Ants Optimization for Minimal Test Case Selection and Prioritization as to Reduce the Cost of Regression Testing

Neha Sethi
M.TECH, Student
GZSCET, PTU Campus
Bathinda, India

Shaveta Rani, PhD
Assistant Professor
GZSCET, PTU Campus
Bathinda, India

Paramjeet Singh, PhD
Assistant Professor
GZSCET, PTU Campus
Bathinda, India

ABSTRACT
Software testing is the major process in software development life cycle. Regression testing is very costly and inevitable activity that is to be performed in a restricted environment to ensure the validity of modified software. It is inefficient to re-run every test case from test suite when some kind of modification is done in the software. Test case selection and prioritization techniques select and organize the test cases in a test suite based on some criteria such that the faults are covered quickly with minimum execution time. This task can be done on basis of the Ant Colony Optimization technique (ACO) of Swarm Intelligence as it is not deeply studied yet. The main objective of this thesis is to solve the path problem: Means to find the shortest path and Resolve the time problem: Means to minimize the time of finding shortest path. Because of time and cost constraint, it is not possible to perform extensive regression testing. Techniques such as test case selection and prioritization are used to solve the problem of time and cost constraints. In this paper we are modifying the previous technique to get better results in case of execution time and then the Effectiveness of techniques is checked with the help of APFD metric.

General Terms
Algorithms, Test Case, Software Testing, Prioritization, APFD

Keywords
ACO, Pheromone, Regression Testing, Test Case Selection, Test Case Prioritization.

1. INTRODUCTION
When the software is designed then the software development life cycle is used to develop the complete software. For the software development, different phases are followed. After the designing phase testing phase is used to test the whole design code. There are different types of testing that are used to test the code. The regression testing is one of the types of the testing that is used to test the software code.

The maintenance phase of a software product needs to go through the inevitable regression testing process. It is required to retest the presented test suite whenever any accumulation, removal or modification is made to the software [3]. During the testing process, different test cases are generated. Regression test selection is a process of reducing the test suite by selecting a subset from the original test matching set. Although this is a extremely cost efficient method for regression testing but it can leave out certain important test cases from the selected subset of test cases. Regression test prioritization means arrangement of the test cases in an increasing or decreasing order to meet some performance goal. Various prioritization criteria may be applied to the regression test suite with the objective of meeting those criteria [17].

In the test case selection and prioritization, different techniques are used to generate the test cases. There are BCO and ACO two techniques that are used to generate the test cases selection and prioritization [7]. ACO is the Ant colony optimization technique that is a set of commands, based on look for algorithms of artificial intelligence for optimal solutions; here the iconic member is ANT System. Ants are unsighted and small in size and still are able to find the shortest route to their food resource. They make the use of antennas and pheromone liquid to be in contact with each other. ACO deals with two significant processes, namely: Pheromone deposition and trail pheromone disappearance. Pheromone deposition is the phenomenon of ants adding the pheromone on all paths they follow. Pheromone trail evaporation means diminishing the amount of pheromone deposited on every path with respect to time. ACO motivated from the behavior of live ants, are capable of bringing together with searching solutions for local problem by maintaining array list to maintaining previous information gathered by each ant [1].

2. RELATED WORK
A handful of researchers have been presented in the literature for the prioritization of test cases. A brief review of some researchers is presented below:

Minjie Yi proposed an algorithm for path oriented testing by combining two algorithms namely: ant colony system algorithm and genetic algorithm (ACSGA) and the proposed algorithm are used to generate path oriented test data. The problem used is Triangle discrimination problem. The result of the proposed algorithm is then compared with the Genetic Algorithm and it is found that the proposed algorithm is more efficient than genetic algorithm [23].

Nirmal Kumar Gupta, Mukesh Kumar Rohil, in their work, they have proposed a strategy that uses genetic algorithm to reduce the number of unfeasible test cases and establishing genetic algorithm to generate suitable test cases and hence improving the genetic algorithm .The genetic algorithm is used to generate the test cases for Object Oriented Software ,that is for java programs. But practically to use genetic

algorithm there is a problem as there is local convergence problem [24].

K. Karnavel et.al used the Bee Colony Optimization (BCO) algorithm and performed regression testing and prioritization. The fault coverage is incorporated with the BCO algorithm for fault revealing within less execution time. Using this approach to test the software application, it is found that the testing cost and the execution time have been reduced [7].

Daniel Di Nardo et.al discussed an industrial case study on the Coverage-Based Test Case Prioritization with real regression faults. Because of the time or resource constraints Retest-All' strategy to perform the regression testing is time consuming and not efficient with the growing number of test. Therefore, strategies for regression testing such as test suite minimization, test case selection and test case prioritization will block and decision are better than the other mentioned coverage-based prioritization techniques. It has provided answers to many research questions such as: How granularity of coverage criteria when used in the analysis for test case prioritization affects the rate of fault detection? Etc. Four different prioritization techniques are evaluated .The coverage-based prioritization techniques are namely: Total Coverage, Additional Coverage, Total Coverage of Modified Code, and Additional Coverage of Modified Code. The result is that the additional coverage with finer grained coverage criteria such as block, fundamental [8].

Chengying Mao et.al have reformed the basic Ant Colony Optimization to generate better quality test data for the purpose of earlier fault- revealing .Four rules have been re-defined to achieve the objective namely: the local transfer rule, the global transfer rule and pheromone update rule. The criterion used is the branch coverage. And a comparison analysis has been made among GA, SA and ACO .To validate the method, experimental analysis is carried out by taking five programs. The result of the reformed Ant Colony Optimization outperforms the algorithms namely: Simulated Annealing and genetic algorithm [4].

Praveen Ranjan Srivastava presents a new test case prioritization algorithm, which calculates average faults found per minute and effectiveness of algorithm can be found with the help of APFD metric. Analysis is done for both prioritized and non-prioritized cases with the help of APFD (Average Percentage Fault Detection) metric and results prove that prioritized case is more effective [15].

Ruchika Malhotra et al. proposed a technique based on the version specific test case prioritization where information about changes in the program is known. The technique identifies those test cases that execute the modifies lines of source code at least once and execute the lines of source code after deletion of deleted lines from the execution history of the test case and that are not redundant. The proposed technique uses two algorithms modification and deletion. Software testers can use this technique and reduce the cost of regression testing significantly [16].

Suri Bharti et al. ACO is a promising technique for solving test case selection and prioritization problem. In this study a tool ACO_TCSP for the same has been developed and applied on an example. Though in these tests the best solution was not found for all cases still the results obtained are in close proximity to the optimal results. The reduction in test suite size is achieved to be 62.5% in all the 4 test runs. This encourages the use of the developed tool by testers. In future we plan to apply the tool on many more examples to prove the usability and effectiveness of the proposed technique [3].

A.Pravin et al. proposed that priority is given to the test cases based on the code coverage and improve the testing process by finding the faults earlier. For each test case a value is generated depending upon certain factors then comparison of test cases is done and priority is assigned to the test cases based on their values. Algorithm was compared with the random prioritization technique on two application projects and describes the effectiveness [13].

Gurinder Singh et al. have proposed test case selection approach from a large test suite using hybrid technique based on genetic algorithms and Ant colony optimizations. The technique developed using this approach identifies and reduces the test data. The approach provides better results in the initial iteration of the whole process. It provides positive feedback and hence it can lead to better solutions in optimum time [17].

Pradipta Kumar Mishra et al. proposed a new test case prioritization technique using genetic algorithm. The proposed technique separate the test case detected as severe by customer and among the rest test case prioritizes subsequences of the original test suite so that the new suite, which is to run within a time-constrained execution environment. It will have a superior rate of fault detection when compared to rates of randomly prioritized test suites. This experiment analyzes the genetic algorithm with regard to effectiveness and time overhead by utilizing structurally-based criterion to prioritize test cases [18].

3. RESEARCH METHODOLOGY

3.1 Strategy

On deciding the broad area, literature survey on various websites and latest papers of software testing and algorithms have been done. In accordance with the information gathered from the survey, the algorithm ACO is chosen for test case selection and prioritization. With the following reasons ACO has been chosen:

1. In real time the ACO can adapt to changes and it is better than Simulated Annealing and Genetic Algorithm for problems like Travelling Salesman Problem where the graph changes dynamically [22].
2. ACO outperforms simulated annealing and genetic algorithm for the metrics namely average coverage (AC), successful rate (SR) and average convergence generation (AG) [22].
3. The limitation of Bee Colony Optimization is that it is complicated to map the waggle dance to the solution, it requires deep knowledge of mathematics and one must have the knowledge of factors such as waggle dance, food source quality etc. Premature convergence is the limitation of Particle Swarm Optimization (PSO) and GA [22].
4. ACO algorithms, has been largely applied in the field of management and industrial to obtain the optimal solution .But its application in the field of software testing for generating test data has not been deeply studied until today[22].

3.2 Steps for implementation

Ant Colony Optimization (ACO) algorithm will work as follows:

- Input details about the Test Suite.
- Run ACO algorithm for all ants.
- The best path will be chosen from each iteration based on:
  - ✓ Maximum fault coverage
  - ✓ Minimum Execution Time
- Formula to make pheromone is as follow:

$$\tau_{xy}^{k} = (1-\rho)\tau_{xy}^{k} + \Delta\tau_{xy}^{k}$$

Where,

τ_{xy}^{k} Is the capacity of pheromone to be deposit for a state transition xy

ρ Is the pheromone disappearance rate or pheromone decay

and

$\Delta\tau_{xy}^{k}$ is the total amount of pheromone deposit.

- Update the Pheromone
  - ✓ Deposit Pheromone on the best path(add)
  - ✓ Reduce Pheromone on the other edges.(evaporate)
- Terminate the search/iteration process when all Faults are Covered.
- Check whether all the test cases are considered or not.
- Determine the final path based on:
  - ✓ Minimum execution time with Maximum Fault detection.

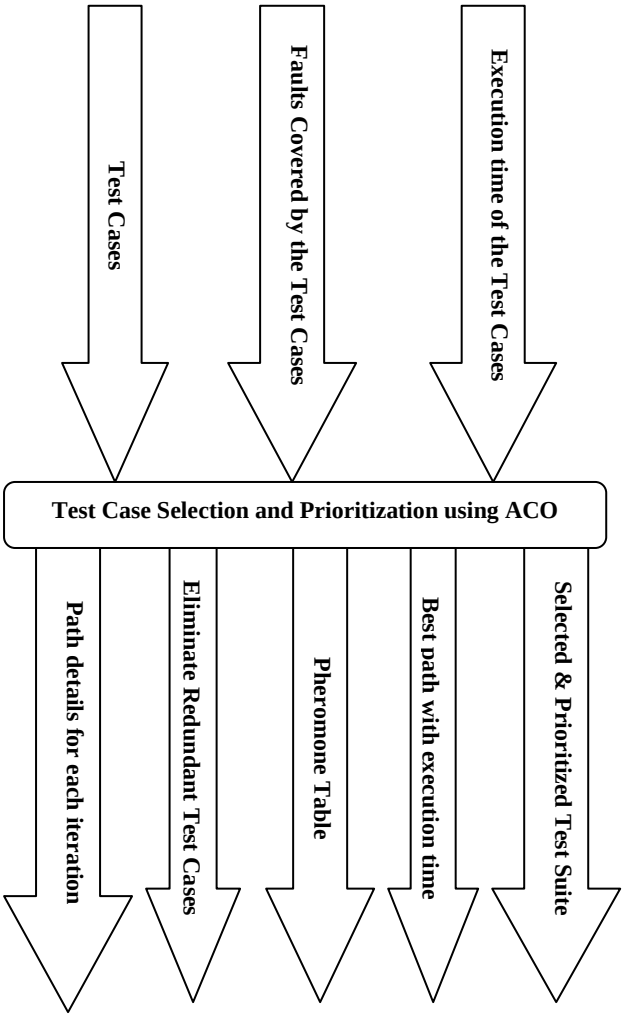

Fig 1: Input-Output of the ACO

4. FORMULATION USED

Consider the test case i and we have to find probability of test case j then previous formula for probability [1]:

$$Probablity_j = \frac{(Errorscovered_j)^{\beta} X\ (Phermononetrail_{i,j})^{\alpha}}{\sum_{k=1}^{n}((Errorscovered_k)^{\beta} X\ (Phermononetrail_{i,k})^{\alpha})}$$

where $j, k\epsilon\ \{Non\ visited\ test\ cases\}$

Now modified formula used by us is:

$$Probablity_j = \frac{(\frac{Errorscovered_j}{ExecutionTime_j})^{\beta} X\ (Phermononetrail_{i,j})^{\alpha}}{\sum_{k=1}^{n}(\left(\frac{Errorscovered_j}{ExecutionTime_k}\right)^{\beta} X\ (Phermononetrail_{i,k})^{\alpha})}$$

where $j, k\epsilon\ \{Non\ visited\ test\ cases\}$

In order to select next test case above modified formula is used. A new prioritization technique is make based on the above formula, test case which has the highest probability according to the modified formula is chosen. Now, both the number of faults covered as well execution time of each test case is considered and the results obtained shown that the total execution time of all selected test cases from the prioritized test suite is less as compared to result obtained from the previous formula.

5. FLOWCHART

Figure 2 shows the process chart of ACO algorithm. Initially read the information from the input file that describes the number of iterations used, number of test cases used, their execution time and faults covered by them.

After this, ants select test cases randomly because initially each test case has the same probability. Once a test case is selected, check if it covers all faults if yes, read the execution time of that test case otherwise, if all faults are not covered, then select another test case that has the highest probability by using the above described modified formula and update the pheromone amount at each edge in the pheromone table and stores the execution time of selected test cases. Then the same procedure is repeated for all iterations.

Explore all the paths in each iteration and then the best path from all the explored paths is selected. The best path is that one which covers all the faults with minimum execution time. Various test cases in the test suite are selected and prioritized according to the modified probability formula and thus the cost of regression testing will be reduced.

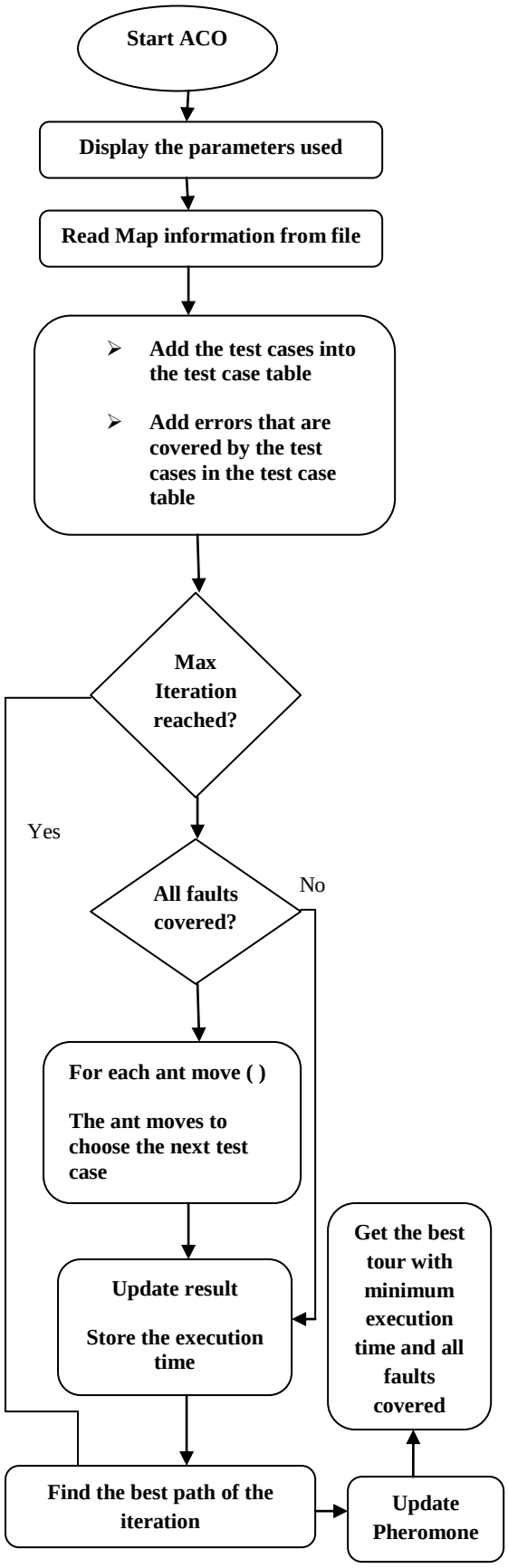

Fig. 2 Flowchart for the modified algorithm

6. RESULTS AND DISCUSSION

While Running:

At first we will initialize the parameters as given below:-

- ✓ Number of Ants
- ✓ Number of test cases used
- ✓ Total errors to find
- ✓ Number of iterations
- ✓ Pheromone deposition factor
- ✓ Pheromone evaporation factor
- ✓ Constant alpha
- ✓ Constant beta

Table 1: Input table used for 8 test cases and 10 faults

Test cases/Faults	*F 1*	*F 2*	*F 3*	*F 4*	*F 5*	*F 6*	*F 7*	*F 8*	*F 9*	*F 10*	*Number of faults covered*	*Execution time*
T1		*		*			*		*		*4*	*7*
T2	*		*								*2*	*4*
T3	*				*		*	*			*4*	*5*
T4		*		*					*		*3*	*4*
T5			*			*				*	*3*	*4*
T6	*						*				*2*	*4*
T7			*			*		*			*3*	*4*
T8		*								*	*2*	*2*

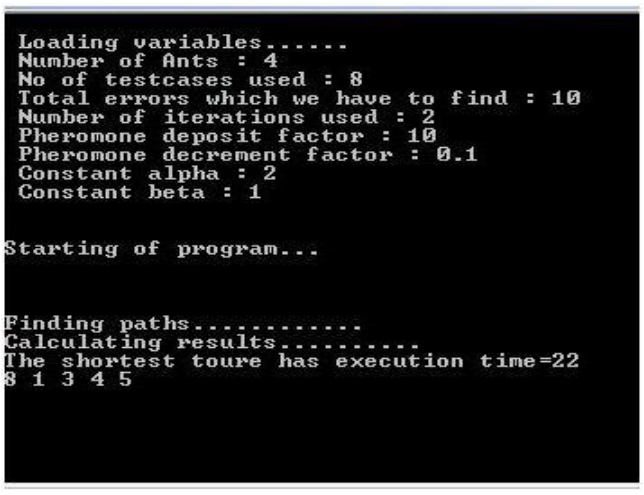

Fig. 3: Output screen of previous technique

It can be clearly seen from Figure 3 that execution time of selected test cases using previous technique is 22.

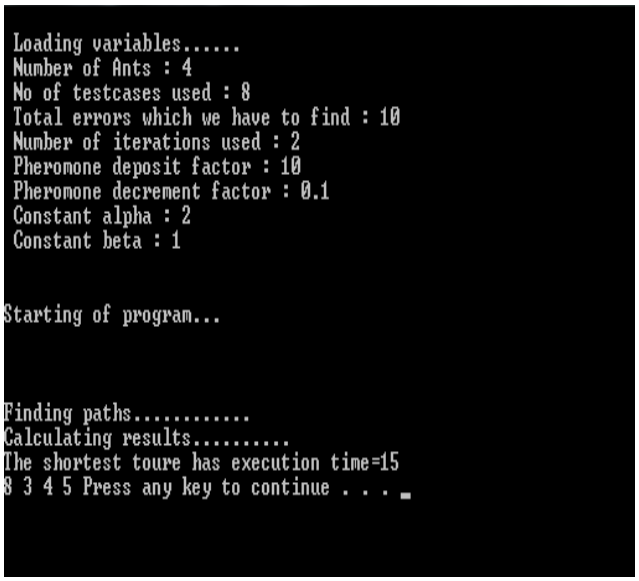

Fig. 4 Output screen of proposed technique

Above figure shows that execution time of selected test cases using modified technique is 15.

- Here test cases 8, 3, 4 and 5 needs to be executed on the basis of their corresponding priority.
- Total out of 8 test cases, we need to run only 4 test cases that will cover all the faults with minimum execution time. Thus, there will be 50% reduction of test cases in the given test suite.

The proposed method is compared with the equivalent traditional ACO for test case selection and prioritization approach. Putting the same parameters into the algorithm and changing the formula to calculate the probability for visiting the unvisited test cases and also to update the length of tour according to the approach. It can be seen that the execution time of selected test cases by the modified probability formula is less as compared to previous one.

The graphical representation of comparison of both techniques is shown in figure below:

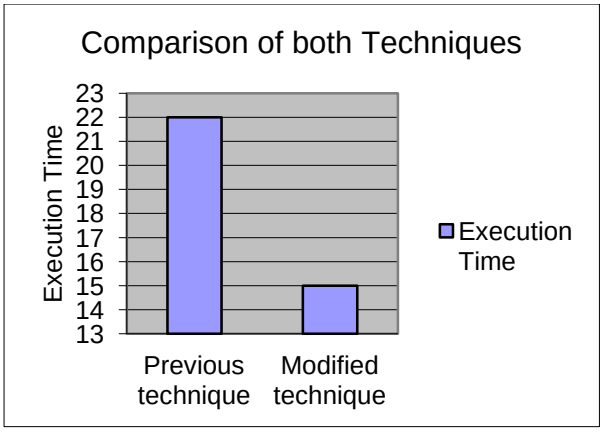

Fig. 5: Graphical comparison of both techniques

7. DETERMINING EFFECTIVENESS OF PROPOSED TECHNIQUE

There are many of the possible goals for prioritization. Our main focus is on increasing the likelihood of revealing faults as earlier as possible in the testing process. By this goal one can informally improve the test suite's rate of fault detection. And to quantify this goal, we will use a metric:

APFD (Average Percentage Faults Detection), introduced by Rothermel et.al. [10]

APFD helps to measures the weighted average of the percentage of faults detected over the life of the suite. The APFD values range from 0 to 100; the higher the number, the faster (better) is the fault detection rates.

The APFD for test suite *T* is given by the equation:
Let T - The test suite under evaluation program under test P.
n - Total number of test cases and m- total number of faults covered by test suite.
TFi - The position of the first test in T that exposes fault i.

APFD = 1-(TF1+TF2+……+TFm)/nm + 1/2*n

As the formula for APFD shows that calculating APFD is only possible when prior knowledge of faults is available.

APFD value for previous technique is calculated below:
Above table 1 is used as input table. Figure 3.shows that the selected test cases are T8, T1, T3, T4, and T5.

Table 2. After the selection of test cases by using previous technique

	T8	T1	T3	T4	T5
F1			*		
F2	*	*		*	
F3					*
F4		*		*	
F5			*		
F6					*
F7		*	*		
F8			*		
F9		*		*	
F10	*				*

APFD= 1- (24/80) + (1/16)
= 0.76

Table 3. After the selection of test cases by using modified technique

	T8	T3	T4	T5
F1		*		
F2	*		*	
F3				*
F4			*	
F5		*		
F6				*
F7		*		
F8		*		
F9			*	
F10	*			*

APFD= 1- (12/80) + (1/16)
= 0.91

It can be clearly seen from the above calculations that the valve of APFD metric for the proposed technique is greater as compared to previous one. So, the proposed technique is more effective.

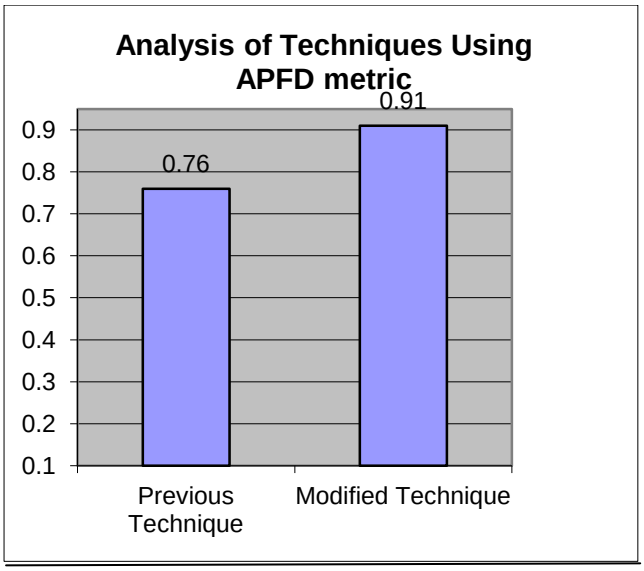

Fig. 6: Comparative analysis of both techniques

7. CONCLUSION AND FUTURE WORK

In this paper we have worked on Ant Colony optimization. This approach of test case selection and prioritization uses fault coverage and execution time allowing tester to prioritize the test case without considering number of lines covered by test case which may contain extra comment line. We have modified the existing technique and as we can see screen shots and comparison graph, it is very clear that modified technique gives better results and more effective than previous technique. In future more work can be done on other applications of Ant colony optimization. Moreover, further modifications can be done in existing techniques so as to get better results.

8. ACKNOWLEDGEMENTS

From the enormous of my heart, I thank my mentors for their continuous support and for their patience, motivation, enthusiasm, and for their immense knowledge. Without their valuable suggestions, it is not possible for me to carry out with the study.

9. REFERENCES

[1] M. Dorigo and C. Blum (2005), "Ant colony optimization theory: A survey", Theoretical Computer Science, vol. 344, no. 2-3, pp. 243-278.

[2] Osman Gokalp and Aybars Ugur (2012), "Improving Performance of ACO Algorithms Using Crossover Mechanism Based on Mean of Pheromone Tables", 2012 International Symposium on Innovations in Intelligent Systems and Applications (INISTA), Trabzon, pp. 1-4

[3] Bharti Suri and Shweta Singhal (2012), "Literature Survey of Ant Colony Optimization in Software Testing" , 2012 CSI Sixth International Conference on Software Engineering(CONSEG),Indore,pp1-7

[4] Chengying Mao, YuXinxin, Chen Jifu (2012)"Generating Test Data for Structural Testing Based on Ant Colony Optimization "12th International Conference on Quality Software, Xi'an, Shaanxi, pp. 98 – 101.

[5] Priyanka Bansal (2013), "A Critical Review on Test Case Prioritization and Optimization using Soft Computing Techniques", 2nd International Conference on Role of Technology in Nation Building (ICRTNB-2013), pp74-77.

[6] M.Dorigo, V.Maniezzo, and A.Colorni (1996), "Ant System: Optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man and Cybernetics, vol. B (26), pp. 29-41.

[7] K.Karnavel, J. (2013),"Automated Software Testing for Application Maintenance by using Bee Colony Optimization algorithms (BCO)" Information Communication and Embedded Systems, Chennai pp. 327-330.

[8] Daniel Di Nardo, N. A. (2013)," Coverage-Based Test Case Prioritization: An Industrial Case Study", IEEE Sixth International Conference on Software Testing, Verification and Validation, Luembourg. pp. 302-311.

[9] Luciano S. de Souza, P. B. (2011),"A Multi-Objective Particle Swarm Optimization for Test Case Selection Based on Functional Requirements Coverage and Execution Effort", 23rd IEEE International Conference on Tools with Artificial Intelligence, Boca Raton, FL ,pp. 245 - 252.

[10] Rothermal, Roland H. Untch, Chengyun Chu and Mary Jean Harrold (2001): "Prioritizing Test Case for Regression Testing", IEEE Transactions on Software Engineering.

[11] Rui Ding, X. F (2012)," Automatic Generation of Software Test Data Based on Hybrid Particle Swarm Genetic Algorithm", IEEE Symposium on Electrical & Electronics Engineering, Kuala Lumpur, pp. 670-673.

[12] Wang Jun, Z. Y. (2011)," Test Case Prioritization Technique based on Genetic Algorithm", International Conference on Internet Computing and Information Services, Hong Kong, pp. 173 - 175

[13] A.Pravin and Dr. S.Srinivasan(2013): " An Efficient Algorithm for reducing the test cases which is used for performing regression testing", 2nd International Conference on Computational Techniques and Artificial Intelligence, Dubai (UAE), pp. 194-197.

[14]Ashima Singh (2012): "Prioritizing Test Cases in Regression Testing using Fault Based Analysis", International Journal of Computer Science, vol. 9, Issue 6, pp. 414-420.

[15] Praveen Ranjan Srivastava (2008): "Test Case Prioritization", Journal of Theoretical & Applied Information Technology, pp. 178-181.

[16] Ruchika Malhotra, Arvinder Kaur and Yogesh Singh (2010): "A Regression Test Selection and Prioritization Technique", Journal of Information Processing Systems, vol.6, pp. 235-252.

[17] Gurinder Singh and Dinesh Gupta (2013). : "An Integrated approach to Test Suite Selection using ACO and Genetic algorithm", International Journal of Advanced Research in Computer Science & Software Engineering, vol.3, Issue 6, pp. 1770-1778.

[18] Pradipta Kumar Mishra and B.K.S.S Pattanaik (2013), "Analysis of Test Case Prioritization in Regression Testing using Genetic Algorithm", International Journal of Computer Applications, vol. 75, pp.1-10.

[19] A.Pravin and S.Srinivasan (2013), "Effective Test Case Selection and Prioritization in Regression Testing", Journal of Computer Science, pp. 654-659.

[20] K.K.Aggarwal and Yogesh Singh (2005), "Software Engineering Programs Documentation, Operating Procedures", New Age International Publishers, Revised Second Edition.

[21] Shaveta Malik (2010), "Performance Comparison between Ant Algorithm and Modified Ant Algorithm", International Journal of Computer Science and Applications, vol. 1, No. 4, pp. 42-45.

[22] Kevilienuo Kire and Neha Malhotra (2014),"Study of test case selection and prioritization", International journal of computer applications vol. 85-No. 5, pp.28-30.

[23] Yi Minjie (2012),"The Research of path-oriented test data generation based on a mixed ant colony system algorithm and genetic algorithm", International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), Shanghai, pp.1-4.

[24] Gupta Nirmal Kumar and Rohil Mukesh Kumar (2013) "Improving GA based Automated Test Data Generation Technique for Object Oriented Software", IEEE International Advance Computing Conference (IACC), Ghaziabad, pp.249 - 253.