10] S.R. Sugave, S.H. Patil, B.E. Reddy, “ DIV-TBAT Algorithm For Test Suite Reduction In Software Testing” , IET Software ,pp. 271-279, 2018
[11] S. Parsa, A.Khalilian, “ A Bi-Objective Model Inspired Greedy Algorithm for Test Suite Minimization” , pp. 208-215, 2009.

[12] S.R. Sugave, S.H. Patil, B. Eswara Reddy, “A Cost-Aware Test Suite Minimization Approach Using Tap Measure And Greedy Search Algorithm”, International Journal of Intelligent Engineering and Systems, vol.10, no. 4, 2017

[13] C.P.Indumathi, S.Madhumathi, “Cost Aware Test Suite Reduction Algorithm For Regression Testing”, International Conference on Trends in Electronics and Informatics ICEI , pp. 869-874, 2017.

[14] S. Jeyaprakash, Dr. K. Alagarsamy, “A Distinctive Genetic Approach For Test- Suite Optimization”, International Conference on Soft Computing And software engineering , pp. 427-434, 2015.

[15] M.J. Harrold,R. Gupta, M.L. Soffa, “A Methodology For Controlling The Size Of A Test Suite”, ACM Transactions on Software Engineering Methodology,vol. 2, pp. 270-285, 1993.

[16] H. Zhong, L. Zhang, H. Mei, “An experimental study of four typical test suite reduction techniques”, Information and Software Technology ,vol. 50, pp. 534–546, 2008.

[17] J.W. Lin, R. Jabbarvand, J. Garcia, S.Malek, “Nemo: Multi-Criteria Test-Suite Minimization With Integer Nonlinear Programming”, Proceedings of 40th International Conference on Software Engineering , 2018.

[18] H.Y. Hsu, A. Orso, “MINTS: A General Framework and Tool for Supporting Test Suite Minimization”, IEEE ,2009 .

[19] Y.T.Chen, M. F. Lau , “A New Heuristic For Test Suite Reduction”, Information & Software Technology pp.347–354, 1998.

[20] J. Black, E. Melachrinoudis, D. Kaeli, “Bi-Criteria Models for All-Uses test suite reduction”, Proceedings of 26th International Conference on Software Engineering , 2004