IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Vol.-6, Issue-11, Nov 2018 E-ISSN: 2347-2693

ACO for Regression Testing By the Process Automated Slicing

Riza Dhiman*[1], Vinay Chopra[2]

[1,2] DAV Institute of Engg and Technology, Jalandhar, Punjab, India

Available online at: www.ijcseonline.org

Accepted: 24/Nov/2018, Published: 30/Nov/2018

Abstract- The Regression testing is used to retest the component of a system that verifies that after modifications. The test case prioritization is the technique of regression testing which prioritizes the test cases according to the changes which are done in the developed project. This work is based on automated and manual test case prioritization To test the new version of software test case prioritization is applied which prioritize the test cases according to changes and generate maximum number of faults. In this work, technique is been proposed which will traverse the DFD of the project and calculate the function importance which is calculated automated slicing. The functional importance values are given as input to hill climbing algorithm which prioritizes the test cases in the ascending or descending order according to function importance. The algorithm is performed in MATLAB and it is detect more faults in less time period.

Keywords: Regression Testing, Test Case Prioritization, m-ACO, Automated slicing, FTV(function traversal value).

I. INTRODUCTION

Software testing is a procedure of testing or comparing the actual outcome with the expected outcome. Testing of the software is being done in order to check the correct functionality of the system or project. If the testing will not be performed then system may lead to catastrophic or improper results in the field. So it's better to check or test the system earlier, so that the excellent results can be produced.

1.1. Regression Testing

It is a software testing that refers to section of the test cycle in which programs are being tested to make sure that changes do not affect features that are not believed to be affected. The process of verifying the customized software in the maintenance phase is commonly known as Regression testing. Time and budget constraints are the major disadvantage due to its complex process. Regression testing is basically the re-execution of a number of subset of test that has been previously conducted. In regression testing as integration testing takings, number of regression tests increases and it is not practical approach and ineffective to re-execute every test for each program function if once change occurs. It is quite an expensive testing process that is being used to detect regression faults. Research has been shown that at least 50% of the total software cost is comprised of testing activities [1].

1.2. Techniques for Regression Testing

1.2.1. Retest All

It is one of the methods for regression testing in which all the tests that are present in the existing test bucket or suite has been re-executed. This is very expensive as it requires huge time and resources.

1.2.2. Regression Test Selection (RTS)

Due to expensive nature of "retest all" technique, Regression Test Selection is being performed. In this technique instead of rerunning the whole test suite we must select a certain part of test suite to rerun if the cost of selecting a certain part of test suite is less than the cost of running the tests that RTS allows us to omit [2].

1.2.2. Test Case Prioritization

A mechanism is needed for arranging a test case in an appropriate order to increase their effectiveness in order to meet some performance goal and rate of fault detection such mechanism is commonly known as test case prioritization. Test case prioritization is a method to prioritize and schedule test cases in an appropriate order. Test cases that are having higher priority must be run before than the lower priority test case in order to minimize time, cost and effort during software testing phase. Various performance goals such as rate of fault detection which is a measure of how quickly the fault is being detected so that during testing faster feedback can provide about system under testing and allow the software tester to correct the software at earlier phase as possible [3].

1.3. Test Case Prioritization

The Regression testing is technique in which changes can be tested using existing test cases. The test case prioritization is the technique which will prioritize the test cases according to their priority which depends upon changes made in the project. The various techniques have been proposed in recent times for test case prioritization and these techniques are [4]:-

1.3.1. Customer-requirement based Technique

In this technique Customer requirement factors are taken into account and provided some weights and based of these values test case weight for requirement is evaluated. Test cases with high weights value are executed first following the ones with lower value. Customers requirement factors are Customer assigned priority on requirement, Requirement complexity and Requirement volatility [5].

1.3.2. Coverage-based Technique

It is based on code coverage analysis and the measurement of code covered by a test case. Various coverage criterions are considered and the amount of coverage is evaluated and used to prioritize the test cases. Coverage-based technique is a white-box testing technique *i.e.*, a method that tests internal structures of software [6].

1.3.3. Cost effective based Technique

This technique prioritizes the test cases which are based upon costs factors like cost of operation of test cases, cost of analysis, cost of prioritization, cost of execution, validating test cases. Cost is of two types *i.e.*, Direct cost include test selection, test execution, result analysis and Indirect cost include overhead cost and tool development cost.

1.3.4. Chronographic History-based Techniques

This technique prioritizes the test cases based on test case's earlier executions in order to enlarge or reduce the probability that it will be considered into account in current test execution [7].

II. LITERATURE SURVEY

Leung *et al.*, [1989] identified that regression testing can be grouped into progressive regression testing and corrective regression testing depending on whether the specification is changed or not. The test cases can be grouped into five classes: reusable, retestable, obsolete, new-structural and new-specification test cases [1].

Khan et al. [2006] described a test case reduction technique called Test filter that reduces the size of test suites by simply eliminating unnecessary test cases and also decreases the test case storage, management and execution cost. The results show that our technique is quite beneficial in identifying non-redundant test cases at a quite little cost. Ultimately it is beneficial to optimize time & cost spent on testing and it is also helpful during regression testing [2].

Zheng *et al.*, [2007] identified most effective algorithm in solving the test case prioritization problem for regression testing and factors that could affect the efficiency of the algorithm *i.e.*, this paper addresses the problems of choice of fitness metric, characterization of landscape modality and determination of the most suitable search technique to apply [3].

Rothermel *et al.*, [2009] described multiple techniques for prioritizing test cases and measure the effectiveness of these techniques by using metrics for improving the rate of fault detection and report the empirical results that measure the effectiveness of these techniques for improving rate of fault detection [4].

Daengdej *et al.*, [2010] identified two efficient prioritization methods aim to resolve the problem of many test cases assigned the same weight values. The second method is being developed to effectively prioritize multiple suites. As a result of which this paper discussed an ability to reserve high prioritize tests in multiple suites while minimizing a prioritization time[5]

Chen *et al.*, [2010] described dependence analysis based test case prioritization technique. In this firstly they have analyze the dependence relationship using control and data flow information then construct a weighted graph and do impact analysis to identify modification-affected elements. After that, prioritize test cases according to covering more modification-affected elements with the highest weight [6].

Haidry *et al.*, [2012] described the concept of functional dependency that exists between test cases. In this paper two techniques are used to find the dependency between the test cases namely: open dependency and closed dependency and the technique which is used to assign the priority to test case based on dependency information of the test cases are known as dependency structure prioritization [7].

Jatain *et al.*, [2013] illustrated various techniques of regression testing and test case prioritization and also describes various search algorithms used in the process of test case prioritization. This paper also list various challenges associated with the requirement based prioritization and also proposes an approach to overcome these challenges [8]. Muthusamy *et al.*, [2014] identified that Test Cases are prioritize on the basis of following factors namely: Prioritization Weight Factor, Customer Allotted Priority, Developer-observed Code Implementation Complexity, Changes in Requirements, Fault Impact of Requirement, Completeness, and Traceability [9].

Joshi *et al.*, [2014] described a new prioritization technique that prioritize test cases in a descending order for Component Based Software Development (CBSD) by using the concept of Prim's algorithm. It makes use of CIG (Component Interaction Graph) as input for medium/large size CBSD (Component Based Software Development Process) [10].

Konsaard *et al.,* [2015] described an algorithm that prioritize the test cases based on total coverage which includes five steps namely: graph generation, test case generation, test suite generation, fitness calculation and genetic algorithm. Its performance on the average percentage of condition covered and execution time are compared with other approach [11].

Wang *et al.,* [2015] illustrated that most of current regression test case prioritization researches neglect to use internal structure of software which is a significant factor that influence the prioritization of test cases. In this paper prioritization approach schedules test cases based on dependence analysis of internal activity in service oriented workflow application [12].

III. NEED OF REGRESSION TESTING

- Change in requirements and code is modified according to the requirement.
- New feature is added to the software.
- Defect fixing.
- Performance issues fix.

3.1 Algorithms For Test Case Prioritization

Greedy Algorithm

It is based upon the principle that the element with the highest weight is taken into account first, followed by the element with second-highest weight and this process continues until a complete solution has been obtained. It is quite a simple algorithm but in some situations where the results are of high quality it is also prove to be attractive one because it is quite inexpensive both in terms of implementation and execution time.

Additional Greedy Algorithm

The Additional Greedy Algorithm is one of the type of Greedy Algorithm, but it follows quite different process. It combines feedback from previous selection and randomly selects the maximum weighted element of the problem from that part that is not being already consumed by the previously selected elements.

Genetic Algorithm

The population is a set of random individuals. In which each individual is represented by the sequence of genes commonly known as the chromosome. In this selection procedure depends upon the fitness value which decides that which individuals are to be selected as the "parents" for producing the next generation. Crossover is a genetic operator which combines two individuals in order to produce a new individual known as offspring. The mutation operator will alter one or more gene values in the individual depending on the probability of mutation.

IV. ANT COLONY OPTIMIZATION ALGORITHM

Ant colony Optimization algorithm is a mathematical optimization Approach which belongs to the family of local search. Therefore because of this reason it is also known as local search approach. It is an iterative algorithm that starts its search from an arbitrary solution of the problem and then it will find a better solution by simply changing a single element of the solution. If the alteration will produce a better solution than a change is becomes a new solution, repeating this process until no further improvements can be found.

The procedure of ACO Meta-heuristic algorithm can be defined as given steps.

ACO ALGORITHM

```
While(Not termination)
Generate solutions()
daemon Actions
pheromone Update
      end while
```

The Ant colony Optimization algorithm for test case prioritization is composed of the following steps:

1. Pick a random solution state and make this the current (i.e.initial) state.
2. Evaluate all the neighbors of the current state.
3. Move to the state with the largest increase in fitness from the current state. If no neighbor has a larger fitness than the current state, then no move is made.
4. Repeat the previous two steps until there is no change in the current state.
5. Return the current state as the solution state [3].

4.1 Applications of Ant colony Optimization Approach

- Ant colony Optimization can be applied to any problem where the current state allows for an accurate evaluation function. For example, the travelling salesman problem, the eight-queens problem, circuit design and a variety of other real-world problems.
- Ant colony Optimization has been used in inductive learning models
- Ant colony Optimization has also been used in robotics to manage multiple-robot teams which allows scalable and efficient coordination in multi-robot systems.
- Ant colony Optimization allows robots to choose whether to work alone or in team.[22]

V. RESEARCH METHODOLOGY

The regression testing is the type of testing which tests the changes made in the project by prioritizing the test cases. To prioritize the test cases, functional dependency is calculated

using automated slicing. The functional dependency is based on number of functions associated and number of functions affected when certain change is made in the function. To calculate number of associated functions and number of functions affected, automated slicing is applied using m-ACO . The Hill climbing algorithm is the Greedy algorithm which is based on the initial population, mutation value and fitness functions. The function's importance is given as initial population on which mutation formula is applied and mutation value of particular function is calculated until best value is achieved which is known as fitness value. To calculate functional importance equation number 1 is applied

$$\text{Functional Importance}=\frac{number\ of\ clicks}{number\ of\ assocated\ functions} \quad \textbf{-(1)}$$

Number of functions affected =

$$\sum_{F.I=o}^{F.I=n} F.I_{n_+}F.I_{n+1} \text{----(2)}$$

The output of equation 2 is the initial population which is given as input to algorithm to achieved best value of functional importance.[22].

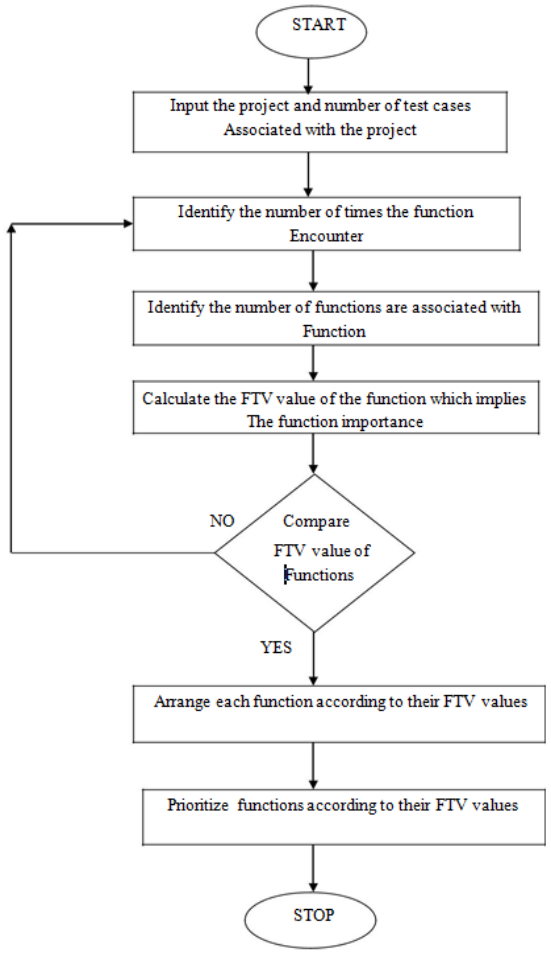

Figure 1. Proposed Flowchart

5.1. Pseudo Code of Proposed Algorithm

Input : Test cases=P(i)

Number clicks on each function =F(i)
Output: prioritized testcases
I<-Consider value of F(i) for the each test case
Test case F(i) value <- i
while (fault value of each test case is calculated)

a=F(i)
calculate number of links L(i)=F(i)'/F(i)
if(L(i)>L(i+1)
b=L(i)
else

b=L(i)
end

Calculate fault value Fault (i+1)=fault(i)/L(i)
if Fault(i) > Fault(i+1)
best_so_far <-Fault(i)

i <- generate an individual randomly end

VI. EXPERIMENTAL RESULTS

To analyze performance of proposed and existing systems dataset is considered which gave attributes defined in Table 1.

Table 1. Dataset Attributes

Number of projects	10
Number of changes	4
Function association	Yes
Functional faults	YES

The MATLAB is used to performance experimental results of proposed and existing algorithm and performance is measured in terms of fault detection rate.

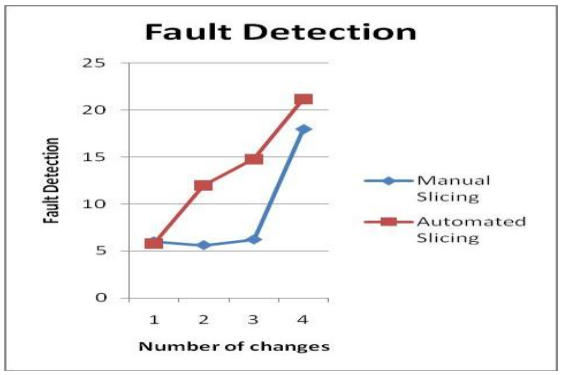

Figure 2. Fault Detection Comparison

As shown in Figure 2, the fault detection of manual slicing and automated slicing is compared and it is been analyzed that with the use of automated slicing fault detection rate in improved regression testing.

VII. CONCLUSION

In this work, it is been concluded that test case prioritization is applied to prioritize the test cases when certain changes as been done in the project. In this work, the test cases are prioritized according to their functional importance which is calculated using automated slicing. The simulation is performed in MATLAB and it is been analyzed that proposed technique performs batter in terms of fault detection. In future, proposed algorithm can be compared with some other algorithms of test case prioritization to check reliability. The paper is formulated to solve the two objective test case problems. The first objective is that to cover maximum fault from list and the second is to take less time to find fault for intention to minimize the effort and cost.

REFERENCES

[1] H. Leung and L. White, "Insight into regression testing", In Proceeding 27th IEEE International Conference Software Engineering, vol. 20, (1989), pp. 60-69.

[2] S. Khan and A. Awais, "TestFilter: A Statement-Coverage based test case reduction technique", In Proceeding 12th IEEE International Conference Engineering, vol. 11, (2006), pp. 5-12.

[3] Z. Harman and R. Hierons, "Search algorithms for regression test case prioritization", In Proceedings 12th IEEE International Journal Software Engineering, vol. 33, (2007), pp. 225-237.

[4] S. Khan and A. Nadeem, "TestFilter: A Statement-Coverage Based Test Case Reduction Technique", IEEE Conference on multitopic, (INMC' 06), (2009), pp. 275-280.

[5] J. Daengdej, "Test case prioritization techniques", Proceedings of IEEE International Journal Software Engineering Knowledge Engineering, vol. 22, (2010), pp. 161-183.

[6] S. Mirarab, "The effect of time constraint on test case prioritization", IEEE Transaction on Software Engineering, vol. 36, no. 7, (2010), pp. 85-91.

[7] R. Malhotra, A. Kaur and Y. Singh, "A Regression Test Selection and Prioritization Technique", Journal of Information Processing Systems, vol. 6, no.2, (2010), pp.167-171.

[8] E. Engstrom and P. Runeson, "A Qualitative Survey of Regression Testing Practices", Springer-Verlag Berlin heidelbreg, LNCS 6156, (2010), pp. 3-16.

[9] D. Hyunsook and S. Mirarab, "The effect of time constraint on test case prioritization", IEEE Transaction on Software Engineering, vol. 36, (2010), pp. 145-151.

[10] H. Srikanthi and J. Williams, "System test case prioritization of new regression test case", IEEE Transaction on Software Engineering, vol. 36, no. 2, (2011), pp. 87-94.

[11] A. Kaur and S. Goyal, "A genetic algorithm for regression test case prioritization using code coverage", International journal on computer science and engineering 3.5, (2011), pp. 1839-1847.

[12] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey", International journal on computer science and engineering, (2012), pp. 67-120.

[13] J. Hwang, "Selection of regression system tests for security policy evolution", Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ACM, (2012), pp. 69-74.

[14] X. Zhang and G. Uma, "Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm", European Journal of Scientific Research, vol. 74, (2012), pp. 34-37.

[15] H. Mei, D. Hao, L. Zhang, L. Zhang, J. Zhou and G. Rothermel, "A Static Approach to Prioritizing Junit Test Cases", IEEE Transactions on Software Engineering, vol. 38, no. 6, (2012), pp. 1258-1275.

[16] S. Yoo and M. Harman, "Regression Testing Minimisation, Selection and Prioritisation: A Survey", Software Testing, Verification and Reliability, vol. 22, no. 2, (2012), pp. 67-120.

[17] A. Jatain and G. Sharma, "A systematic review of techniques for Test case prioritization", International Journal of Computer Applications, vol. 68, (2013), pp. 132-135.

[18] M. Athar and L. Ahmad, "Maximize the Code Coverage for Test Suit by Genetic Algorithm", International Journal of Computer Science and Information Technologies, vol. 5, (2014), pp. 431-435.

[19] P. Konsaard and L. Ramingwong, "Total Coverage Based Regression Test Case Prioritization using Genetic Algorithm", Proceeding IEEE International Journal Software Engineering Knowledge Engineering, vol. 24, no. 5, (2015), pp. 24-31.

[20] H. Wang, J. Xing and Q. Yang Q, "Modification Impact Analysis based Test Case Prioritization for Regression Testing of Service-Oriented Workflow Applications", in Proceedings 39th IEEE International Journal Software Engineering Knowledge Engineering, vol. 30, no. 8, (2015), pp. 67-72.

[21] Saloni Ghai and Sarabjit Kaur, "A Hill-Climbing Approach for Test Case Prioritization" International Journal of Software Engineering and Its Applications,Vol. 11, No. 3 (2017), pp. 13-20

[22] Riza Dhiman and Dr Vinay Chopra, "Modified ACO model for Regression Testing Using Automated Slicing "Journal of Emerging Technologies and Innovative Research,Vol. 5, Issue 6 (June 2018), pp. 180.

