ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 9(30), DOI: 10.17485/ijst/2016/v9i30/86423, August 2016*

A Comparative Evaluation of "m-ACO" Technique for Test Suite Prioritization

Kamna Solanki*, YudhVir Singh and Sandeep Dalal

M. D. University, Rohtak - 124001, Haryana, India

Abstract

Objectives: The novel test case prioritization technique "m-ACO" ("Modified Ant Colony Optimization") for regression testing has been comparatively evaluated. **Methods**: "m-ACO" prioritize the test cases by altering the food source selection criteria of natural ants to enhance fault diversity. The code for the proposed technique for prioritizing test case "m-ACO" has been implemented in Perl language. This paper makes a comparative evaluation of proposed "m-ACO" technique for prioritization of test cases with GA ("Genetic Algorithm"), BCO ("Bee Colony Optimization") Algorithms and ACO ("Ant Colony Optimization") Algorithms using three case studies. Two metrics namely APFD ("Average Percentage of Faults Detected") and PTR ("Percentage of Test Suite Required for Complete Fault Coverage") have been used to measure the effectiveness of the proposed "m-ACO" technique. **Findings**: The proposed technique "m-ACO" produced optimal or near optimal solutions. The proposed "m-ACO" technique proves its efficiency in comparison to GA, BCO and ACO methods individually. **Improvements**: The proposed technique improves the ACO method by altering food source selection criteria of natural ants. The future work in this direction will comparatively evaluate the proposed "m-ACO" technique using some well known software testing problems and open source software. An automated tool for the proposed technique is being developed.

Keywords: Fault Coverage, Genetic Algorithm, Regression Testing, Software Testing, Test Suite Prioritization

1. Introduction

Verification and validation activities are conducted throughout the entire life cycle of software development to enhance and evaluate the software quality. Verification and validation makes sure that the entire software system as a whole works as defined in software requirement specification and satisfies the customer's needs and requirements. Verification and validation is a commonly used term that actually refers to software testing. The choice of a software testing technique highly affects the quality of the software. So, the need of the hour is to use the most efficient and effective testing technique which can reveal maximum faults within resource constraints like time, efforts and cost. Software testers carefully design test suite (or test ensemble) for software testing to reveal maximum faults. Software has to be re-tested quite frequently as software code keeps on changing when revealed faults are fixed by the developers. The re-testing of the software code is extremely important to uncover any undesired effect of the amended software code on working of rest of the code. This re-testing of the software is known as regression testing. The original test suite is re-utilized during regression testing. It is inseparable step in the software development and, in conjunction with other protocols, it can influence around half of the cost for proper software maintenance[1–3]. However, due to monetary and temporal constraints, the complete ensemble of tests cannot be re-utilized. Hence, software testing now uses an approach based on selective analysis of code

**Author for correspondence*

modules having highest priority as per code coverage rate, fault detection rate, order of failure propensity or order of expected rate of employment. These approaches are jointly known as "Test case prioritization techniques". Recently, a novel technique "m-ACO" for prioritization of test cases has been proposed for regression testing[4]. This paper presents a comparative evaluation of the proposed "m-ACO" technique with existing techniques like Genetic Algorithm, Bee Colony Optimization and Ant Colony Optimization Algorithms for prioritization of test cases on two parameters namely APFD ("Average Percentage of Faults Detected") and PTR ("Percentage of Test Suite Required for Complete Fault Coverage").

Numerous researchers in the field of software testing have tried to optimize the prioritization conundrum and proposed many techniques. Some of the techniques in this direction were "Total fault-detection technique" which targeted to uncover all the defects or faults present in particular code modules[5]. The main approaches to enhance the rate of fault detection referred to "Greedy algorithms", which uses a greedy approach for the selection of the test cases i.e., prioritization of the most optimum initially[6]; "Evolutionary algorithms" which uses a progressive kind of evolution among various combination of test sub-ensemble to eventually create a prioritized test suite[7]; "Non-evolutionary algorithms" which follows a goal based prioritization[8,9]. "Need specific algorithms" have been formulated to satisfy specific needs of the prioritization[10]. These were not same as the general test suite prioritization methods, which can be applied on any type of prioritization problem. Another method for solving conundrum of prioritization is "Variable analysis algorithms" which considers the analysis of the relationship among variables which are modified and its utilization in other fields of code is perceived. Evolutionary Algorithms often performs well on most of the problems as they do not make any assumption about the underlying fitness landscape; however such types of algorithms have higher computational complexity and generally lack in clear distinction of genotype-phenotype. Greedy algorithms are most suitable only for those problems which possess 'optimal substructure'. Such algorithms mostly (but not always) shows failure in finding the global optimal solutions as they generally do not exhaustively operate on complete data. Greedy Algorithms make early commitments for certain choices which may prevent these algorithms from finding the most optimized solution later.

Natural processes have always inspired people for development of similar algorithms. These algorithms have already been employed for optimization of the fault detection among software modules and to improve the efficiency of software testing. A number of algorithms have been developed to identify prioritized test sets for a given problem. The approaches adopted in these solutions include statistical techniques; evolutionary approaches, such as GA (Genetic Algorithms); swarm based collective behavioural approaches. The last category includes algorithms such as BCO (Bee Colony Optimization) or ACO (Ant Colony Optimization). These work on the principle that collective swarm and detection approach can identify local and global solutions to the problems of test cases selection and prioritization. A novel technique "m-ACO" (Modified Ant Colony Optimization) has been proposed recently to prioritize the test cases in regression testing. The objective of the current work is to carry out one such analysis in order to understand the optimal application of proposed "m-ACO" technique.

Ant Colony Optimization (ACO) approach, which can be used to solve a variety of problems is a meta-heuristic technique[11]. Artificial ants have found applications in numerous applications, often delivering good results for problems like traffic management. Researchers have put their efforts towards solving the problem of test case prioritization by application of ACO algorithm; however they did not succeed and could only identify a near best solution[12–14]. None of the proposed techniques towards test suite prioritization using Ant Colony Optimization have ever discussed and utilized the concept of diversity of food captured by ants. Natural ants select every type of food source they come across, which decreases the diversity of food deposited. However, food diversity is an important factor in case of test suite prioritization as the diversity of faults captured by a particular test suite using ACO technique depends highly on the diversity of food captured by ants. In this regard, a novel test suite prioritization technique "m-ACO" (Modified Ant Colony Optimization) has been proposed by altering the food source selection criteria of natural ants to enhance the diversity of food captured. Enhanced diversity of food captured ultimately enhances the diversity of defects (faults) revealed by a prioritized test suite[15].

2. Comparative Evaluation of "m-ACO"

The "m-ACO" technique for which prioritize the given test suite measured the suitability of every node based on the optimal values of the code covered, number of faults detected and total time taken for test case execution. The modified ants approached the faulty modules in a pseudo-random order and evaluated its suitability. A pheromone factor was considered and quantified to attract many other ant like processes. The pheromone trails belonging to the code module that were most suitable finally became stronger in later rounds by making progressive coverage. The pheromone trail belonging to the less visited modules became weaker in later iterations at a constant rate. Ants can still move towards these trails which are progressively weakening; but the probability for the same is very low.

To conduct the experimental evaluation of "m-ACO", three case studies have been taken. These three case studies have been implemented in Perl language for "m-ACO", BCO, GA and ACO technique. The following parameters have been calculated for comparative evaluation of "m-ACO" technique against GA, BCO and ACO techniques for test case prioritization:

- APFD ("Average Percentage of Faults Detected").
- PTR ("Percentage of Test Suite Required for Complete Fault Coverage").

APFD metric basically deals with quantifying the goal of optimizing rate of fault detection by using various test suites combination[10,11]. This metric measures the average rate of fault (defect) detection rate per percentage of test suite execution. Higher value of APFD means higher percentage of faults detected. A comparatively high APFD means a better prioritization technique.

$$APFD = 1 - \frac{TF_1 + TF_2 + TF_3 + \cdots\cdots\cdots TF_m}{mn} + \frac{1}{2n}$$

Notion for APFD calculations are:

'T' refers to the test suite under Observation (evaluation), 'm' refers to total no. of defects (faults) in a system under test, 'n' refers to total number of test cases in any test suite and TF_j refers to the position of the first test case in T which uncovers fault j. PTR ("Percentage of Test Cases Required for Complete Fault Coverage") is a metric which can be used to calculate the effectiveness of the test suite prioritization technique[12]. An effective test suite prioritization will position the test cases which are most likely to find faults at the starting of the prioritized test sequence. So, it would be helpful to calculate the percentage of those test cases which must run before all faults of the application are revealed. A comparatively low value of PTR means a better prioritization technique.

Notion for PTR calculations are:

$$PTR = \frac{\text{No.Of Test Cases Required for Complete Fault Coverage}}{\text{Total Number of Test Case}} \times 100$$

To make the comparative analysis of the proposed "m-ACO" technique for test suite prioritization, three case studies[4] have been taken namely "Case Study 1-College Program for Admission in Courses", "Case Study 2-Library Management" and "Case Study 3-Hotel Reservation System". Case Study 1 has initially has 10 test cases with initial un-prioritized execution order "N1->N2->N3->N4> N5->N6->N7->N8->N9->N10" covering 10 faults. Case Study 2 has 5 test cases with initial un-prioritized execution order as "N1->N2->N3->N4->N5" covering 5 faults. Case Studty 3 has 9 test cases with initial un-prioritized execution order as "N1->N2->N3->N4->N5->N5->N6->N7->N8->N9" covering 5 faults.

2.1 Application of GA

GA is a search heuristic based on the process of natural selection which generate solutions of many optimization and search problems. GA uses the phenomenon of natural evolution techniques like inheritance, mutation, selection and crossover. Genetic Algorithm has found application in the area of software testing for test data generation and to solve test optimization problems[21–23]. The three case studies taken were executed using a test suite prioritization technique using Genetic Algorithm implemented in Perl language.

APFD values for three case studies: APFD values yielded for the above mentioned three Case Studies were 0.73 for case study1, 0.86 for Case Study 2 and 0.86 for Case Study 3 respectively as shown in Table 1.

Table 1. APFD values using GA

	Prioritization Order using GA	APFD Values
Case Study1	{N4->N3->N1->N9->N6->N2->N7->N10->N3->N8}	0.73
Case Study2	{N5->N1->N3->N4->N2}	0.86
Case Study3	{N1->N6->N8->N3->N4->N5->N9->N2->N7}	0.86

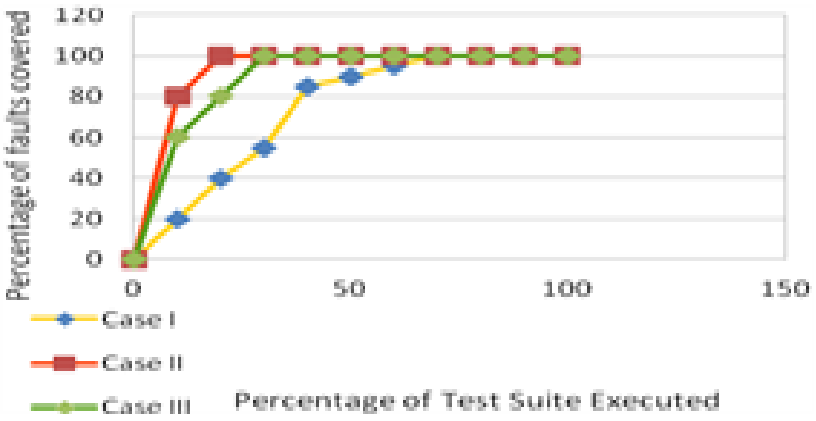

Figure 1. PTR Chart for all case studies using GA.

PTR values for three case studies: The following Figure 1 depicts that execution of 70% of test cases for case study1, 20% of test cases for Case Study 2 and 30% of test cases for case study3 in prioritized order generated using GA ensures complete fault coverage.

2.2 Application of BCO

The BCO is a nature inspired technique that follows the foraging behaviour in honeybees. The main objective of the BCO is using multi agent technology ("colony of artificial bees") for effectively solving many types of hard combinatorial optimization problems. BCO has a specific ability of finding high quality solutions within a reasonable amount of computer time for difficult combinatorial problems. The Bee Colony Algorithm is a stochastic search technique. Bee Colony Optimization has been applied successfully by researchers for test case optimization and prioritization[24–26]. So, the three case studies taken were executed using a test suite prioritization technique using Bee Colony Optimization Algorithm.

APFD values for three case studies: APFD values yielded for the above mentioned three Case Studies were 0.71 for case study1, 0.82 for Case Study 2 and 0.86 for Case Study 3 respectively as shown in Table 2.

PTR values for three case studies: The following Figure 2 depicts that execution of 60% of test cases for case study1, 20% of test cases for Case Study 2 and 45% of test cases for Case Study 3 in prioritized order generated using BCO ensures complete fault coverage.

Table 2. APFD values using BCO

	Prioritized Order using BCO	APFD Values
Case Study1	{N4->N3->N1->N2->N7->N6->N5->N8->N9->N10}	0.71
Case Study2	{N3-> N5-> N1-> N4-> N2}.	0.82
Case Study3	{N6->N1->N4->N8->N3->N2->N9->N5->N7}.	0.86

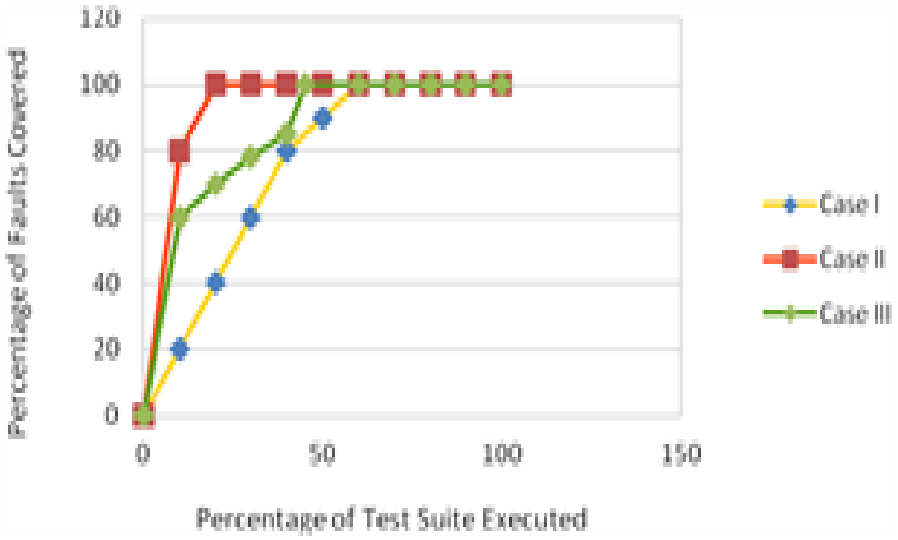

Figure 2. PTR Chart for all case studies using BCO.

2.3 Application of ACO

ACO is a nature inspired technique for solving combinatorial optimization problems. It uses food source searching pattern of natural ants to find the optimized path to reach to its food source. As discussed earlier, ACO has already been used for solving many types of optimization and prioritization problems.

APFD values for three case studies: When the three case studies were executed using ACO for test case prioritization, the APFD values yielded for the three Case Studies were 0.76, 0.82 and 0.88 respectively for three case studies considered as shown in Table 3.

PTR values for three case studies: As evident from following Figure 3, the execution of 50% of test cases for case study1, 20% of test cases for Case Study 2 and 30% of test cases for Case study3 in prioritized order generated using ACO ensures complete fault coverage.

Table 3. APFD values using ACO

	Prioritized Order using BCO	APFD Values
Case Study1	{N4->N2->N3->N7->N6->N1->N5->N9->N8->N10}	0.76
Case Study2	{N3-> N5-> N1-> N4-> N2}.	0.82
Case Study3	{N4->N3->N1->N6->N8->N2->N5->N7->N9}.	0.88

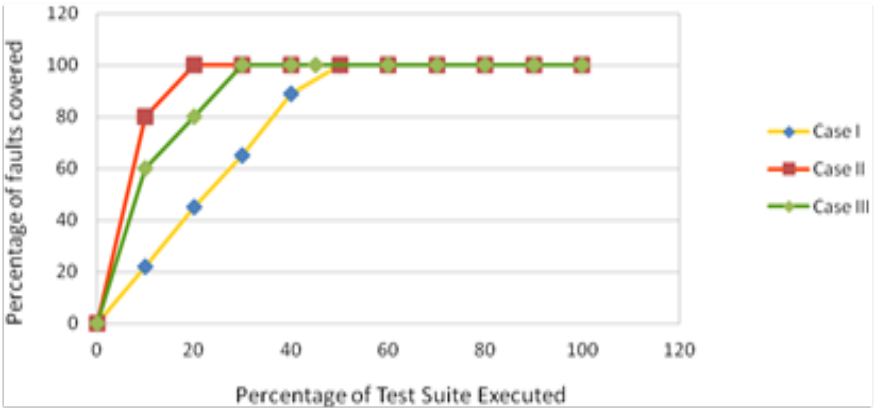

Figure 3. PTR chart for all case studies using ACO.

2.4 Application of "m-ACO"

The proposed "m-ACO" technique prioritizes the ensemble of test cases by changing the food source searching behavior of natural ants to enhance the diversity of food accumulated in its nest by evaluating the food fitness before selection. This helps in covering the diverse faults earlier in a prioritized test suite thereby reducing the time taken to cover the faults. The proposed "m-ACO" technique has been experimentally evaluated using the three case studies discussed earlier.

APFD values for three case studies: When the three case studies were executed using "m-ACO", the APFD values yielded for the three Case Studies were 0.81, 0.86 and 0.88 as shown in Table 4.

PTR values for three case studies: The following Figure 4 depicts that execution of 50% of test cases in prioritized order for Case Study 1, 20% test cases for Case Study 2 and 30% test cases for Case Study 3 ensures complete fault coverage.

Table 3. APFD values using m-ACO

	Prioritized Order using m-BCO	APFD Values
Case Study1	{N4->N2->N1->N7->N6->N9->N10->N5->N8->N3}	0.81
Case Study2	{N5-> N3-> N1-> N4-> N2}.	0.86
Case Study3	{N6->N1->N4->N8->N3->N2->N5->N9->N7}.	0.88

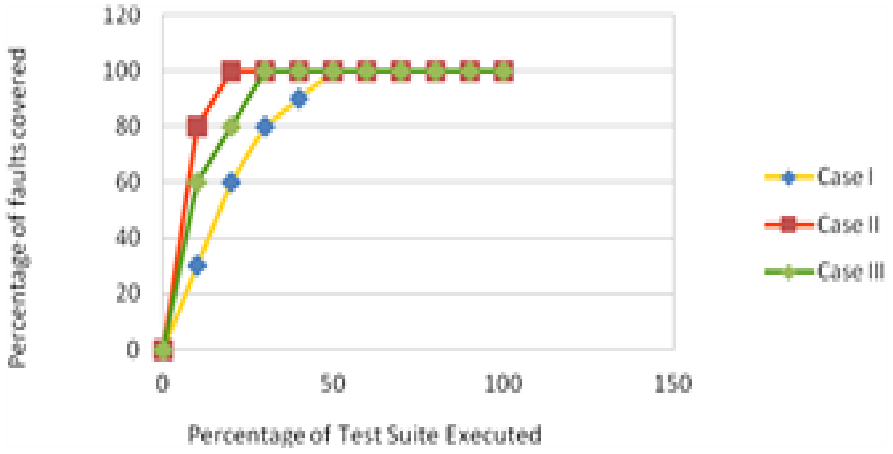

Figure 4. PTR chart for all case studies using m-ACO.

It can be clearly observed that the proposed "m-ACO" technique for prioritization of test suite either performs equally good or better than other contemporary meta-heuristic techniques based test suite prioritization techniques on two parameters i.e., APFD and PTR as shown in following Figure 5 and Figure 6. The food uniqueness fitness function of the proposed "m-ACO" technique works by selecting only the unique food so that unique faults are covered earlier by a prioritized test sequence which ultimately reduces the PTR values of the prioritized test suite and enhances the APFD values i.e., enhanced fault detection rate of the prioritized test sequence.

The efficiency and effectiveness of regression testing is determined by many factors including automation of test sequence generation[27], the percentage of original test suite required to achieve complete fault coverage and percentage of faults detected per unit time[28]. So, the proposed "m-ACO" technique for prioritizes the test suite and improves the effectiveness of regression testing by producing optimal or near optimal solutions.

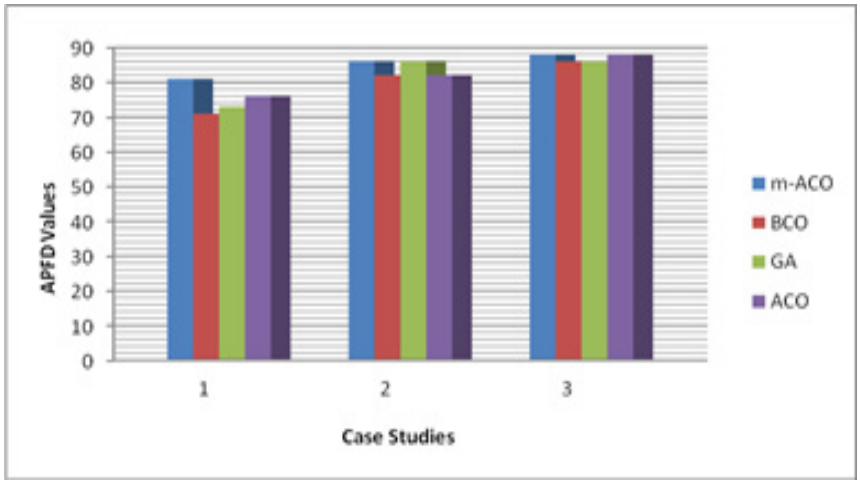

Figure 5. Comparative APFD values of m-ACO, BCO, GA and ACO.

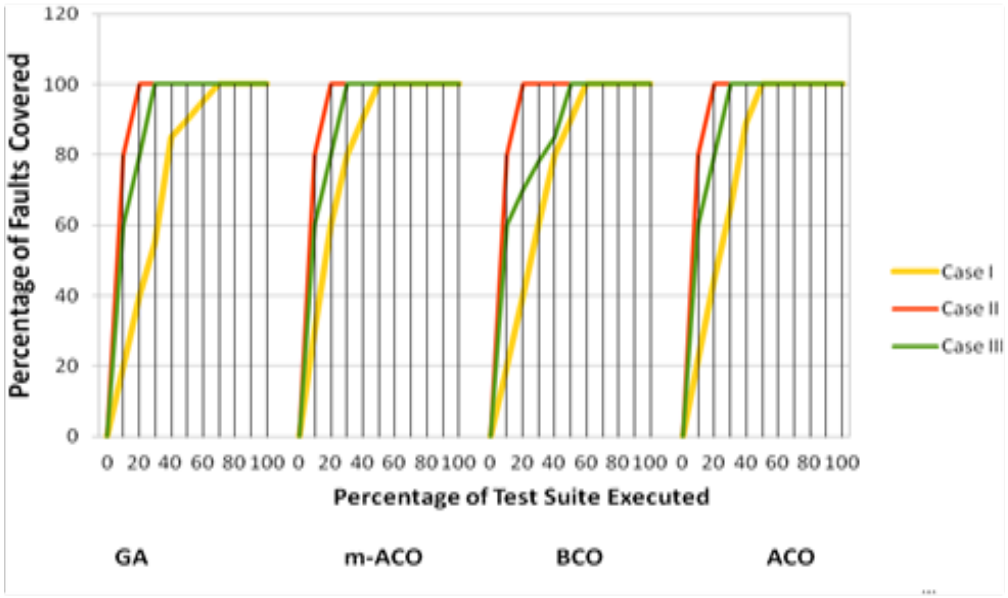

Figure 6. Comparative PTR values of GA, m-ACO, BCO and ACO.

3. Conclusion

Regression testing basically re-authenticates the older version of the functional software to avoid un-necessary side effects of the amendments in the software code. It is most crucial and time consuming testing activity which requires a lot of resources. Test Suite Prioritization is one of the most widely used technique for regression testing which enhances the fault detection rate of a test suite by re-scheduling the order of execution of test cases and

reduces testing efforts by reducing the fraction of test suite needed to achieve complete fault coverage (i.e., PTR values). The proposed algorithm "m-ACO" modifies the ACO algorithm and alters the food source selection criteria of natural ants for prioritizing test cases. This paper makes a comparative evaluation of the proposed "m-ACO" algorithm against GA, BCO and ACO based test case prioritization techniques using three case studies. The performance of the proposed "m-ACO" algorithm clearly demonstrates its power. The future work in this direction will try to comparatively evaluate the proposed "m-ACO" technique against some other techniques using well known software testing problems as well as open source software problems. The application of "m-ACO" technique can be of utmost importance for boosting the effectiveness and efficiency of a test suite.

4. References

1. Onoma K, Tsai WT, Poonawala M, Suganuma H. Regression testing in an industrial environment. Communications of the ACM. 1998 May; 41(5):81–6.
2. Beizer B. Software testing techniques. 2nd ed. India: Dreamtech Press; 2003.
3. Leung H, White L. Insights into regression testing. Proceedings of the IEEE International Conference on Software Maintenance; 1989 Oct. p. 60–9.
4. Solanki K, Singh Y, Dalal S. Test case prioritization: An approach based on modified ant colony optimization. Proceedings of IEEE International Conference on Computer, Communication and Control; Indore, India. 2015 Sep. Available at IEEE-xplore Digital Library.
5. Rothermel GU, Chu C, Harrold MJ. Test case prioritization: An empirical study. Proceedings of the International Conference on Software Maintenance; Oxford, UK. 1999. p. 179–88.
6. Li Z, Harman M, Hierons RM. Search algorithms for regression test case prioritization. IEEE Transactions on Software Engineering. 2007; 33(4):225–37.
7. Salami AL. Evolutionary algorithm definition. American Journal of Engineering and Applied Science. 2009; 2(4):789–95.
8. Byson N. A goal programming method for generating priorities vectors. Journal of Operational Research, England. 1995; 46(5):641–8.
9. Crawford G, Williams C. A note on the analysis of subjective judgment matrices. Journal of Mathematical Psychology. Elsevier Publications. 1985; 29(4):387–405.
10. Singh Y, Kaur A, Suri B. Regression test selection and prioritization using variables: Analysis and experimentation. Software Quality Professional Magazine. 2009 Mar; 11(2):1–15.
11. Dorigo M, Maniezzo V, Colorni A. The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics. 1996; 26(1):29–41.
12. Suri B, Singhal S. Implementing ant colony optimization for test case selection and prioritization. International Journal of Computer Science and Engineering. 2011 May; 3(5):1924–32.
13. Srivastava PR, Baby K. Automated software testing using meta-heuristic technique based on an ant colony optimization. International Symposium on Electronic System Design (ISED); Bhubaneshwar, India. 2010 Dec. p. 235–40.
14. Singh Y, Kaur A, Suri B, Singhal S. Test case prioritization using ant colony optimization. ACM SIGSOFT Software Engineering Notes. 2012; 35(4):1–7.
15. Chandu PMSS, Sasikala T. Implementation of regression testing of test case prioritization. Indian Journal of Science and Technology. 2015 Apr; 8(S8):290–3. DOI: 10.17485/ijst/2015/v8iS8/61922.
16. Elbaum S, Malishevsky A, Rothermel G. Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering. 2002; 28(2):159–82.
17. Elbaum S, Rothermel G, Kanduri S, Malishevsky AG. Selecting a cost-effective test case prioritization technique. Software Quality Journal. 2004; 12(3):185–210.
18. Malishevsky AG, Ruthruff JR, Rothermel G, Elbaum S. Cost Cognizant Test Case Prioritization, Technical Report. University of Nabraska Lincoln, 2006.
19. Srivastava PR. Test case prioritization. Journal of Theoretical and Applied Information Technology. 2008; 4(3):178–81.
20. Raju S, Uma GV. Factors oriented test case prioritization technique in regression testing using genetic algorithm. European Journal of Scientific Research. 2012; 74(3):389–402.
21. Berndt DJ, Watkins A. Investigating the performance of genetic algorithm based software test case generation. Proceedings of IEEE International Symposium on High Assurance Systems Engineering; 2004. p. 261–2.
22. Xanthakis S, Ellis C, Gall AL, Karapoulios K. Application of genetic algorithms to software testing. Proceedings of

International Conference on Software Engineering and its Applications; 1992. p. 625–36.
23. Maheswari RU, Mala DJ. Combined genetic and simulated annealing approach for test case prioritization. Indian Journal of Science and Technology. 2015 Dec; 8(35):1–5. DOI: 10.17485/ijst/2015/v8i35/81102.
24. Kaur A, Goyal S. A bee colony optimization algorithm for code coverage test suite prioritization. International Journal of Engineering Science and Technology. 2011; 3(4):2786–795.
25. Jeyamala D, Mohan V. ABC-artificial bee colony optimization based test suite optimization technique. International Journal of Software Engineering. 2009; 2(2):1–33.
26. McCaffrey JD. Generation of pair-wise test sets using a simulated bee colony algorithm. Proceedings of IEEE International Conference on Information Reuse and Integration; 2009 Aug. p. 115–9.
27. Maheshwari V, Prasanna M. Generation of test case using automation in software systems: A review. Indian Journal of Science and Technology. 2015 Dec; 8(35):1–9. DOI: 10.17485/ijst/2015/v8i35/72881.
28. Jacob TP, Ravi. An optimal technique for reducing the effort of regression test. Indian Journal of Science and Technology. 2013 Aug; 6(8):5065–9. DOI:10.17485/ijst/2013/v6i8/36345.