Test Case Minimization Techniques : A Review

Rajvir Singh[1] and Mamta Santosh[2]
[1,2]*Department of Computer Science and Engineering, DCRUST, Murthal, Haryana(India)*

Abstract

Software testing is most expensive phase of development. It becomes unfeasible to execute all the test cases. Test case minimization techniques are used to minimize the testing cost in terms of execution time, resources etc. The purpose of test case minimization is to generate representative set from test suite that satisfy all the requirements as original test suite with minimum number of test cease. Main purpose of test case minimization techniques is to remove test cases that become redundant and obsolete over time. Several techniques have been purposed in literature. These techniques can be categorized as Heuristics, Genetic Algorithm, Integer Linear Programming based techniques. This paper presents a survey on the work that has been done in test case minimization.

Keywords: Test Case Minimization, Software Testing, Survey, Literature review, Test suite reduction techniques.

1. Introduction

Software testing is most important and expensive part of software development process. Test cases are run on the software to find errors. Test cases need to be defined along with the requirement specification. Test case is defined in IEEE standard as [6]: "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". A test suite consists of all the test cases that satisfy all the testing requirements. As software is developed, test suite grows larger. It becomes unfeasible to run all the test cases as it result in high testing cost. Test case minimization techniques are used to minimize the testing cost. Test case minimization techniques generate a representative set from the original test suite that satisfy all the requirements as original test suite but contains less number of test cases. Redundant test cases are removed from the test suite. A test case is said to be redundant if same requirements can be satisfied by other test cases.

The problem of selecting a representative set of test cases that provides the desired testing coverage of a program or part of a program is stated as follows[1]:

Given: A test suite TS, a list of testing requirements $r_1, r_2, \ldots r_n$, that must be tested to provide the desired testing coverage of the program, and a list of subsets of TS, T_1, $T_{2,\ldots}$ T_n, one associated with each of the r_i's such that any one of the test cases t_i belonging to T_i can be used to test the requirement r_i.

Problem: Find a representative set of test cases t_i that will satisfy all of the r_i's.

Various techniques have been proposed for minimizing test suites. These techniques can be categorized as:

- Heuristic based
- Genetic algorithm based approach
- Integer Linear Programming based approach
- Hybrid techniques

Heuristic based techniques include Heuristic H, GE and GRE. These strategies are based on three strategies - essential, redundant and 1-to-1 redundant test cases. Genetic algorithm based approach uses production, mutation and crossover to produce representative set. Integer Linear Programming based approach uses equation form to find minimal set.

Rest of the paper is organized as: Section 2 contains a review on existing techniques. Table *Quick Review* shows a year wise brief description of paper. Section 3 contains conclusion. Section 4 contains References.

2. A Review

Harrold et al. [1] defined heuristic H for test case minimization. Their technique identified redundant and obsolete test cases and removed them from test suite. Test cases are selected according to their degree of essentialness. A test case is essential if requirement satisfied by that test case cannot be satisfied by any other test case. Next it selects test cases which satisfy most uncovered requirements.

Chen and Lau [22] proposed heuristic GE for test

case minimization. Heuristic GE is based on the greedy strategy and essential strategy. Firstly essential strategy is applied i.e. all the essential test cases are added to representative set. Then greedy strategy is applied on remaining test cases repeatedly until the entire test requirements are satisfied. Heuristic GE performed better than heuristic G since essential test cases are selected at first stage.

Chen and Lau [2] introduced GRE approach for minimization. This approach is based on three strategies – Essential strategy, 1-to-1 redundant strategy, Greedy strategy. In this approach, firstly essential test cases are selected and added to representative set, then 1-to-1 redundant test cases are removed repeatedly and then greedy approach is applied on the remaining test cases until all the requirements are satisfied. GRE guaranteed to generate optimal representative sets.

Time complexity of GRE approach in worst case is $O(\min(m,n)(m+n^2k))$ where m and n number of elements in requirement set and test suite respectively, k is maximum number of requirements that a test case can satisfy.

Concept of GRE approach can be illustrated as (Figure 1):

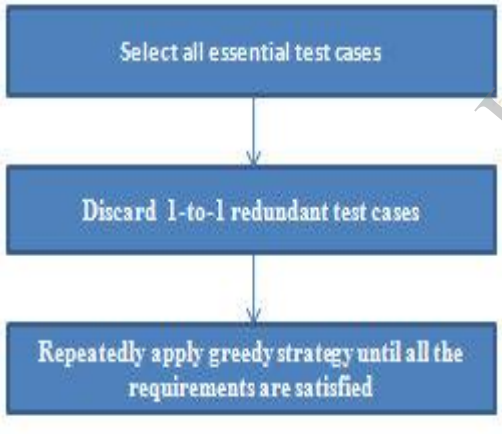

Figure 1

Chen and Lau [23] performed a simulation study on heuristics for test suite reduction. They judged the performance of heuristics by size of their representative sets. They concluded that when overlapping (< 2) among the requirements is very small then all the heuristic produce same sized representative sets. Greedy heuristic G is recommended because it has smallest worst case time complexity and requires no extra steps. When overlapping is large (> 15) then heuristic H performs better than others, heuristic G, GE, GRE has same performance. Heuristic G is preferred because of smallest worst case time complexity and no extra processing steps. When overlapping is moderate (>2 & <15), GRE performs best and heuristic G performs worst. Among heuristic GE and H, heuristic H consistently performs better than heuristic GE.

Chen and Lau [3] proposed divide-and-conquer approach towards test suite reduction. They concentrated on dividing strategies that are complete with respect to the minimal and optimal representative sets, from the perspective of essential test cases. Divide-and-conquer approach basically decompose the original problem into smaller sub problems, find optimal solutions for the sub problems, and construct a solution for the original problem from solution of the sub problems. They derived essential subset and redundant subset corresponding to essential test cases and redundant test cases respectively. An essential subset contains essential test case. A redundant subset is that whose satisfied requirements can be satisfied by other test cases. To form representative set, essential subset can be included and redundant subset can be discarded.

Tallam and Gupta [4] proposed inspired greedy algorithm for test suite reduction which is based on Formal Concept Analysis of the relation between test cases and testing requirements. Concept analysis can be used for objects with discrete properties. For minimization, test cases are considered as objects and requirements as their attributes. Relationship between object and attributes corresponds to the coverage information of test case. Context can be analyzed using a concept analysis framework. Concept analysis identifies maximum grouping of objects and attributes called contexts. Object reduction rules and attribute reduction rules are used for reducing objects and attributes. In classical greedy heuristic only object implications are used, attribute implication was not considered. In Tallam and Gupta's delayed greedy algorithm, initially context table was made then the size of context table was reduced by applying on the object reductions, attribute reduction and owner reduction. The object and attribute reductions slightly reduce the size of the context table by removing redundant objects and attributes from further consideration, the owner reduction removes redundant objects and attributes and also selects a test case which will be added to the minimized suite. At each iteration, the owner reduction selects test cases that will be included in the reduced and the requirements covered by these test cases are removed from further consideration. Interference among test cases was removed using greedy heuristic. They conducted experiments with the programs in the Siemens test suite and the space program to measure the extent of test suite size reduction and evaluated that for each test suite for each program, the size of minimized suite generated by their technique was of

the same size or of smaller size than that generated by the traditional heuristic algorithms.

Jeffrey and Gupta [5] proposed test suite reduction with selective redundancy to decrease the loss of fault detection effectiveness. They observed that removing some specific redundant test cases fault detection capability suffers significant loss. So the test cases that result loss in fault detection effectiveness should be included in the reduced set. For this, they used primary and secondary coverage criteria. First they selected test cases which satisfy primary requirements and then which satisfy secondary requirements. They modified HGS heuristic with selective redundancy in which they selectively added those test cases that provide additional def-use coverage at the time they become redundant with respect to branch coverage. To find redundant test cases they used branch coverage information and def-use information obtained by data flow analysis. The uncovered primary requirements are considered in increasing order of associated testing set cardinality. Then test case that covers the most uncovered requirements are selected whose testing sets are of the current cardinality. When a tie occurs, the preference is given to the test case that covers the most uncovered requirements whose testing sets are of higher cardinalities. If maximum cardinality is reached and there are still remaining ties, a random selection is made among the test cases that are tie. The selected test case is then added to the reduced suite. Then newly covered requirements are marked and removed from further consideration. Coverage information is updated after addition of each test case. For secondary requirements also, data structures is updated to reflect the updated secondary coverage information of reduced suite. After all the primary requirements are satisfied, more test cases are selected from redundant test cases until all the test cases are selected or the remaining test cases does not satisfy any secondary requirement. They experimented their technique on Siemens programs. Errors were injected to the programs such as changing the operator or operand in an expression, changing value of a constant, adding and removing code and changing the logical behavior of the code. For secondary requirements they used *all-uses* coverage information for each test case which is computed by the ATAC tool. Their technique produced representative sets with better fault detection effectiveness by slightly increasing size of reduced test suite.

Khan and Nadeem [6] proposed TestFilter in which they used the statement-coverage criterion for reduction of test cases. Weights were assigned to test cases. Weights referred to number of occurrences of particular test case that covers different statements. They selected non-redundant test cases based on their weights. They calculated weighted set of test cases and selected higher weighted test cases firstly. Then test cases of low weights are selected until all the requirements are satisfied. In case of tie among test cases, a random selection is made. Selected test cases are added to reduced set. They performed experimental evaluation on triangle problem to study effectiveness of Test Filter. Their technique consumed fewer resources like CPU cycles for selection of test cases and storage space. Their technique made significant reduction in the size of test suite approximately 90% and also decreased cost in terms of execution, storage and management cost.

Zhong, Zhang and Mei [20] did an experimental study on heuristic H, heuristic GRE, genetic algorithm based approach and ILP based test suite reduction techniques. They implemented four typical test suite reduction techniques on the same platform and performed an experimental comparison of them by applying them on both small and large subject programs. All the studied techniques were implemented using Microsoft Visual C++6.0. All the techniques were executed on a PC with 512 M memory Intel Pentium 2.26 GHz CPU, running the Windows 2000 Professional operating system. Eleven programs were used in the experiment among those seven were Siemens programs and four were XMLPPL, TCC, GNU tar and PdfToHtml. In their experiment statement coverage was used as test requirement. Main focus of comparison were execution time and representative set. They found that except genetic algorithm all other techniques generate smaller and almost same sized representative set. Genetic algorithm was less efficient in generating less sized representative set. ILP-based approach can always produce the smallest representative sets among other approaches. Genetic algorithm based technique performs worst in execution time. Among others, heuristic H is the fastest, while among heuristic GRE and ILP based approach, heuristic GRE is a little faster. Different reduction techniques produce different representative sets of same size. They concluded that heuristic H is best than others after that ILP based approach should be used where representative set of smallest size and high fault detection effectiveness are required.

Smith, Geiger and Soffa [7] introduced use of call trees for prioritizing and reduction. They constructed tree based model of program behavior. Using dynamic call tree, reduction component finds the subset among test cases that covers same call tree paths and prioritization is used to reorder the test cases so that all the requirements are met as early as

possible. Coverage effectiveness is used for prioritization. They implemented the call tree constructor with the Java 1.5 and AspectJ 1.5 programming languages. The tree constructor initialize the call tree prior to first test case runs, then store the tree and measure the execution time of each test. The tool builds a call tree, this call tree contains a node for every test case invocation . Each path is a unique test requirement because it represents a series of method calls that took place during testing. After creating call tree, reduction algorithm was used to generate reduced test suite that satisfies all the requirements but contains less number of test cases than the original test suite. they performed their study on a GradeBook application containing 1455 non-commented source statements (NCSS), 10 classes and 147 methods. The experiments showed that the call tree construction probes increase test suite execution time by 12.3%. When using the overlap-aware greedy algorithm and testing time by 82%. The coverage effectiveness of prioritized test suite was .96 while of original test suite was .38. Their reduced set was coverage effective but more time is consumed in constructing call trees. Their reduced set contained 45% less test cases than the original suite. They observed that prioritized suites are able to achieve coverage faster.

Chen, Zhang and Xu [8] proposed degraded ILP Approach for Test Suite Reduction. They developed the technique to bridge the gap between ILP approach and traditional heuristic approach. They produced a lower bound of minimum test suite and feasible solution near lower bound was searched. If the size of representative set equals to lower bound then representative set is best result, if size of representative set is closer to lower bound then it can be considered as good result, if size of representative set is far from lower bound then it need to use Integer Linear Programming or any other expensive methods to improve representative set. Firstly they applied 1-1 reduction on test cases and requirement to ensure that there are no 1-1 redundant test cases and no 1-1 redundant requirements. The basic idea of degraded ILP is the single-branch strategy in which only one most possible sub problem is selected for each variable. They compared their result with heuristic based approaches and found that their approach always performed better than the traditional approaches and sometimes guarantee minimum size reduced set. DILP could generate minimum test suite for all Boolean specifications. However, the complexity of their technique was higher than the heuristic approaches.

Lin and Huang [9] analyzed test suite reduction with enhanced tie-breaking technique. They used some additional coverage criteria for breaking the tie among two test cases which was different than traditional approaches where a random decision was made. They used coverage information as first and def-use pair as second criterion for breaking the tie among test cases. They chose HGS and GRE approach and developed new algorithms. In modified HGS (M-HGS) algorithm, when tie occurs between test cases then the test case which covers more secondary requirements is selected. In modified GRE (M-GRE) approach they modified 1-to-1 redundancy strategy and greedy strategy. They used Siemens suite programs and Space programs for comparing their results. They compared the results and found that M-HGS and M-GRE produce reduced set with better fault detection capability than original HGS original GRE respectively. Their technique improved the fault detection effectiveness.

Khalilian and Parsa [10] proposed Bi-criteria test suite reduction with cluster analysis of execution profiles. They combined the two general techniques called distribution-based and coverage-based techniques to construct full coverage reduced test suites with minimum overlap in the execution profiles. Coverage based techniques uses def-use pair criterion for the selection of test cases because such test cases cover execution paths which may contain faults. Distribution based techniques clusters the test cases on the basis of their execution profiles and can be described by two methods: cluster filtering and failure pursuit. In cluster filtering, cluster analysis is used to partition the test cases into clusters such that the objects with similar attributes are in same cluster. After that, test cases are sampled from each cluster. Therefore, they combined these two techniques to form full coverage reduced set and minimum overlapping between the test cases. They analyzed their technique on Siemens suite. Their technique generated reduced test suites with less fault detection capability.

Parsa, Khalilian and Y. Fazlalizadeh [11] proposed a new algorithm based upon the cluster analysis. The proposed approach combined the idea of coverage based and distribution based approaches for test suite reduction. Firstly execution profile of test cases is calculated. Then test cases are divided into clusters based on the similarity of their execution profiles. Test cases whose execution profiles are similar are placed in same cluster. Test cases in same cluster likely cover same program elements. Heuristic is applied to sample test cases from clusters to make

coverage of reduced test suite equal to that of reduced suite. In their proposed approach, heuristic method sorts the clusters in ascending order on the basis of number of test cases in each cluster. Then each cluster from first to last is considered repeatedly until all the requirements are satisfied. In each iteration, sample test case is selected from each cluster. They performed experiment on Siemens. Each program of the Siemens suite contains a single fault seeded in it. They calculated percentage of test suite size reduction and percentage of fault detection loss. For clustering of test cases based on their execution profiles they used tool Weka 3.5.8 . Clustering algorithm CLOPE was used as CLOPE is efficient and fast method for clustering large and high dimensional data. For controlling level of inter-cluster similarity, a value Repulsion is used. By varying this value Repulsion, number of clusters can be changed. They concluded that their technique produced reduced size test suite. Their reduced set was coverage adequate but less fault detection effectiveness than H algorithm.

Yoo and Harman [12] proposed multi-objective test suite reduction. They utilized a hybrid, multi-objective genetic algorithm. This algorithm combined the efficient approximation of the greedy approach with the genetic algorithm to produce high quality Pareto fronts. The main aim was to achieve multiple objectives. Objective functions are mathematical description of test criterion. For two-objective optimization computational cost and statement coverage were considered as objective, a cost cognizant version of the additional greedy algorithm was implemented. In three objective optimization past fault detection history was also considered, code coverage, fault coverage and execution time were combined by taking the weighted sum of code coverage per unit time and fault coverage per unit time using the classical weighted-sum approach. Testing decisions made by their technique were more efficient.

Nachiyappan, Vimaladevi and SelvaLakshmi [13] proposed genetic algorithm for test suite reduction. Their approach used mathematical model for test suite reduction. The model built the initial population based on test history. The fitness value of test cases was calculated based on the block based coverage value and execution time of the test cases. The test cases with optimum fitness were selected. Test cases which violate fitness constraint were rejected. Their approach reduced the test suite size with same coverage as original test suite.

Zhao and Luo [14] proposed an algorithm for Reducing Test Suites based on Interface Parameters for Black Box Testing. They used interface parameters and bipartite graph for eliminating redundant and obsolete test cases. Their technique was based upon relationship among interface parameters. Their approach can greatly reduce the size and redundancy of test suite but coverage size was same. Major limitation of this technique was that a graph may not be complete bipartite graph.

Galeebathullah and Indumathi [15] proposed set theory for test suite reduction. They used set theory and greedy algorithm to form reduced sets. Intersection function was used to identify the unique requirement that have not been satisfied. Set theory was used to find the intersection between one requirement to other requirements of branch coverage criteria for the set of test cases. Firstly, intersection among the requirements is calculated. If any intersection elements occur then the test case is added to the reduced suite. This process is repeated until all the requirements are satisfied. They experimented their approach on a small program based on the branch coverage criterion. Their approach covered all the requirements and produced test suites same similar size to traditional approaches. They compared their

Huang, Liu et al. [16] proposed improved quantum genetic algorithm for reducing test suites. In their approach chromosome is encoded with quantum bit as information bit. Improved quantum genetic algorithm (IQGA) is the combination between evolutionary algorithm and quantum computing. Improved Quantum Genetic Algorithm is based on the vector representation of the quantum. Quantum bits are encoded to represent the chromosomes and the quantum rotating gate are used to achieve the update of chromosome. Improved quantum genetic algorithm can adjust the quantum rotating gates dynamically according to the individual fitness value. It can simplify the query computation and reduce its complexity. Their technique improved test efficiency and reduced testing costs greatly. Reduced test set generated by IQGA was smaller than the traditional techniques.

Zhang, Marinov et al. [17] evaluated empirical study of JUnit test suite reduction. They implemented four test-suite reduction techniques – Greedy, H heuristic, GRE heuristic and ILP approach on Java programs with real-world JUnit test suites. Performance of traditional test-suite reduction techniques on larger programs was studied.)). They proposed that which strategy tester should use. The time complexity for the greedy technique is O(mlmin(m, n)) where m corresponds to original test suite, n corresponds to requirements, l corresponds to maximum number of requirements satisfied by a test case . The time complexity of this heuristic H is O((m

+ n)nk), here k corresponds to maximum number of test case that can satisfy one requirement. The time complexity for heuristic GRE is O((n + m2l)min(m, n)). They used 19 versions of four real-world Java programs for their study, which includes 3 versions of jtopas, 3 versions of xml-security, 5 versions of jmeter, and 8 versions of ant.jtopas1. Each version of program comes with JUnit test suites and a set of manually seeded faults. They concluded that technique H always achieves largest reduction in test-suite sizes while achieving nearly the least reduction in fault detection capability on both seeded and mutated faults. To achieve cost-effective reduction in practice, heuristic H should be used. The techniques which achieve high reduction in test suite also have high reduction of same degree in fault detection capability.

You and Lu [18] proposed Genetic Algorithm for the Time-Aware Regression Testing Reduction Problem. Time criteria were added with the genetic algorithm. Aim of time aware regression testing reduction is to minimize the total running time. Fitness function minimizes the total running time in terms of objective function. The algorithm removes all redundant test cases and also decreases total running time.

Xu, Miao and Gao [19] proposed Weighted Set Covering Techniques also called weighted greedy algorithm for test suite reduction. In first step it is determined whether any test case which can satisfy all the requirements is present. If so, we select that test case otherwise repeatedly eliminate 1-to-1 redundant test cases and update test suite and remaining uncovered requirements. Essential test cases are selected and added to reduced set. For remaining uncovered requirements, priorities are assigned to test cases and sorted. Test cases are selected in decreasing order of priority until all the requirements are satisfied. The optimized test suite had a higher efficiency. They experimented on the test suite of Student Achievement Retrieval Navigation Model. The algorithm produced minimum size test suites and minimum cost test.

Although so many techniques have been addressed in literature, Still it is hard to tell which one performs best among these.

- Greedy algorithm based approach provide significant reduction in test suite but need to be optimized in large scale test suites.

- Genetic algorithm based approach need to further investigate the fault detection capability of block based test suite on software and coverage or some other criteria may also be included.
- Integer Linear Programming based approach can always produce the smallest representative sets among other approaches but cost and increased complexity need further discussion .
- Hybrid techniques combine two or more techniques into single for significant reduction in test suites and multi-objective optimization but provide high complexity. More number of techniques can be incorporated with existing hybrid techniques.

Other techniques include call tree and clustering based techniques. Call tree based techniques generate optimal set but high running time makes them insignificant to some extent. Clustering technique selects test cases on based of coverage and distribution based techniques. They produce smaller representative sets but less fault detection ability.

Table 1: Quick Review

Sr. No.	Year	Author	Title of Paper	Technique	Conclusion
[1]	1990	Mary Jean Harrold, Rajiv Gupta, Mary Lou Soffa	A methodology for controlling the size of test suite	Heuristic H	Produced smaller size reduced set
[2]	1998	T.Y. Chen, M.F. Lau	A new heuristic for test suite reduction	Heuristic GRE	Produced optimal representative set
[3]	2002	T.Y. Chen , M.F. Lau	On the divide-and-conquer approach towards test suite reduction	Dividing strategies for computing the minimal and optimal representative sets.	Dividing strategies related to the divide- and-conquer approach toward TSR

					studied
[4]	2005	Sriraman Tallam, Neelam Gupta	A Concept Analysis Inspired Greedy Algorithm for Test Suite minimization	Concept Analysis of relation between test cases and requirements	reduced sets were either same or less in size than greedy approach
[5]	2005	Dennis Jeffrey, Neelam Gupta	Test Suite Reduction with Selective Redundancy	branch coverage and all-uses coverage obtained by data-flow analysis	Larger test suites but better fault detection capability
[6]	2006	Saif-ur-Rehman Khan, Aamer Nadeem	TestFilter: A Statement-Coverage Based Test Case Reduction Technique	Statement coverage as weight	can find redundant test cases and reduced cost
[7]	2007	Adam Smith, Joshua Geiger, Mary Lou Soffa	Test Suite Reduction and Prioritization with Call Trees	Dynamic call trees for reducing and prioritizing test cases	constructing call trees increase 13% testing time
[8]	2008	Zhenyu Chen, Xiaofang Zhang and Baowen Xu	A Degraded ILP Approach for Test Suite Reduction	Searches a feasible solution close to the produced lower bound.	problem can be solved in polynomial time but more complex
[9]	2009	Jun-Wei Lin, Chin-Yu Huang	Analysis of test suite reduction with enhanced tie-breaking techniques	Additional coverage criteria for breaking tie among test cases	Improved fault detection effectiveness
[10]	2009	Alireza Khalilian and Saeed Parsa	Bi-criteria Test Suite reduction by Cluster Analysis of Execution Profiles	Combination of distribution-based and coverage-based techniques	Reduced test suites with less fault detection loss
[11]	2009	S. Parsa, A. Khalilian and Y. Fazlalizadeh	A New Algorithm to Test Suite Reduction Based on Cluster Analysis	Clusters test cases based on the similarity of their execution profiles	reduced suite is coverage adequate
[12]	2009	Shin Yoo , Mark Harman	Using hybrid algorithm for Pareto efficient multi-objective test suite minimization	Hybrid, multi objective genetic algorithm with greedy approach	More efficient testing decisions
[13]	2010	S.Nachiyappan ,A.Vimaladevi , C.B.SelvaLakshmi	An Evolutionary Algorithm for Regression Test Suite Reduction	Genetic algorithm	Produced optimal sized test-suite taking execution time and coverage factors into account
[14]	2010	Liang Zhao, Wenbin Luo	An Algorithm for Reducing Test Suite Based on Interface Parameters	Used interface parameters and bipartite graph for removing redundant tc	greatly reduced the size and redundancy of test suite

[15]	2010	B.Galeebathullah, C.P.Indumathi	A Novel Approach for Controlling a Size of a Test Suite with Simple Technique	Set theory, Greedy algorithm	All requirements are covered, reduced set same as greedy and HGS
[16]	2010	Yi-kun ZHANG, Ji - ceng LIU, Ying-an CUI, Xin-hong EI, Ming-hui ZHANG	An Improved Quantum Genetic Algorithm for Test Suite Reduction	Chromosome is encoded with quantum bit as information bit	Can reduce testing costs greatly and improve test efficiency
[17]	2011	Lingming Zhang, Darko Marinov, Lu Zhang, Sarfraz Khurshid	An Empirical Study of JUnit Test-Suite Reduction	Performance of traditional test-suite reduction techniques on larger programs were studied	To achieve cost-effective reduction in practice, heuristic H should be used
[18]	2012	Liang You Yansheng Lu	A Genetic Algorithm For The Time-Aware Regression Testing Reduction Problem	Genetic algorithm with time constraints	Reduced test suite and minimum running time
[19]	2012	Shengwei Xu, Huaikou Miao, Honghao Gao	Test Suite Reduction Using Weighted Set Covering Techniques	Weighted Set Covering to adopt the heuristic method to eliminate redundancy, and determined priority of test cases to lower cost	Reduced test suite size and reduced cost

3. Conclusion

This paper outlined the brief summary of techniques that has been proposed in literature for test case minimization. The techniques studied include Heuristic H, GRE, and Divide and conquer approach, Genetic algorithm, selective redundancy, TestFilter, Integer Linear Programming based DILP, Cluster analysis, set theory etc. Almost among these produced reduced test suites. Each technique is superior to another in some aspect. Many of them generated significant reduction in test suite, but it is harder to tell which one performs best. Heuristic based approach produced significant reduction but less fault detection effectiveness. ILP based approach guaranteed minimal set but more complex and increased cost. For a technique to be efficient it should be good in both - reduced test suite size and improved fault detection efficiency.

4. References

[1] Mary Jean Harrold, Rajiv Gupta, Mary Lou Soffa, A methodology for controlling the size of test suite, CH2921-5/90/0000/0302$01 1990 IEEE
[2] T.Y. Chen, M.F. Lau, A new heuristic for test suite reduction," Information and Software Technology ",40, (1998), 347-354
[3] T.Y. Chen , M.F. Lau ,On the divide-and-conquer approach towards test suite reduction," Information Sciences ",152, (2003), 89–119
[4] Sriraman Tallam, Neelam Gupta, A Concept Analysis Inspired Greedy Algorithm for Test Suite minimization, 2005 ACM 1595932399/05/0009
[5] Dennis Jeffrey, Neelam Gupta, Test Suite Reduction with Selective Redundancy, Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), 1063-6773/05 $20.00 © 2005 IEEE
[6] Saif-ur-Rehman Khan, Aamer Nadeem, TestFilter: A Statement-Coverage Based Test Case Reduction Technique, 1-4244-0794-X/06/$20.00 ©2006 IEEE

[7] Adam Smith, Joshua Geiger, Mary Lou Soffa, Test Suite Reduction and Prioritization with Call Trees, ACM 978-1-59593-882-4/07/0011.
[8]Zhenyu Chen, Xiaofang Zhang and Baowen Xu, A, Degraded ILP Approach for Test Suite Reduction, National Natural Science Foundationof China (60425206, 60773104, 60403016, 60633010)
[9] Jun-Wei Lin, Chin-Yu Huang, Analysis of test suite reduction with enhanced tie-breaking techniques, Information and Software Technology", 0950-5849/$, 2008 Elsevier
[10] Alireza Khalilian and Saeed Parsa, Bi-criteria Test Suite reduction by Cluster Analysis of Execution Profiles", LNCS 7054, pp. 243–256, 2012. c IFIP International Federation for Information Processing 2012
[11] S. Parsa, A. Khalilian and Y. Fazlalizadeh, A New Algorithm to Test Suite Reduction Based on Cluster Analysis, 978-1-4244-4520-2/09/$25.00 ©2009 IEEE
[12] Shin Yoo , Mark Harman, Using hybrid algorithm for Pareto efficient multi-objective test suite minimization, The Journal of Systems and Software 83 (2010) 689–701
[13] S.Nachiyappan,A.Vimaladevi, C.B.Selva Lakshmi, An Evolutionary Algorithm for Regression Test Suite Reduction, Proceedings of the International Conference on Communication and Computational Intelligence – 2010, pp.503-508
[14] Liang Zhao, Wenbin Luo, An Algorithm for Reducing Test Suite Based on Interface Parameters, 978-1-4244-5392-4/10/$26.00 ©2010 IEEE
[15] B.Galeebathullah, C.P.Indumathi, A Novel Approach for Controlling a Size of a Test Suite with Simple Technique, (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 03, 2010, 614-618
[16] Yi-kun ZHANG, Ji -ceng LIU, Ying-an CUI, Xin-hong EI, Ming-hui ZHANG, An Improved Quantum Genetic Algorithm for Test Suite Reduction, 978-1-4244-8728-8/11/$26.00 ©2011 IEEE
[17] Lingming Zhang, Darko Marinov, Lu Zhang, Sarfraz Khurshid, An Empirical Study of JUnit Test-Suite Reduction, 2011 22nd IEEE International Symposium on Software Reliability Engineering, © 2011 IEEE
[18] Liang You Yansheng Lu, A Genetic Algorithm For The Time-Aware Regression Testing Reduction Problem, 2012 8th International Conference on Natural Computation (ICNC 2012), ©2012 IEEE
[19] Shengwei Xu, Huaikou Miao, Honghao Gao,Test Suite Reduction Using Weighted Set Covering Techniques, 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, © 2012 IEEE
[20] Hao Zhong, Lu Zhang , Hong Mei, An experimental study of four typical test suite reduction techniques, Information and Software Technology 50 (2008) 534–546.