Novateur Publication's
International Journal of Innovation in Engineering, Research and Technology [IJIERT]
ICITDCEME'15 Conference Proceedings
ISSN No - 2394-3696

OPTIMIZATION USING DIFFERENT CRITERION FOR TEST CASE SELECTION IN REGRESSION TESTING

Mr.Vishal Mangave
Student of M.Tech.Technology Dept.Shivaji University, Kolhapur, India

Prof.S.K.Shinde
Associate Professor , Bharati Vidyapeeths Engg,College Kolhapur ,India

ABSTRACT
Modification in the software can cause the breaking of the previously verified functionalities of the system, causing the regression faults. One of the regression testing approaches is to rerun existing test cases. But this leads potentially costly task. Cost minimization can be carried out by optimization of testing effort by executing only a selected subset of test cases that believed to have a better chance of revealing faults. This proposes an approach for selecting and ordering a predetermined number of test cases from existing test cases. These are then prioritized using a greedy algorithm that maximizes minimum coverage in an iterative manner. The proposed approach would provide more consistency compared to existing approaches. The Individual running times of the test cases and total time required for the approach is providing better results than the existing system.

INTRODUCTION
Software is constantly evolved according to required functionalities and with changing needs of customer .But software modification can break previously verified functionalities and cause the regression faults. Regression testing is required for detecting such regression faults. The very basic strategy is to rerun test cases that are available from the earlier version of the software product. But running the entire test cases is much expensive. Regression testing is the process of validating modified software for detection of new errors that have been introduced in to the previously tested code and giving the confidence that modifications are correct. As regression testing is an expensive process, researchers have developed techniques for optimization of regression test suite to make the process cost effective.
The modified software system is not regressed, can be ensured by rerunning existing test cases. But this is much costly task. This cost mitigation can be performed by executing the optimized subset of the test cases with a believe that it more better chance of faults detection. Here we proposing a new approach for selecting and ordering a predetermined number of test cases from an existing test suite.
This approach forms two different coverage-based criteria, by using Integer Linear Programming problem and to find many close-to-optimal solution points using constraint relaxation. The points then combined by using to voting mechanism to obtain a final solution. Then use greedy algorithm to prioritize the selected subset of test cases which iteratively maximizes minimum coverage for the test case.
Two approaches have developed for this :

- Regression Test Selection (RTS)
- Regression Test Prioritization (RTP).

Selection of subset of select a subset of test cases for execution and reducing the efforts is done through RTS techniques ([1],[2]).Test case reordering based on the their importance of faster fault detection is done through RTP techniques([3],[4]). Different studies evaluate that RTS and RTP existing techniques are depends on the different programs and process assumptions. Two most interesting observations are much common among from these studies. The relative performance of technique varies across different programs/processes. This is motivating new kind of regression-testing approaches which are consistent for different programs. Second, the studies show that gap between existing and hypothetical optimal solutions at existing criteria and algorithms are not sufficient.

Fault finding is the probabilistic process, a single criterion might just indicate where the faults reside, thus it limits the fault detection capability potentially. So, using more than one criterion can help us to solve the problem. The proposed technique gives more consistent results in comparison with existing approaches.

PROBLEM STATEMENT

Studies shows that no single technique is providing cost effectiveness in all situations its best. So regression testing techniques are preferred depends upon the characteristics and goals on the technique .This observation implies that no any single criterion will outperform the other criteria's in all cases. This gives encouragement for the developing of multicriteria approach for the problem.

So, considering both time constraints and multicriteria we introduce the Size-constrained Regression Test Selection (SRTS) problem, which is to select a subset of test suite with a given number of test cases, is solved by using multicriteria no greedy optimization technique.

AN OVERVIEW OF RELEVANT LITERATURE

REGRESSION TEST CASE SELECTION

The RTS techniques existing can be divided into the three type's minimization, safe and coverage-based selection techniques. The Minimization-based RTS techniques is for selecting the smallest test case that can satisfy some minimum coverage criteria for the changed code . For example, in 1977, Fischer has used zero-one Integer Programming technique for finding minimal test case subset that can cover each and every element of the system.

The minimization technique discards some number of test cases those are covering modified code that are covered already by some other test cases selected. The Rothermel and Harrold have developed safe RTS technique for the elimination of missing faults. Safe techniques are result in marginal test suite size reduction, but achieving high accuracy ([11],[12]).

The Coverage-based RTS technique uses some requirement coverage on the modified code as the selection criterion; the test case selected which providing the quired coverage ([13]). From these algorithm comparison one observation has been identified that no any single criterion is outperforming the others in all cases. This is encouraging the researchers to develop multicriteria approaches to the problem. Yoo and Harman has developed Pareto efficient multi objective test case selection[8].

REGRESSION TEST CASE PRIORITIZATION

The different prioritization techniques are proposed and had been studied ([6],[7]). The RTP techniques are the algorithms that reordering the test suite by optimizing certain criteria. The most developed techniques are using greedy algorithms that improve criteria that are related to faults. For the given code coverage data, test cases are ordered in a greedy manner in terms of the number of code elements (e.g., methods, statements, or blocks) that they cover.

The greedy algorithms are used by conventional techniques that simplify the search in space of the RTP problem, but optimal solution is never ensured .Studies have been carried for the investigation of the performance of three techniques

1. 2-Optimal greedy approach 2. Genetic Algorithms 3. Hill Climbing

and two coverage-based techniques

1. Greedy and Additional 2. Greedy Algorithms.

The studies also indicate that greedy approaches outperform the Genetic Algorithm and Hill Climbing.As the software development is complex process The much practical problems faced in its development ,if optimization algorithms are used to solve them it is known as Searched-Based Software Engineering (SBSE)[15]. Prioritizing the requirements for the next release, test case generation planning are examples optimization problems.

BACKGROUND

SIZE CONSTRAINED

Size constrained test case selection is the technique in which selecting only a subset of the given test case suite to minimize and control the size of the test cases that are to be executed. So in this way it is called as the Size constrained test selection.

Lets consider an example that there are N test cases and where N = 7 cover M software elements where M=3 and the coverage data can be given below table.

Table 1.Software Elements & Code Coverage

	$n=$ 1	2	3	4	5	6	7	row sum
$m=1$	15	1	2	82	5	78	30	213
$m=2$	36	68	4	90	1	2	60	261
$m=3$	670	17	710	15	590	80	450	2532
column sum	721	86	716	187	596	160	540	3006

Now suppose that we are interested for selecting L test cases such that the Dsum is maximized. Then the solutions are Dsum({1,3,5,7})=721+716+596+540=2573 where L=4. Dsum({1,3,4,5,7})=2573+187=2760 where L=5,Dsum({1,3,4,5,6,7})=2760+160=2920 where L=6.Thus here it is observed that increasing the value of L from 4 to 5 and then to 6 results just in the marginal increases in values of Dsum.

TIME CONSTRAINTS

The studies show that the if the time constraints are absent in the regression testing techniques then the benefits of the techniques are not achieved and not well justified .They also shows that cost and benefits of the running RTS and RTP is justified by revealing more faults as when time constraints is applied for running only the subset of the test cases from the entire test suite. So if it is not required to execute all the test cases then Optimizing the test cases to desired size and ordering them appropriately can be more beneficial that is achieved by applying time constraints. Consider that only L test cases have to be run, then the RTP technique orders all the test cases and then the testers can run only the first L tests and leave out the remaining.

Table2.Fault matrix

	F_1	F_2	F_3	F_4	F_5	F_6
T_1	✓	✓	✓			
T_2				✓	✓	✓
T_3		✓	✓	✓	✓	

The above table shows an example fault matrix. Each check mark shows whether that fault is revealed by that test case. The optimum solution for the RTP designates the T3 as the highest order then T1 then T2.If with such ordering if testers have to execute only two test cases then along with T3 either T1or T2 gets executed and one fault left uncovered.

DESIGN ANALYSIS

METHOD OF DATA COLLECTION.

The test cases are collected from the different sources and stored in a folder. The files of software are taken from the user and stored in the folder.

METHOD OF DATA ANALYSIS.

Analysis can be carried on the size of old test cases and test cases that are selected from the proposed system. Faults detection can be analysed fault detection ratio versus number of run test cases. Timing constraints can be analyzed by analyzing individual running time of test cases.

PROPOSED SYSTEM

The following are the steps in the proposed System.

1) Test cases first collected from available sources.

2) Selection of a D Function that is correlated with score function F.

3) Score function F optimized by two criterias. Dmin ,Dsum

$$D_{sum}(\mathcal{S}) = \sum_{a_m \in \mathcal{A}} w_m d_m(\mathcal{S}) \qquad D_{min}(\mathcal{S}) = \min_{a_m \in \mathcal{A}} w_m d_m(\mathcal{S})$$

4) Dmin is the maximum coverage of the tests case on one of the software element and is one of the optimization criterion it ensures that no software element is left uncovered, provided that for each element, there exist test cases that cover it and that the time constraints allow covering all elements.

5)Dsum is the sum of all the minimum coverages given by Dmin values. It is also used as the second optimization criteria.

6) A multicriteria non greedy selection algorithm is used to select size constrained subset of test cases based on Dmin and Dsum. It consist of the following,

a.Finding a "Elite" window for the subset of the possible solution points.

b.These points are voted for test cases to be included in final solution.

7) Greedy algorithm (Greedy Minimization is then used to prioritize the test cases chosen using the nongreedy selection algorithm.

8) Calculating and analyzing the individual running time of test cases.

9) Minimizing the total time required to perform operation of the proposed system to achieve time constraints on the regression testing.

GMIN ALGORITHM:

In each iteration, steps are

1)It is finding the set of bottleneck elements between the all covered by using at least one of remaining test cases.

2)For of the each remaining test cases ,it then finds number of bottleneck elements that it the test case covers.

3)Then it sorts all the software elements based on ascending value of the code coverage .In case of ties they are breaked using the descending measure of the priority of that element.

4)For finding the test cases that are to be added in iteration it first starts with the greatest number of bottleneck elements that are based on the computation in step 2.

In case of any tie , it finds test case that providing largest coverage of step 2.

The first element in that ordering is computed in step 3.

For the case if the tie does not fully break, it moved to second software element in that order and so forth and this continues until the tie breaks. If there is still a tie after considering all of the elements, the test case with the highest index is selected.

5) Then the current coverage values are being updated to reflect coverage of selected test case. Then the test case is added in the result set.

The Steps 1 to 5 are repeated until all the test cases are added in the result set.

MULTICRITERIA NON GREEDY SELECTION

There are two criteria's Dsum and Dmin. Consider the vector x=(x1,x2,x3,…..,xN) in which xn designates whether nth test case be selected. Problem of selecting the L test cases for optimizing Dsum can be formulated as

$$\begin{aligned} \text{maximize} \quad & \sum_{n=1}^{N} \sum_{m=1}^{M} w_m \delta_m(n) x_n \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i = L \\ & x_n \in \{0,1\} \quad \text{for } n = 1, 2, \ldots, N. \end{aligned}$$

and problem of optimizing the Dmin is formulated as

$$\begin{aligned} \text{maximize} \quad & \min_{m=1}^{M} \sum_{n=1}^{N} w_m \delta_m(n) x_n \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i = L \\ & x_n \in \{0,1\} \quad \text{for } n = 1, 2, \ldots, N, \end{aligned}$$

BLOCK DIAGRAM

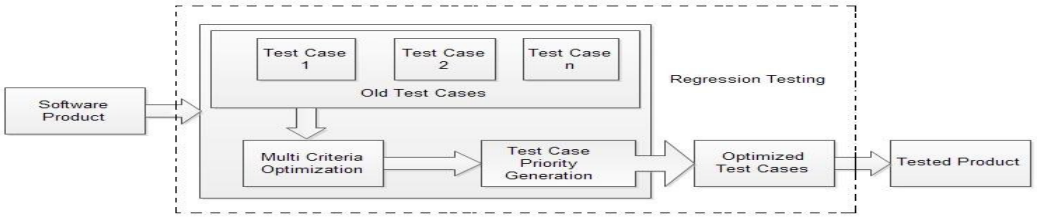

Fig 1: Proposed System Architecture

The software product taken for testing is the input for the proposed system. Regression testing is comprised of number of modules First the number of old test cases that are used for testing old test version of the product are collected. These are shown in the diagram as Test cases1 to n. These are then given to a multicriteria nongreedy optimization algorithm for the selection of only those test cases that are executed within given time limits. Such a test cases are referred as the size constrained test cases. After selecting the test cases in the optimization algorithm these are then given to test case priority generation module to assign a priorities to test cases so that with those they can be executed much faster within given limits of time and size. These are shown as optimized test cases that are applied to perform final regression testing. So, the dotted square box forms the entire regression testing process. And finally the optimized test cases are then applied on the new version of the software product to carry out regression testing phenomenon.

RESULTS

The first process within the regression testing is to provide the input project and its test cases to our project. The following snap is the detail information about the input

Program	Test C...	Test C...	Test C...	Test C...	Test C...	Test C...	Test C...	Total D Min
ALogin.java	8.0	18.0	50.0	2.0	18.0	14.0	2.0	112.0
AMainPage.java	0.0	12.0	48.0	75.0	27.0	30.0	12.0	204.0
AMainPageAddStoc...	64.0	16.0	400.0	4.0	16.0	200.0	36.0	736.0
AMainPageShowSto...	5.0	5.0	45.0	5.0	45.0	35.0	20.0	160.0
NAddStock.java	64.0	4.0	256.0	4.0	256.0	36.0	16.0	636.0
	141.0	55.0	799.0	90.0	362.0	315.0	86.0	1848.0

Fig.2 Test case Code Coverage First Version

Project and its forms and test case columns show the code coverage of each test type of test case on each project form. The each project form is a software element and is shown as the amount of code coverage the test case is covering form. It shows two types of values Dmin and Dsum .Dmin is the last column and the overall sum is the Dsum.By using these two values the number of test cases are optimized.

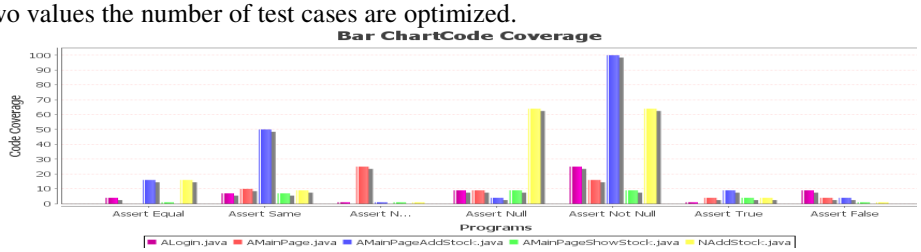

Fig.3 Code Coverage Bar Chart

The above bar chart is the code coverage bar chart for the different kinds of the test cases on different software elements .It is code coverage versus programs or software elements.

The following table shows the basic scenario of the regression testing in which test cases are reapplied for the testing by using optimization, it is shown by using test case type 2 and test case type 7 which are ignored in the testing. So its code coverage is 0. For the newly modified code as the added forms (purchase and view stock) it shows code coverage. So this is the second version code coverage table with Dmin and Dsum values.

Program	Test C...	Test C...	Test C...	Test C...	Test C...	Test C...	Test C...	Total D Min
ALogin.java	8.0	0.0	50.0	2.0	18.0	12.0	0.0	**90.0**
AMainPag...	0.0	0.0	48.0	75.0	27.0	24.0	0.0	**174.0**
AMainPag...	64.0	0.0	400.0	4.0	16.0	128.0	0.0	**612.0**
AMainPag...	5.0	0.0	45.0	5.0	45.0	15.0	0.0	**115.0**
NAddStoc...	64.0	0.0	256.0	4.0	256.0	24.0	0.0	**604.0**
Purchase...	128.0	8.0	512.0	8.0	512.0	72.0	32.0	**1272.0**
Sale.java	0.0	0.0	252.0	112.0	252.0	595.0	0.0	**1211.0**
ViewSale....	0.0	0.0	216.0	96.0	216.0	510.0	0.0	**1038.0**
ViewStoc...	80.0	20.0	500.0	5.0	20.0	250.0	45.0	**920.0**
	490.0	**83.0**	**3078.0**	**401.0**	**1724.0**	**1945.0**	**163.0**	**7884.0**

Fig.4. Code Coverage Second Version

The following is the test case bar chart showing number of different kinds of test cases that has been executed.

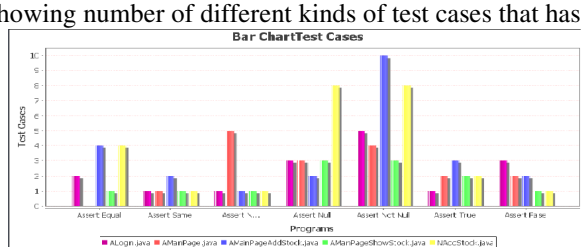

Fig.5 Test Cases Bar Chart

The time differences between the normal testing and the use of regression testing can be given by using the following comparison graph. The graph generated shows the two different lines. The upper line shows the amount of time for testing the first version of the software and second lower line shows the amount of time required testing the second version of the software product. The performance lines show that regression testing is minimizing the time required of testing. The second version testing comprise of the minimized test suite and some test cases for the modified code in the new version.

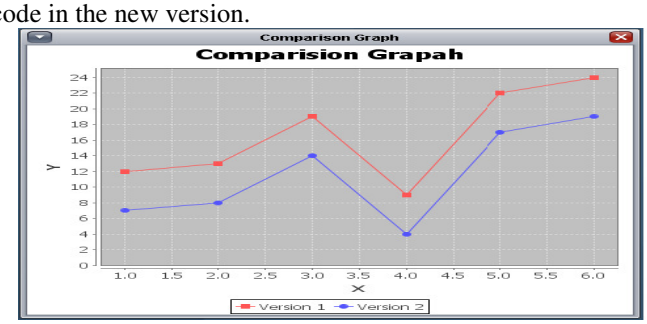

Fig.6 Comparison Graph

CONCLUSION

Thus we have gathered information about regression testing and studied different approaches on it. By comparing various approaches, we analyzed that each model has some drawbacks and has some characteristics for which it is used. No any single technique or its criteria is effective in different circumstances. So, we proposed a approach for regression testing which is based on two criteria's so a multicriteria approach for regression testing and then applying size reduction on the available test cases name size constrained regression testing using multicriteria optimization. From our experiments it shows that our approach is performing better than other techniques. It provides better consistency compared to existing systems. We are also considered the time constraint on the test cases.

Thus the proposed regression testing technique is helping the software development life cycle to minimize the regression testing execution time. The software development testing phase has been modified for providing the better efficiency in the testing phase and thus provide efficient software development model. The proposed idea can be extended to more than two criteria in addition time constraint based can be applied to get more efficient and effectiveness.

REFERENCES

[1] D. Binkley, "Semantics Guided Regression Test Cost Reduction," IEEE Trans. Software Eng., vol. 23, no. 8, pp. 498-516, Aug. 1997.

[2] L. Briand, Y. Labiche, and G. Soccar, "Automating Impac t Analysis and Regression Test Selection Based on UML Designs," Proc. Int'l Conf. Software Maintenance, pp. 252-261, 2002.

[3] S. Elbaum, A.G. Malishevsky, and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies," IEEE Trans. Software Eng., vol. 28, no. 2, pp. 159-182, Feb. 2002.

[4] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Trans. Software Eng., vol. 27, no. 10, pp. 929-948, Oct. 2001.

[5] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, "An Empirical Study of the Effect of Time Constraints on the Cost-Benefits of Regression Testing," Proc. ACM SIGSOFT Int'l Symp. Foundations of Software Eng., pp. 71-82, 2008.

[6] S. Mirarab and L. Tahvildari, "A Prioritization Approach for Software Test Cases on Bayesian Networks," Proc. Int'l Conf. Fundamental Approaches to Software Eng., pp. 276-290, 2007.

[7] D. Jeffrey and N. Gupta, "Test Case Prioritization Using Relevant Slices," Proc. Int'l Computer Software and Applications Conf., pp. 411- 420, 2006.

[8] S. Yoo and M. Harman, "Pareto Efficient Multi-Objective Test Case Selection," Proc. Int'l Symp. Software Testing and Analysis, pp. 140-150, 2007.

[9] K.R. Walcott, M.L. Soffa, G.M. Kapfhammer, and R.S. Roos, "Timeaware Test Suite Prioritization," Proc. Int'l Symp. Software Testing and Analysis, pp. 1-12, 2006.

[10] P. Tannenbaum, Excursions in Modern Mathematics. Prentice Hall, 2006.

[11] T.L. Graves, M.J. Harrold, J.-M. Kim, A. Porter, and G. Rothermel, "An Empirical Study of Regression Test Selection Techniques," ACM Trans. Software Eng. and Methodology, vol. 10, no. 2, pp. 184- 208, 2001.

[12] G. Rothermel and M.J. Harrold, "Analyzing Regression Test Selection Techniques," IEEE Trans. Software Eng., vol. 22, no. 8, pp. 529-551, Aug. 1996.

[13] S. Mirarab,Soroush Akhlaghi, L. Tahvildar,"size constrained regression test case selection using multicriteria optimization", IEEE Trans. Software Eng., vol. 38, no. 2, pp. 936-956, july. 2012

[14] L. White and H. Leung, "A Firewall Concept for Both Control- Flow and Data-Flow in Regression Integration Testing," Proc. Int'l Conf. Software Maintenance, pp. 262-271, 1992.