Test Case Prioritization Techniques "An Empirical Study"

[1]Neha Sharma
Department of
Computer Science and Engineering.,
ITM University
Gurgaon, India
nehasharma9315@gmail.com

[2]Sujata
Department of
Computer Science and Engineering.,
ITM University
Gurgaon, India
sujata@itmindia.edu

[3]Prof. G.N. Purohit
Banasthali Vidhypith
Banasthali, India
gn_purohitjaipur@yahoo.co.in

***Abstract*— Regression testing is an expensive process. A number of methodologies of regression testing are used to improve its effectiveness. These are retest all, test case selection, test case reduction and test case prioritization. Retest all technique involves re-execution of all available test suites, which are critical moreover cost effective. In order to increase efficiency, test case prioritization is being utilized for rearranging the test cases. A number of algorithms has been stated in the literature survey such as Greedy Algorithms and Metaheuristic search algorithms. A simple greedy algorithm focuses on test case prioritization but results in less efficient manner, due to which researches moved towards the additional greedy and 2-Optimal algorithms. Forthcoming metaheuristic search technique (Hill climbing and Genetic Algorithm) produces a much better solution to the test case prioritization problem. It implements stochastic optimization while dealing with problem concern. The genetic algorithm is an evolutionary algorithm which gives an exact mathematical fitness value for the test cases on which prioritization is done. This paper focuses on the comparison of metaheuristic genetic algorithm with other algorithms and proves the efficiency of genetic algorithm over the remaining ones.**

Keywords—regression testing; test case prioritization; genetic algorithm; greedy algorithm; APFD

I. INTRODUCTION

Software testing throughout maintenance phase is the most difficult task for software engineers. They depend on reusability of test suite in regression testing that incurs the cost which is a major concern [4]. Several regression techniques such as test case selection and reduction techniques are offering the kind of relief in this concern. Selection and minimization of test cases can certainly retain comprehensive coverage, however, empirical results give an evidence which defines that safety in addition to fault detection capability of the test suite is a compromise [1][2]. The primary goal of test case prioritization regression technique is to achieve maximal feasible coverage in addition to have a higher fault detection rate sooner and also to increase the reliability of the system [3][5][10]. Whereas accomplishing test case prioritization is quite sophisticated with solely source code that's the reason historical information such as test case cost, fault covered, fault severity allows highest test performance [9].

II. TEST CASE PRIORITIZATION PROBLEM

Test cases in the test suites are reschedule which will further prioritize using an algorithm. Before dealing with prioritization algorithms the problem associated with test case prioritization requires understanding which is defined as follows [3] [5] [9] [11]:

978-1-4799-5958-7/14/$31.00 ©2014 IEEE

Given: T is test suites, PT refers to a number of ways they are chosen, where f is a function whose value depends on permutation of these T to some real number.

Problem: We have to find T' such that T' ∈ PT For all T, (T '∈PT) where (T! = T') and f (T') >= f (T).

III. RELATED WORK

Test case prioritization is broadly divided into two categories: general test case prioritization and version specific test case prioritization [3][5]. To know general test case prioritization, consider a program P has test suite T with prioritized suite T' which is calculated without having familiar with modified program P'. It gives more successful prioritized suite than the original one. Within version specific test case prioritization, knowledge of modified program P' must require for prioritizing T test suite. Version specific prioritization may be more effective for modified program P' as compare to general test case prioritization, which is less effective in a successive release version [3] [5]. Some of the test case prioritization techniques achieve 100% code coverage. Throughout this paper, we get to know about the techniques of test case prioritization with its most applicable approach for implementation.

A. Approaches for Test Case Prioritization Techniques

Existing algorithms for test case prioritization are based on greedy approach. Greedy algorithm selects maximum weighted element based on the criteria decided to be chosen. It can be statement coverage, branch coverage, function coverage and fault coverage. Some of the techniques are mentioned in table I. The same procedure is followed until gets the order of test cases with suboptimal solution[11]. Elements are sorted in an increasing order using quick sorting which gives complexity of O(mn) where m is the no. of statements and n is no. of test cases. The major drawback of greedy algorithm is that it gives a local optimum solution of the problem concern. The solution provided could be either maximal or minimal based on neighboring available test cases. Next to the simple greedy algorithm, additional greedy algorithm came into existence which is similar as Greedy algorithm with the use of different strategies. The test cases that focus on maximum coverage are selected by using greedy algorithm and the information of previous covered code additionally used to perform greedy algorithm which gives complexity of order O (mn^2). Another 2-Optimal greedy algorithm used which is based on the travelling salesmen problem i.e. "finding the minimum cost path passing through every node in a graph G at least once"[7]. As per the statement test cases, achieving complete coverage earlier has given highest priority. The pair of test case is selected with some readjustment of coverage information which gives complexity of O(mn^3). 2-Optimal algorithm and additional greedy algorithm are little similar [11].

Popularize metaheuristic search techniques help in finding a solution to a certain problem with reasonable computational cost. It implements some sort of stochastic optimization in which the resultant solution is dependent on the number of random variables generated. Hill climbing and Genetic algorithm comes under such technique. Hill climbing algorithm is very simple to implement. In this algorithm, the initial solution state is randomly selected which will further compare with its neighboring states. If the neighboring state has higher fitness then it will become the current state and thus the most appropriate state has been chosen to get a solution state. State here, refers to the test suite contains test cases and neighbors are the same test suite with different ordering of test cases [11].

All the above algorithms known till now gives suboptimal results, but not globally optimal. It can achieve using Genetic Algorithm which imitates the Darwinian theory of biological evolution. It evaluates individuals from the population of test cases. The selected couple of individuals will combine and mutated to offer a new generation before suitable termination condition has been met. The most advantageous feature of using a genetic algorithm is assigning the priority based on the fitness value of an individual. The major issue of cost must be taken into consideration before applying any approach of test case prioritization. It may happen cost is less during the starting of the applied order, but it will go to high at the ending stage. To fulfill this issue we must determine the fitness value of an individual on which entire ordering has been done using genetic algorithm [6] [8] [11].

$$\text{Fitness} = 2 * (\text{Pos} - 1) / (\text{Nind} - 1) \quad (1)$$

Where, Pos is the position of an individual and Nind is population size.

Depending on coverage criteria a number of metrics have been proposed in literature as mentioned below [11]:

- Average percentage block coverage (APBC): It focuses on the rate of block coverage by prioritizing test suite.
- Average percentage decision coverage (APDC): It gives the rate of covering branches or decision by the prioritize test suite.
- Average percentage fault detection (APFD): It gives the maximum fault detection rate by test suite prioritization.
- Average percentage statement coverage (APSC): It considers the rate at which test suites covers the maximum statements.

Table I does not focuses the earlier prioritization techniques like untreated, random and optimal which comes before the following said six techniques[11]. These techniques can be used under any of the approaches (greedy or metaheuristic approach). The selection of an approach based on the problem concern.

TABLE I. LIST OF SOME PRIORITIZATION TECHNIQUE

Code	Mnemonic	Description	Approach Used
C1	Total-stmt	prioritization based on statement coverage	Greedy
C2	Addtl-stmt	prioritization based on uncovered statement using earlier coverage information	Greedy
C3	Total-branch	prioritization based on branch coverage	Greedy
C4	Addtl-branch	prioritization based on uncovered branches using earlier coverage information	Greedy
C5	Total-FEP	prioritization based on probability of fault exposing potential	GA
C6	Addtl-FEP	prioritization based on probability of fault exposing potential using earlier test cases impact	GA

IV. EMPIRICAL STUDY AND RESULTS

This paper contains the experimental results showing the comparison between the metaheuristic genetic approach and greedy approach by measuring an efficiency with available APFD (Average percentage of fault detection) metric. [11].

$$APFD=1-(TF_1+TF_2+TF_3+\ldots..+TFm)/(nm)+1/2n \quad (2)$$

Here, m is no. of statement, n is no. of test cases, TFi is the location of the test case finding the i^{th} fault early. The same formulae is employed for the other metrics like APBC (Average percentage of fault detection), APSC (Average percentage of statement coverage), APDC (Average percentage of decision coverage), and APFD (Average percentage of fault detection) based on the coverage criteria like statement, decision and branch coverage. Here, in this paper author is using APFD measure for the efficiency of a greedy and metaheuristic approach. Each test case is represented by its position in the test suite. We consider the number of faults with their corresponding test cases for achieving complete coverage.

TABLE II SAMPLE PRIORITIZATION SPECIFIES FAULT EXPOSED

Test cases	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	No. of fault detected
T1	*	*	*								3
T2	*		*	*	*		*		*	*	7
T3	*	*	*	*							4
T4	*		*	*	*		*				5
T5					*	*	*	*	*	*	6

Test case order highly impacts on the effectiveness of the test suite for performing prioritization of test cases.
For the evaluation of greedy and genetic approaches two basic programs of database management have been taken into consideration. These programs focus on view, and conflict serializabilty of the database. There are 45 and 53 LOC (line of code) having 100 and 114 no. of test cases respectively.
Table II consists of its partial test suit which shows the total number of test cases, faults revealed and total number of faults detected. The order of test cases from table II differs for both the approaches.

The performance of Genetic algorithm depends on selection of fitness function, mutation probability and crossover probability, whereas Greedy algorithms gives higher priority to higher fault detected test case. We get prioritize order P (T4-T5-T3-T2-T1) from example program as explain in Table II ,by applying 2-Optimal algorithm and order Q (T1-T3-T4-T5-T2) using genetic algorithm with the use of fitness function and mutation probablity as 0.8 with crossover probablity 1.0. The APFD measure is different for both the order. The coverage through these orders is shown in figure-1 and figure-2. Its x-axis defines the coverage area where, y-axis defines coverage done by particular test case. The order Q covers more area with the initial test cases as compare to order P. It clearly defines GA (genetic algorithm) give higher efficiency with early complete coverage as compared with greedy algorithm.

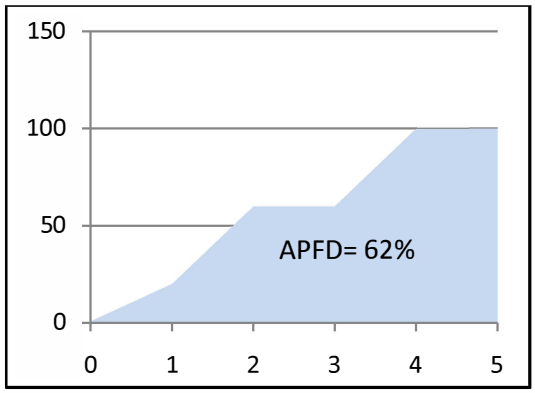

Fig. 1. Example demostrate APFD measure : APFD for priortized test suite Q

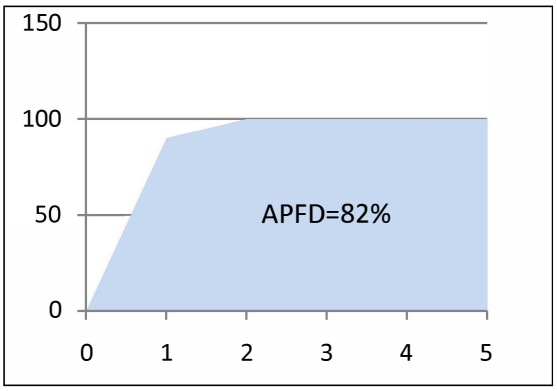

Fig. 2. Example demonstrate APFD measure : APFD for priortized suite P

The fault to be detected were initially be mutated for analyzing the data sets. Using candlestick diagrams efficiency is shown for both the programs. The progress in APFD value of test suite for various prioritization techniques in table I shows in figure-3 and figure-4. The high value of the sticks represents highest coverage attain by applying an individual technique. The least coverage of test cases in stick is described by its low value. The body of the candle represents the opening and closing efficiency of the test case coverage. Figure-3 and figure-4 shows the box plots representation of APFD values of six categories of test case prioritized technique for both the programs.

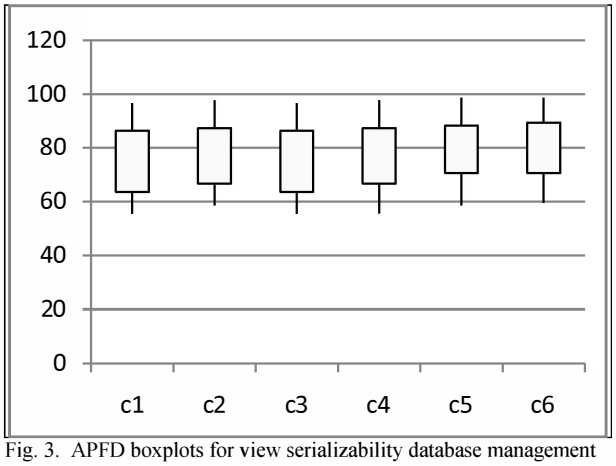

Fig. 3. APFD boxplots for view serializability database management

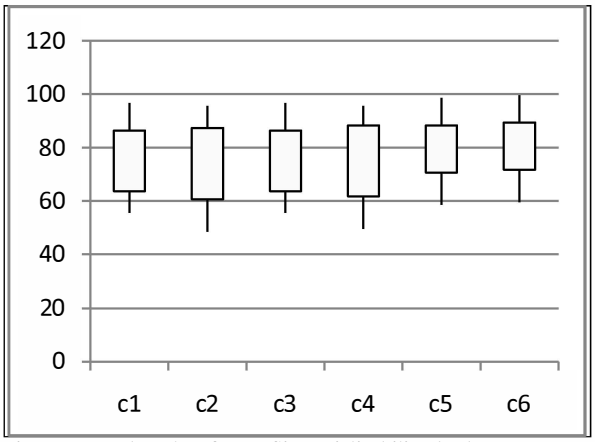

Fig. 4. APFD boxplots for conflict serializability database management

The techniques are: C1: total-stmt, C2: addtl-stmt, C3: total-branch, C4:addtl-branch, C5:total-FEP, C6:addtl-FEP.

The first four techniques in table I come under coverage based technique. The resultant efficiency in figure 5 is calculated by taking mean that defines efficiency of additional coverage technique outperformed total coverage based techniques. Additional branch coverage prioritization outperformed all the code coverage techniques. Fault Exposing Potential refers to the ability of the test case to expose the fault with its severity. On applying other algorithms rather than genetic algorithm deviations in APFD values are not clearly visible, but as it based on fitness values give the exact mathematical solution. Thus, the resultant efficiency of additional FEP (fault exposing potential) is higher than total FEP technique and other coverage based techniques with genetic algorithm. Figure 6 shows that the efficiency of using genetic algorithm is high as compare to greedy algorithm as we move towards the fault exposing test case prioritization technique.

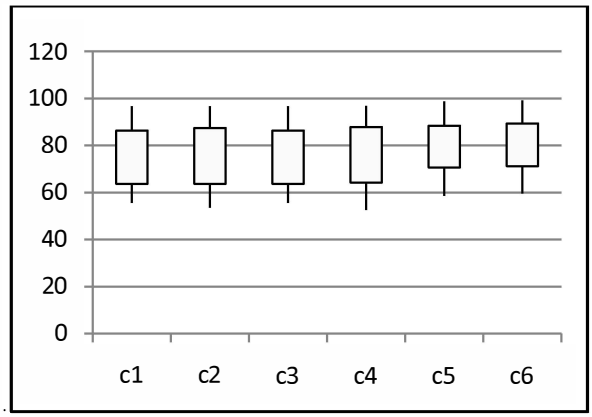

Fig. 5. Resultant APFD boxplots for view and conflict serializability database management.

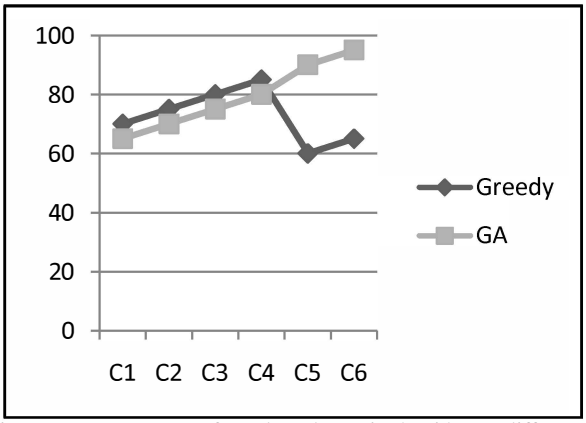

Fig.. 6. APFD measure of greedy and genetic algorithm on different priortization techniques

V. CONCLUSION AND FUTURE WORK

In this paper various algorithms for test case prioritization are described with their comparisons. We can achieve a global optimal solution by using a metaheuristic genetic algorithm efficiently. Here, we compare two different approaches and concluded that genetic algorithm is the most appropriate algorithm to deal with all the available techniques, but all these techniques including genetic are focusing on function i.e. to increase the fault detection rate. There must be some trade through which all the objectives of test case prioritization can be achieved by a single algorithm. So, future work of this paper will focus on accomplishment of all aspects and the various objectives of test case prioritization

ACKNOWLEDGMENT

Author would like to take this opportunity to express her profound gratitude and deep regard to Ms. Sujata, for her exemplary guidance in complete research and its implementation. Her valuable suggestions were of immense help throughout the proposed work.

REFRENCES

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

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

[3] 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.

[4] K. Onoma, W-T. Tsai, M. Poonawala, and H. Suganuma, "Regression Testing in an Industrial Environment", Comm. ACM, vol. 41, no. 5, pp.81-86, May 1988.

[5] S. Elbaum, A. Malishevsky, and G.Rothermel ' Test case prioritization: A family of empirical studies". IEEE Transactions on Software Engineering, February 2002.

[6] Srivastava, Praveen Ranjan, and Tai-hoon Kim: "Application of genetic algorithm in software testing" International Journal of software Engineering and its Applications, vol. 3, no. 4, pp. 87-96, 20

[7] S Lin, "Computer Solutions of the Travelling Salesman Problem," Bell System Technical J., vol. 44, pages2245-2269, 1965.

[8] Wang Jun, Zhuang Yan, Jianyun Chen : "Test Case Prioritization Technique based on Genetic Algorithm" 2011 International Conference on Internet Computing and Information Services, 2011

[9] Yu-Chi Huangc, Kuan-Li Penga, Chin-Yu Huanga " A history-based cost-cognizant test case prioritization technique in regression testing " Science Direct. The Journal of Systems and Software 85 (2012) 626–637

[10] Yu, Yuen Tak, and Man Fai Lau: "Fault-based test suite prioritization for specification-based testing" Information and Software Technology, vol. 54, no. 2, pp. 179-202, 2012.

[11] Zheng Li, Mark Harman, and Robert M. Hierons " Search Algorithms for Regression Test Case Prioritization " IEEE Trans.Software Eng., vol. 33, no.
4, april 2007