Journal of Theoretical and Applied Information Technology
10th November 2014. Vol. 69 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MULTI OBJECTIVE TEST CASE MINIMIZATION COLLABORATED WITH CLUSTERING AND MINIMAL HITTING SET

[1]R.BEENA, [2]Dr.S.SARALA

[1]Research Scholar, Dept. of Information Technology, Bharathiar University, Coimbatore.

[2]Assistant Professor, Dept. of Information Technology, Bharathiar University, Coimbatore

E-mail: [1]beenamridula@yahoo.co.in, [2]sriohmau@yahoo.co.in

ABSTRACT

Software testing aspires to explore and validate the attribute and potentiality of a program to authenticate and cross-verify the requisite results obtained. The broader bifurcation of testing is Precision Testing and Reliability Testing. Regression testing is part of reliability testing as it testifies the changes or modifications pursued to the software have not curtailed the functionality of the software by introducing any bugs. It is a kind of quality assurance to the modifications carried out. The pivotal role of regression testing is comprehended whenever modification towards development of software takes place. Re-execution of large test suites is perhaps an enigma many a times due to the paucity of resources. Here arises the need for a novel technique to minimize the test suite in order to remove the redundant test cases. With this focus to provide an innovate and time-effective strategy to remove the redundant test cases, this paper presents a multi-objective test suite minimization by considering maximum statement coverage and minimum execution time. This article also concentrates on incorporating a multi objective minimization technique using clustering approach and minimal hitting set. Here, the identification of appropriate clusters is achieved, through the weighted distance function for mixed variable type and the minimal hitting set is obtained using HS_DAG (Hitting Set Directed Acyclic Graph) algorithm. The results of this experiment exhibit that the algorithm proposed works with adequate efficacy in minimizing the test cases.

Keywords: *Regression Testing, Test Case Minimization, Similarity, Minimal Hitting Set, Clustering, HS_DAG algorithm*

1. INTRODUCTION

Software Testing has its indispensable role to assure the quality of any software developed. The "process of analysing a software item to detect the differences between existing and required conditions (that is defects/ errors/ bugs) and to evaluate the features of the software item" is what Software Testing means with reference to ANSI/IEEE 1059 standards. Testing becomes a mandatory process in the cycle of software development. It is pursed usually for three significant reasons namely, Quality Improvement, Verification and Validation, besides, Estimation of Reliability. The noteworthy fact, here is the utility of the same techniques developed two/ three decades ago in software testing. Software reliability has prime relationship with several aspects of software that includes the structure in addition to the quantity of testing it has been subjected to. The strategies practiced in software testing assure the correctness or precision of the software; testify the performance of the software; confirm the reliability of the software; besides, reinstating the security parameter of the software.

The significant maintenance of Software Development Life Cycle, above all, confides in the perusal of Regression Testing. Here, the re-execution of test cases from the existing test suites to assure that the modifications done to the existing software have no adverse effects [1] becomes mandatory to assure the quality of the software. Perhaps, the ideal regression testing is performed with a perspective to rerun all the test cases. However, the time and cost constraints permit the rerun based on regression testing techniques exclusively for the subset of test cases. It is noteworthy, that the types of regression testing include test case minimization, test case selection and test case prioritization [2]. While the test case minimization technique [3] serves to eliminate the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

redundant test cases, the test case selection techniques [4] work towards the reduction of the size of a test suite. On the other facet, test case prioritization techniques [5] focus towards the ordering of test cases besides the prior detection of faults, if any.

Generally, Regression Test suites progress over time in connection to the increment of new test cases to supplement additional functionalities or to set-right the defects explored. At the same time the hefty size of the test suite may become a threat because they turn to be an unmanageable predicament which may involve a big budget to curtail them. This problem perhaps could turn more vulnerable, if the test suite is based on either complicated machinery or manual effort.

Following Rothermel et al. [6], the test case minimization is defined as follows:

Given: A test suite, T, a set of test requirements $\{r_1,r_2,\ldots,r_n\}$, that must be satisfied to provide the desired adequate testing of the program, and subsets of T, $T_1,T_2,\ldots,T_n$, one associated with each of the r_is such that any one of the test cases t_j belonging to T_i can be used to achieve requirement r_i.

Problem: Find a representative set, T', of test cases from T that satisfies all r_is.

The testing criterion is satisfied when every test requirement in $\{r_1,r_2,\ldots,r_n\}$is satisfied. A test requirement, r_i, is satisfied by any test case, t_j, that belongs to the T_i, a subset of T. Therefore, the representative set of test cases is the hitting set of the T_is. Furthermore, in order to maximize the effect of minimization, T' should be the minimal hitting set of the T_is. The minimal hitting set problem is an NP-complete problem as is the dual problem of the minimal set cover problem.

Existing approaches to regression test suite minimization have been single-objective approaches that have sought to optimize a single-objective function. This paper presents a multi objective formulation for Test Suite Minimization problem.

The multi-objective test suite minimization problem [7] is to select a subset of test suite, based on multiple objective functions. That is, given a test suite S, a vector of M objective functions, the problem is to find a subset S' of S such that S' is a Pareto optimal set with respect to M. The proposed work is based on multi-objective test suite minimization by considering statement coverage and execution time. The aim is to achieve maximum statement coverage in minimum execution time. Therefore, the problem can be stated as to find a subset of the test suite S with statement coverage SC and execution time ET such that no other subset of S can achieve more statement coverage SC without spending more time than ET.

The need for a novel strategy which could be comfortable as well as cost effective to run the test cases form the vital foundation for the objectives of this paper stated subsequently.

- A multi-dimensional test suite minimization with a special consideration to the statement coverage and execution time act as the twin objectives of this article.
- Clustering technique is practised for the test case minimization and elimination of redundancies between test cases in this paper. In addition, weighted distance function for clustering using mixed variable type is elucidated with adequate explanation.
- The application of a minimal hitting set algorithm HS_DAG is shown effective in exploring and minimizing the test suite substantiated through the results obtained.

To state the precise sketch of this paper, Section 2, outlines the works related to the conceptualization of this paper, Section 3 describes the multi-objective test suite minimization problem, Section 4 projects the experiments combined with results and Section 5, gives a final notation of the entire research.

2. RELATED WORK

This section of the paper, enunciates the list of publications that bear relevant and specific information in connection to the research pursued on multi objective test case minimization strategies to enhance statement coverage within an effective time span. Even though there appears a wide range of researches performed in the same territory only quite a few closely revolve around the crux of this research. The subsequent list evidently proclaims such publications with some relevance to the novel strategy proposed in this paper to devise a multi

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

objective test case minimization strategy with the help of HS_DAG algorithm.

An innovative greedy algorithm titled, Delayed-Greedy algorithm is deployed in [8] with a view to select a minimal cardinality subset of a test suite that covers all the requirements covered by the test suite. This algorithm is developed as a result of a close observation of the Test suites reused and updated recurrently corresponding to the evolution of the software. Perhaps, these test cases in the test suite are expected turn redundant in connection to the modifications of the software over a period of time. An obvious comprehension of the resource and time constraints to re-execute the large test suites, led to the devising of Delayed-Greedy algorithm.

The exploration of redundant test cases from a test suite based on some criterion is pursued in [9]. Further, this knowledge opens an avenue for a novel test suite minimization technique that identifies redundancy in a given test suite based on multiple coverage criteria for example function, function call stack, line and branch coverage of given test cases. The noteworthy observation here is the specific criterion that functions as the base for the reduction techniques.

Several strategies for selecting a smaller number of test cases by reordering the test tests are elucidated in [10]. Besides, there is an illustration on the technique using a proof-of-concept pursued using mutation testing, achieving approximately a 33% reduction in size, and a corresponding reduction in the cost of regression testing, with a cost of only one extra run of the test case set. A suggestion for the extensive application of this test strategy is advised to measure the efficacy of test cases through data flow testing and branch testing and a trial with statement coverage to gain positive results.

The prime aspiration of [11] is test-suite composition and test-suite reduction. Here, an experiment on impact of the test-suite reduction on the effectiveness of fault-localization techniques is dealt in detail. Through the application of various test suite minimisation techniques to a set of programs, the impact of the size reduction on the effectiveness of coverage-based fault localisation techniques are strategically assessed.

While the effectiveness in the execution of certain coverage criteria in test suite reduction is based dual strategies namely, Percentage Size Reduction and Percentage Fault Detection Reduction the empirical evaluation of [12] compares five different criteria for the minimization of test suites for GUI intensive applications: event coverage, event interaction coverage, function coverage, statement coverage and call-stack coverage. The results obtained in [12] indicate the minimal probability of fault detection that aid in the observation of coverage criteria that take a lead role in test suite reduction.

In [13] a model-based regression test suite(RTS) reduction method based on Extended Finite State Machine (EFSM) dependence analysis is proposed. This method reduces the size of the RTS by examining various interaction patterns covered by each test case in the given RTS. It is stated that the automatic identification of the original model and the modified model as a set of elementary model modification.

A 50x faster method called in providing comparable results designed from the combined static slicing and delta debugging to automatically minimize the sequence of failure-inducing method calls is presented by [14]. Perhaps, 11x faster result produced in connection to the combination of slicing and delta-debugging are exhibited

Through the application of either a dynamic call tree or a calling context tree, the test reduction component fixes the subsets of the original tests that cover the paths of the same call tree. In [15], tool is proposed that constructs tree-based models of a program's behaviour during testing and employs these trees while reordering and reducing a test suite.

[16] explores a cluster-based test case prioritization technique, by clustering test cases based on their dynamic runtime behaviour. This application is evaluated on seven test suites ranging in size from154 to 1061 test cases. Besides, the paper demonstrates that clustering can out-perform un-clustered coverage based technologies and discusses an automated process that can be used to determine the yield of this research.

In [17], a branch-and-reduce algorithm is used to solve the Minimum Hitting Set Problem and use a recently developed technique called measure and conquer to perform analysis on the algorithm. Through the application of this strategy besides, quasi-convex programming at the point of

Journal of Theoretical and Applied Information Technology
10th November 2014. Vol. 69 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

optimizing the analysis results, minimum hitting set problem is put under trial for better results.

A branch-and-reduce algorithm to solve the Minimum Hitting Set Problem is proposed in [18] besides it application in recently developed technique called measure and conquer to perform analysis on the algorithm. By applying such technique and quasi-convex programming when optimizing the analysis results, it is proved that the algorithm can solve the Minimum Hitting Set Problem in O(1.23801n) and polynomial space. The concept of minimal diagnoses was originally proposed in [18] and [19] for systems where each component has only two possible behavioural modes, i.e., a normal fault-free mode and a faulty mode.

In [20] an algorithm titled, STACCATO, devised to generalize the minimal hitting-set is presented. This algorithm is exhibited as a potential one to diagnose the behavioural modes and consecutively compute a logical formula that characterizes all diagnoses. Staccato uses a heuristic function, borrowed from a lightweight, statistics-based software fault localization approach, to guide the MHS search.

3. MULTI OBJECTIVE TEST CASE MINIMIZATION

Test suite minimization appeals to be a mandatory requisite in connection to the growth of the regression test-suite of an existing software system to an extreme level where there may not be any possibility for the execution of the entire test-suite [6]. To reduce the size of a test suite the redundancy of test cases in the test suite has to be curtailed. The prime focus of this paper is to exhibit a novel approach to the practice of minimization techniques in general. The collaborated multi-objective test case minimization pursued through the clustering approach and minimal hitting set that utilizes the history of test cases in proportion to the execution time is advised here. The multi-objective test case minimization is pursued with the exploration of appropriate clusters where the weighted distance function for mixed variable type is employed and the minimal hitting set is obtained using HS_DAG algorithm.

The algorithm of HS-tree functionality to compute the minimal hitting set with the embedding of Directed Acyclic Graph, DAG is termed as HS_DAG algorithm. To simplify the description when there is a collection of ordered sets HS_DAG algorithm determines the strategic selection of a specific choice from the collection rather than going for a random or an arbitrary choice. Actually, this prominent perspective has helped in the deployment of HS_DAG algorithm aspiring more reliability and accuracy in the choice of test-suite reduction.

To state precisely, the proposed work minimizes the number of test cases in the test-suite in association with the subsequent objectives

Objective 1: Maximized coverage of statements.
Objective 2: Minimized span (time) of execution.

The example given in Figure 1 represents a Test Suite that comprises ten test cases and their statement coverage. Here, T1 to T10 represent the individual test cases and S1 to S10 exhibit the statements covered.

T1 → {S1, S3, S8, S9}
T2 → {S2, S3, S5, S6, S7, S10}
T3 → {S1, S4, S6, S7, S8}
T4 → {S3, S5, S6, S9, S10}
T5 → {S1, S2, S4, S5, S7, S9, S10}
T6 → {S1, S4, S6, S8, S10}
T7 → {S2, S3, S4, S5, S7, S8, S10}
T8 → {S1, S4, S6, S7, S9}
T9 → {S2, S4, S8, S9}
T10 → {S1, S2, S5, S6, S7, S10}

Figure 1: Test Cases with Statement Coverage

3.1 The Architecture Proposed for Test Case Minimization

The detailed plan of the strategic minimization pursued in this research is depicted in Figure 2. This empirical research is planned in four segments. The first segment focuses on the data matrix form of test cases with statement coverage and execution time, the second segment concentrates on the similarity between the test cases found based on the statements covered and its execution time using the weighted distance function for the mixed type variable. At the same time in the

Journal of Theoretical and Applied Information Technology
10th November 2014. Vol. 69 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

third segment, similar test cases are clustered using a threshold θ (for example 0.6). Normally clustering is carried out in two steps namely, making the distance matrix and applying the clustering algorithm. Finally, in the fourth segment, the exploration of a minimal hitting set for the cluster of test cases obtained from phase 3 with the support of HS_DAG algorithm is pursued.

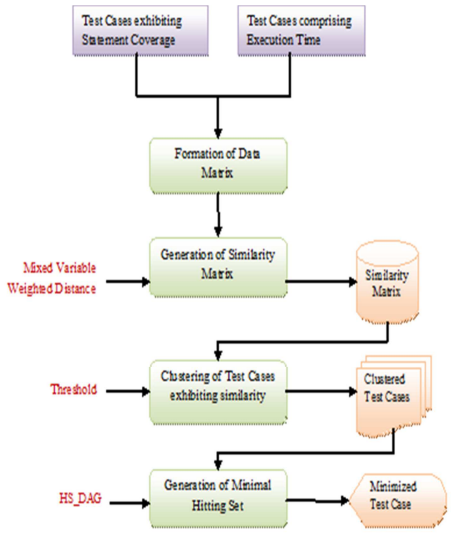

Figure 2: Architecture for Test Case Minimization

3.2 Algorithm for Test Case Minimization

Figure 3 represents the algorithm deployed in this study to minimize the Test Cases. In this tetra-faceted study, the initial facet observes the formation of data matrix of the test cases with statement coverage and execution time. The consecutive facets identify the similar test cases with reference to the statement coverage besides restricting the execution time with the help of weighted distance function for the mixed type variable. Moreover, the clustering of similar test cases using a threshold θ (for example 0.6) is also given equal importance. Noticeably, the twin Clustering steps involve the making of distance matrix and applying the clustering algorithm. In the final facet of the study, the minimal hitting set for the clustered test cases obtained from the previous facet using the HS_DAG algorithm.

Input : Test Suite comprising SC, ET
Output : Minimized Test Suite
begin
  Step 1 : Form a data matrix of test cases considering SC and ET
  Step 2 : Generate similarity matrix between the test cases.
  Step 3 : Cluster similar test cases.
  Step 4 : Find the minimal hitting set.
end

Figure 3: Algorithm REG_MIN

3.3 Data Matrix Representation

In connection to the 10 test cases ranging from T1 to T10 and the statement coverage S1 to S10, in proportion to the execution time of each test case is presented as data matrix which is shown in Table 1. The coverage of a statement by a test case is represented as 1 otherwise it is marked as 0.

Table 1: Test Cases exhibiting Statement Coverage and Execution Time

Test Case/ Faults	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	Execution Time ET (ms)
T1	1	0	1	0	0	0	0	1	1	0	0.75
T2	0	1	1	0	1	1	1	0	0	1	0.81
T3	1	0	0	1	0	1	1	1	0	0	0.89
T4	0	0	1	0	1	1	0	0	1	1	0.49
T5	1	1	0	1	1	0	1	0	1	1	0.97
T6	1	0	0	1	0	1	0	1	0	1	0.91
T7	0	1	1	1	1	0	1	1	0	1	0.85
T8	1	0	0	1	0	1	1	0	1	0	0.64
T9	0	1	0	1	0	0	0	1	1	0	0.72
T10	1	1	0	0	1	1	1	0	0	1	0.34

Moreover, the graphical representation of this data matrix is explicitly presented in Figure 4.

Journal of Theoretical and Applied Information Technology
10th November 2014. Vol. 69 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 4: Statement Coverage and Execution Time Graph of the Test Cases

3.4 Generation of Distance Matrix

The generation of distance matrix from the data procured is pursued with the help of the SC and ET. The statements covered by the test cases are represented as a asymmetric binary digit. A 1 in the cell represents that the test case covers the corresponding statement, otherwise 0. The execution time for each test case is represented as a discrete value. Therefore, a weighted distance function for mixed variable type is required. The weighted distance function is given in Equation 1.

$$d(T_i, T_j) = \frac{\sum_{f=1}^{p} \delta_{ij}(f)\ d_{ij}(f)}{\sum_{f=1}^{p} \delta_{ij}(f)} \quad \text{--------} \quad (1)$$

The Weight $\delta_{ij}(f)$ and distance $d_{ij}(f)$ is computed depending on the value of f. For asymmetric binary value f, the weight $\delta_{ij}(f) = 0$,if x_{if} or x_{jf} is missing or $x_{if} = x_{jf} = 0$, Otherwise, Weight $\delta_{ij}(f) = 1$.

If f is asymmetric binary, the distance $d_{ij}(f) = 0$ if $x_{if} = x_{jf}$, or $d_{ij}(f) = 1$. If f is a discrete value, then the distance function is calculated using Equation 2.

$$d_{ij}(f) = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}} \quad \text{--------} \quad (2)$$

For example consider the test cases T1 and T2 in Table1.

Here, i = T1 and j = T2

$\delta_{ij}(S1) = 1$, $\delta_{ij}(S2) = 1$, $\delta_{ij}(S3) = 1$, $\delta_{ij}(S4) = 0$, $\delta_{ij}(S5) = 1$, $\delta_{ij}(S6) = 1$, $\delta_{ij}(S7) = 1$, $\delta_{ij}(S8) = 1$, $\delta_{ij}(S9) = 1$, $\delta_{ij}(S10) = 1$, $\delta_{ij}(ET) = 1$

$d_{ij}(S1) = 1$, $d_{ij}(S2) = 1$, $d_{ij}(S3) = 0$, $d_{ij}(S4) = 0$, $d_{ij}(S5) = 1$, $d_{ij}(S6) = 1$, $d_{ij}(S7) = 1$, $d_{ij}(S8) = 1$, $d_{ij}(S9) = 1$, $d_{ij}(S10) = 1$

$$d_{ij}(ET) = \frac{|\,0.75 - 0.81\,|}{(0.97 - 0.34)} = 0.095238$$

From Equation 1, the value of d(T1,T2) is obtained as 0.809524. Similarly the distance between all the test cases are calculated and listed in Table 2.

3.5 The Clustering of Test Cases

The generation of distance matrix in fact authenticates the clustering process of the test cases. The clustering of test cases is made easy with the obtaining of the threshold value of the distance measured between the test cases. For instance, if the distance between the test cases is identified less than the threshold value of 0.6, then the corresponding test cases are clustered. Generally, the distance between the destination test case and other test cases are compared for a threshold value. If the distance is less than the threshold, then that test case is also joined in that cluster. For example, consider the test case T2, the distance between T2 and T4 is 0.438492 which satisfies the threshold. Next consider the test case t4 which is now compared with other test cases. The distance between T4 and T10 satisfies the threshold; hence T2, T4 and T10 are clustered. Figure 5 exhibits the test cases that are clustered with T2.

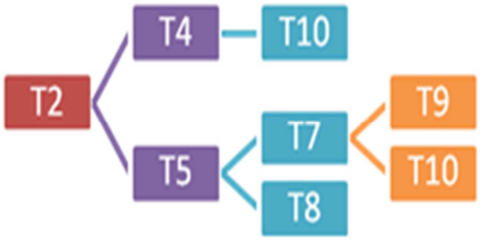

Figure 5: Clustering Procedure of Test Cases

Journal of Theoretical and Applied Information Technology
10th November 2014. Vol. 69 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The various clusters for a threshold value of 0.6 generated from Table 2 are given in Figure 6. A total of seven clusters were formed such as {T1, T6, T8}, {T1, T9}, {T2, T4, T10}, {T2, T5, T7, T9}, {T2, T5, T7, T10}, {T2, T5, T8}, {T3, T8}.

Figure 6: Clustered Test Cases

Table 2: Distance between the Test Cases

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
T1	0	0.809524	0.652778	0.676587	0.734921	0.531746	0.615873	0.646825	0.578231	0.865079
T2	0.809524	0	0.712698	0.438492	0.525397	0.715873	0.340388	0.626984	0.814286	0.305115
T3	0.652778	0.712698	0	0.863492	0.612698	0.171958	0.606349	0.342404	0.65873	0.652557
T4	0.676587	0.438492	0.863492	0	0.67619	0.740741	0.657143	0.693122	0.818342	0.582011
T5	0.734921	0.525397	0.612698	0.67619	0	0.609524	0.419048	0.502646	0.599647	0.444444
T6	0.531746	0.715873	0.171958	0.740741	0.609524	0	0.609524	0.553571	0.662698	0.656085
T7	0.615873	0.340388	0.606349	0.657143	0.419048	0.609524	0	0.757576	0.578483	0.580952
T8	0.646825	0.626984	0.342404	0.693122	0.502646	0.553571	0.757576	0	0.640873	0.608466
T9	0.578231	0.814286	0.65873	0.818342	0.599647	0.662698	0.578483	0.640873	0	0.860317
T10	0.865079	0.305115	0.652557	0.582011	0.444444	0.656085	0.580952	0.608466	0.860317	0

Journal of Theoretical and Applied Information Technology
10th November 2014. Vol. 69 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

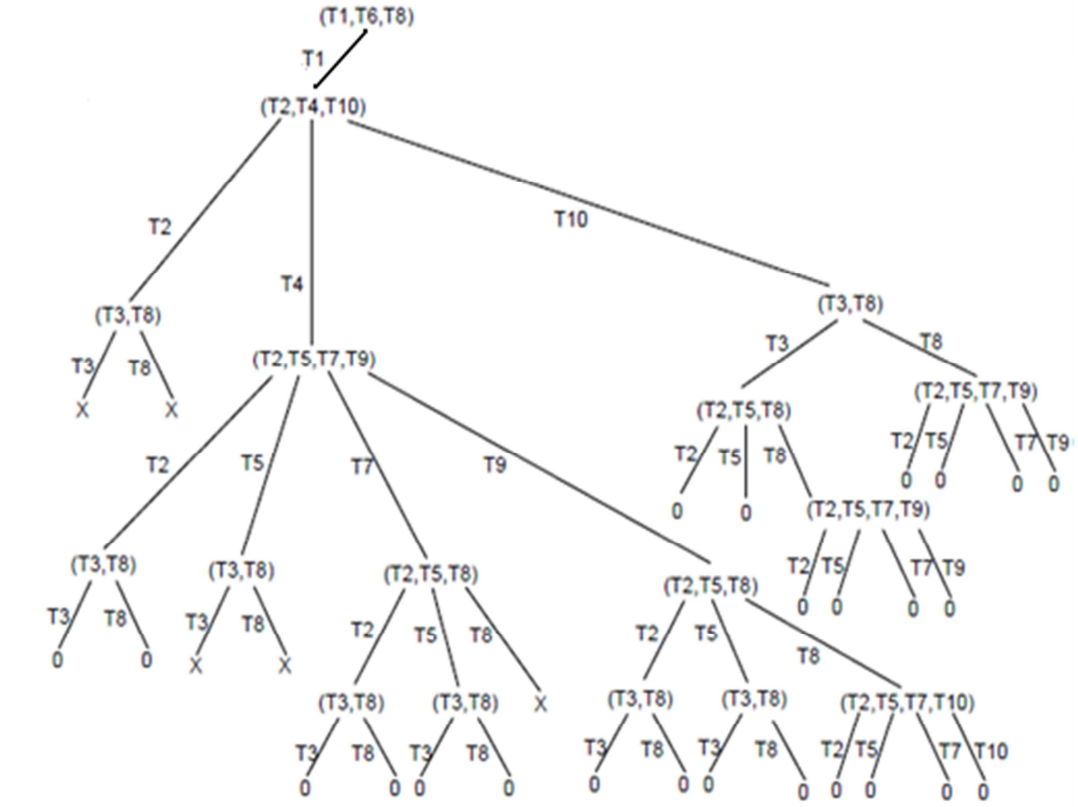

Figure 7: HS_DAG Tree

3.6 Generation of Minimal Hitting Set

Let C = (c1,c2. . . ,cn) be a collection of sets. The set H is a hitting set of C if the intersection of H and any set in C is non-empty. A hitting set is minimal if the removal of any element will destroy the hitting set. A hitting set is said to be minimum if it has the smallest size over all hitting sets.
Reiter's work [12] is taken into account for proper definitions and algorithm. An improved version of Reiter's algorithm is offered by Greiner et al. [4]. Hitting sets are defined over a set of sets with the property that the intersection of a hitting set with every given set is not empty.

Reiter [12] introduced an algorithm for computing hitting sets that has been improved later by Greiner et al. [4]. The algorithm uses the given sets c1,c2,…cn and constructs a directed acyclic graph (DAG) in a breadth first manner. After the construction of the DAG the minimal hitting sets correspond to some vertices of the DAG which are labelled with a X and other vertices are labelled with O. The algorithm needs not to compute all possible hitting sets. Instead the user can specify the maximum cardinality of the obtained hitting sets. For practical applications especially in cases where the size of the input is large, such a boundary value is of great use. If there is more than one minimal set, we have to choose any one as our minimal hitting set. There are two cases to select the minimal hitting set.

Figure 7 shows the HS_DAG tree for the clusters of test cases given in Figure 6. The hitting sets are {T1, T2, T3} {T1, T2, T8} {T1, T3, T4, T5} {T1, T4, T5, T8} {T1, T4, T7, T8}.When finalizing the minimal hitting set, two cases are taken into consideration.

Case 1: Among the four hitting sets obtained by HS_DAG algorithm, we consider the hitting sets {T1, T2, T3} {T1, T2, T8} as our hitting sets because the number of test cases in these hitting sets are minimum when compared with other sets.

Case 2: From the two hitting sets {T1, T2, T3} and {T1, T2, T8},we have to select one set as the final minimal hitting set for which we consider the total execution time of these hitting sets. The total

Journal of Theoretical and Applied Information Technology
10th November 2014. Vol. 69 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

execution time of {T1, T2, T3} is 2.45 msand the total execution time of {T1, T2, T8} is 2.20 ms. Here we consider {T1, T2, T8} as our minimal hitting set because it covers all the statements in minimum time.

3.7 Performance Analysis

The percentage reduction will be used as a measure for comparative analysis. The formula to find the reduction percentage is given in Equation 3.

Reduction = size of reduced test suite / size of unreduced test suite* 100 --- (3)

According to the example, the size of the test suite after reduction is 30%. The reduction in size before and after minimization is given in Figure 8.

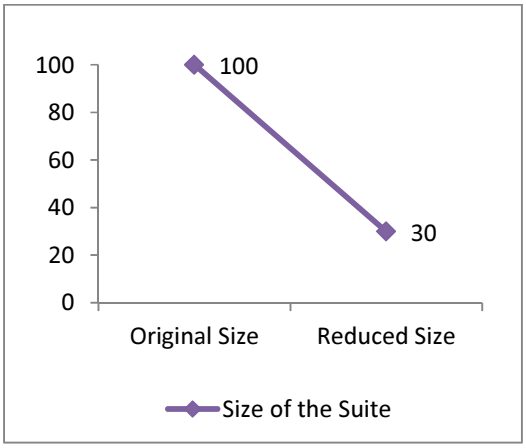

Figure 8: Size of the Test Suite before and after Reduction

4. EXPERIMENTS AND RESULTS

REG_MIN algorithm is implemented using C language. We conducted experiments with five programs of varying sizes and complexity levels to measure the extent of test suite size reduction obtained by the proposed algorithm. The Experimental programs are mentioned in Table 3.

Table 3: Experimental Programs

Program	LoC	# Test Cases
Calculator (CAL)	139	94
Vehicle Management System (VMS)	256	38
Inventory Control System (ICS)	314	47
Library Management System (LMS)	402	53
College Information System (CIS)	560	61

The sizes of reduced suites produced by REG_MIN algorithm for each of the experimental programs are shown in Figure 9.

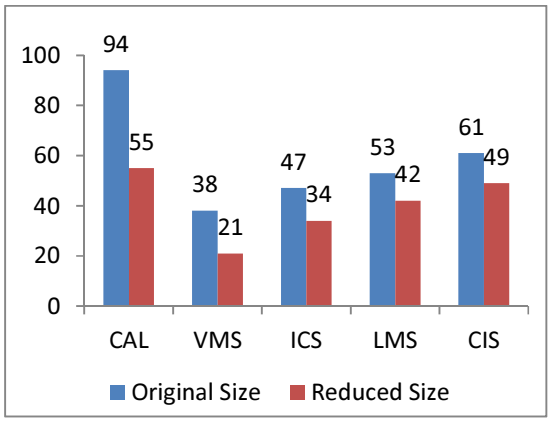

Figure 9: Minimized Test Suites

The size of the test suite before minimization is 100% and the size after minimization is calculated using equation 3.The reduced size is given in Figure 10.

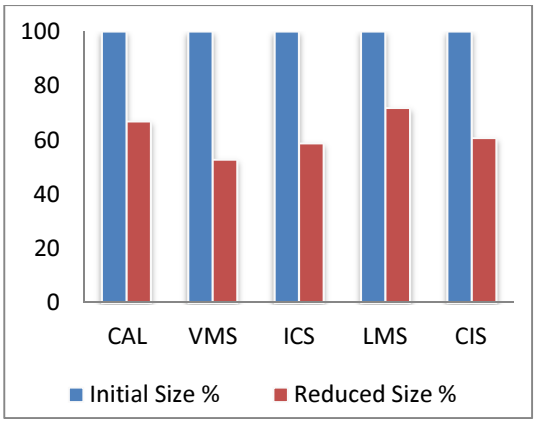

Figure 10: Test suite reductions through REG_MIN algorithm

4 Conclusion and Future Work

Regression testing, that involves test case prioritization, test suit reduction or minimization

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and regression test selection significantly revolves around the criteria that determine the selection and execution of the test cases. This paper feasibly edifies the test-suite minimization through multi-objective technique explicitly through HS_DAG algorithm. It addresses the test-suite minimization from the perspective of minimization of test effort proportionately connected to the maximized optimal statement coverage. Automation of minimized test-suite from HS_DAG can save considerable effort and resources in testing of the software application in the industry. Testing criteria are not generic for all the software applications and are tester specific. In short, more testing criteria are not chosen here for this research to prevent ambiguity and to maintain precision and accuracy of results. However, this strategy shall be advised for large data also. To state precisely, the cost in terms of time requirement to run a test case is an apparent premise in regression testing. Therefore, the entire research discussed in the paper elucidates the requisite modifications to rectify the existing techniques in running a test suite. Above all, the architecture proposed here to execute this research has obviously accomplished the objectives of this study. In fact, the results obtained opens a wide avenue for the further enhancement of this multi-objective test case minimization technique executed through HS_DAG algorithm as discussed in this article.

REFERENCES

[1] W.E. Wong, J.R. Horgan, S. London, and H. Agrawal. A study of effective regression testing in practice.In Proceedings of the Eighth International Symposium on Software Reliability Engineering, pages 230–238, November 1997.

[2] Shin Yoo and Mark Harman. Regression testing minimisation, selection and prioritisation: A survey. Journal of Software Testing, Verification and Reliability, 2011.

[3] 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.

[4] Graves, T.L., Harrold, M.J., Kim, J.M., Porter, A., and Rothermel, G. 2001. An empirical study of regression test selection techniques. ACM Transactions on Software Engineering and Methodology. 10(2), 184-208.

[5] S. G. Elbaum, A. G. Malishevsky, and G. Rothermel. Prioritizing test cases for regression testing. In International Symposium on Software Testing and Analysis, pages 102–112. ACM Press, 2000.

[6] Rothermel G, Harrold M, Ronne J, Hong C. Empirical studies of test suite reduction. Software Testing, Verification and Reliability, December 2002; 4(2):219–249.

[7] Shin Yoo and Mark Harman. Using hybrid algorithm for pareto efficient multi-objective test suite minimisation. Journal of Systems and Software,83(4):689–701, 2010.

[8] Tallam, Sriraman, and Neelam Gupta. "A concept analysis inspired greedy algorithm for test suite minimization." ACM SIGSOFT Software Engineering Notes. Vol. 31. No. 1. ACM, 2005.

[9] Prasad, Saran, et al. "Regression Optimizer A Multi Coverage Criteria Test Suite Minimization."International Journal of Applied Information Systems (IJAIS) ,Foundation of Computer Science FCS, New York, USA Volume 1– No.8, April 2012.

[10] Offutt J, Pan J, Voas J. Procedures for reducing the size of coverage-based test sets. Proceedings of the 12th International Conference on Testing Computer Software, ACM Press, 1995; 111–123.

[11] Yu Y, Jones JA, Harrold MJ. An empirical study of the effects of test-suite reduction on fault localization. Proceedings of the International Conference on Software Engineering (ICSE 2008), ACM Press, 2008; 201–210.

[12] McMaster S, Memon AM. Fault detection probability analysis for coverage-based test suite reduction. Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'07), IEEE Computer Society, 2007.

[13] Chen Y, Probert RL, Ural H. Regression test suite reduction using extended dependence analysis. Proceedings of the 4th International Workshop on Software Quality Assurance (SOQUA 2007), ACM Press, 2007; 62–69.

[14] Leitner A, Oriol M, Zeller A, Ciupa I, Meyer B. Efficient unit test case minimization. Proceedings of the 22nd IEEE/ACM international conference on Automated Software Engineering (ASE 2007), ACM Press, 2007; 417–420.

[15] Smith A, Geiger J, Kapfhammer GM, Soffa ML. Test suite reduction and prioritization with call trees. Proceedings of the IEEE/ACM International Conference on

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Automated Software Engineering (ASE 2007), ACM Press, 2007.

[16] Y. Shin, M. Harman, P. Tonella and A. Susi, "Clustering Test Cases to Achieve Effective and Scalable Prioritisa-tion Incorporating Expert Knowledge," *ACM International Conference on Software Testing and Analysis* (*IS-STA* 09), Chicago, 19-23 July 2009, pp. 201-212.

[17] Shi, Lei, and Xuan Cai. "An exact fast algorithm for minimum hitting set." In *Computational Science and Optimization (CSO), 2010 Third International Joint Conference on*, vol. 1, pp. 64-67. IEEE, 2010.

[18] R. Reiter, "A theory of diagnosis from first principles,"Artif. Intell.,vol. 32, no. 1, pp. 57–95, Apr. 1987.

[19] J. de Kleer and B. Williams, "Diagnosing multiple faults,"Artif. Intell.,vol. 32, no. 1, pp. 97–130, Apr. 1987

[20] Nyberg, Mattias. "A generalized minimal hitting-set algorithm to handle diagnosis with behavioral modes." *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 41, no. 1 (2011): 137-148.