Indian Journal of Science and Technology, Vol 11(48), DOI: 10.17485/ijst/2018/v11i48/131746, December 2018
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Comparative Performance Evaluation of "BA-TPF" Technique for Regression Test Case Optimization

Sandeep Dalal*, Sudhir and Kamna Solanki

Maharshi Dayanand University, Rohtak – 124001, Haryana, India; Sandepdalal.80@gmail.com, Sudhir.gehlawat1234@gmail.com, kamna.mdurohtak@gmail.com

Abstract

Objectives: To make comparative performance evaluation of "BA-TPF" technique for regression test case optimization on "Online Shopping", "Order Management", "ATM System" and "Telephone System". **Methods:** "BA-TPF" focuses on generation of optimized test sequences using bat algorithm and then producing the optimized and prioritized test suite by calculating the prioritized test suite possessing the highest path fitness. This study carries out the comparative evaluation of the recently proposed "BA-TPF" technique for optimized test sequence generation and prioritization. Three metrics namely TCP ("Test Suite Percentage required for Complete Path Coverage"), PRT ("Percentage of Repeated Transitions") and APPC ("Average Percentage of Path Covered") have been used for performance measurement of the proposed technique. **Findings:** The proposed technique "BA-TPF" demonstrates its competence on all the three parameters. It could achieve highest path coverage with minimized repetitions and minimized test suite in comparison to contemporary techniques for regression test optimization. **Improvements:** This approach basically generates the test path sequences using the bat algorithm and then calculates the test path fitness on the generated sequences. The prioritized test sequences are generated by arranging the test sequences in descending order from highest to lowest path fitness values.

Keywords: Genetic Algorithm, Path Coverage, Regression Testing, Software Testing, Test Case Optimization

1. Introduction

Software code keeps on changing as the defects are fixed by the developers. Every time a defect gets fixed, the software code is altered and updated. Moreover, the requirements of the end-user cannot be considered as fixed in most of the cases and these ever-changing user requirements also contribute towards continuous changes in code. Regression Testing is also an essential testing that need to be performed after the amendments are incorporated in the software code to ensure that these amendments do not adversely affect rest of the software system[1]. One can conclude that changes/ amendments in the software code are inevitable. Hence, Regression testing is a necessary evil because regression testing is extremely expensive, time intensive and iterative phase of software testing that must be carried out after every change in software code.

Figure 1 depicts the regression testing process by taking an example of any software program P, which has been released as version *X. After P gets released, any kind of corrective, adaptive or perfective maintenance will change the software code leading to the newer version of software program P' (release version Y). Thereafter, this modified software program P' undergoes rigorous testing again to as a safety measure so that the accurate functioning of newly added functionality can be assured. Moreover, this re-testing of software program P' ensures that the change in code has not affected rest of the system and the newer defects have not been introduced in system. Therefore, running the regression tests cross verifies that all kinds of malfunctioning of the updated software program P' gets detected and fixed before P' is released as version Y.* Regression testing can be implemented in various ways. A software developer can re-execute the test cases of the original test suite

*Author for correspondence

which was executed on the earlier version of the software system; this method is often denoted as "retest-all approach"[1-3]. However, the limitation of this approach is that it is impractical to re-execute all test cases again and leads to useless efforts as modification affect only a portion of a system. Some test suites are extremely large, therefore cost and time constraints do not permit re-execution of all test cases in a test suite to carry out regression tests[4,5]. Therefore, developers may use any of the following options:

- Test Case Selection: "Run only a subset of the test suite, which can be chosen by using regression test selection techniques".
- Test Case Minimization: "Permanently reduce the number of test cases by eliminating redundant test cases from the test suite".
- Test Case Prioritization: "Re-schedule the execution sequences of the test cases within the test suite in a particular order to maximize some goal"

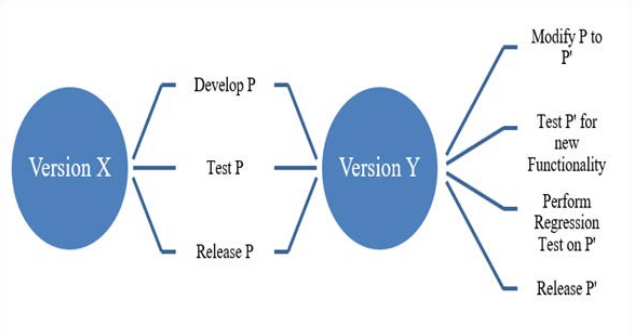

Figure 1. Regression testing process.

Running entire set of regression test suite after every change makes it a cumbersome take that requires cost, time and efforts repetitively[6–8]. Moreover, time and cost constraints always create barrier in running the entire test set repeatedly to carry out regression testing. Keeping in view the complexity and iterative nature of regression testing, there arises a need of minimized and optimized set of set cases that can perform the task of regression testing without compromising the efficiency and efficacy of regression testing. To efficiently handle the quickly changing user requirements of modern-day large scale and complex software systems, there is an urgent need of efficient and effective regression testing paradigms that can reduce overall cost, time, efforts and assists in enhancing software quality[9,10].

1.1 Optimization:

It is the process of achieving certain objective using minimum resources and efforts based on an approach better than other contemporary approaches. "Optimization algorithms are search methods where the goal is to find an optimal solution to a problem, in order to satisfy one or more objective functions, possibly subject to a set of constraints. "The optimization process is shown in Figure 2. Optimization is the method using which one can discover the maximum or minimum value of a function or process. This method is used in many fields such as engineering, physics, economics and chemistry where the objective is to maximize effectiveness or production etc. Optimization can either refer to or maximization or minimization. "Maximization of a function f is equivalent to minimization of the opposite of this function –f." Scientists and researchers may conjure up a novel idea and optimization tries to improve that idea.

Open research questions in the field of regression testing includes efficient reduction and multi-objective optimization of size of regression test suite, test data generation for regression testing techniques, development of customized regression testing techniques for other emerging domains of software development[1–4].

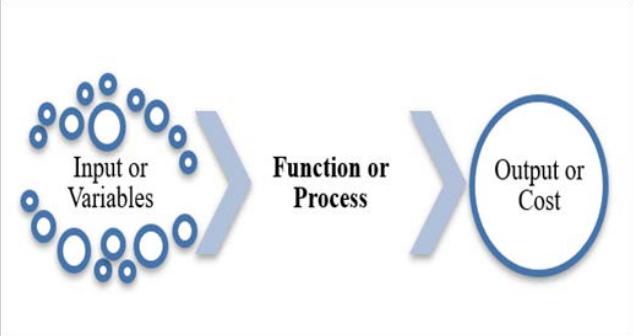

Figure 2. Optimization process.

2. Comparative Performance Evaluation of "BA-TPF"

Viability of testing remains lower than desires, even after the extensive research to develop efficient testing techniques to enhance the quality of the product. Although testing is very costly and time -consuming activity, still there is no substitute for testing as it is the only way to achieve software quality. It is not feasible to completely and exhaustively test software due to cost and time constraints. Therefore, even after usage of maximum efforts

and efficient testing techniques, there is no guarantee of producing bug-free software. Testing can only reveal the existence of bugs; however, it can never guarantee the absence of bugs (or defects or faults). Hence, basic goal of software testing is revealing maximum bugs within minimum stipulated time and efforts. As exhaustive testing of software is not possible, the focus must be towards utilizing the most suitable and optimized testing techniques among the pool of available testing strategies. Therefore, the most challenging job in software testing is to pick most appropriate and influential testing techniques (or approach) that can proficiently assist in intensifying test efficacy and efficiency up to highest extent. Latest research studies endorse that appropriate combination of diverse testing techniques must be used for testing the software projects as diverse techniques emphasis on multiple aspects of software testing. Regardless of the fact that many advancements have been carried out in recent years for evaluating the effectiveness and proficiency of testing techniques; a long way is still pending to cover as the results are sometimes either contradictory or are confined for a specific domain. Subsequently, as software testing is considered the most significant and most expensive phase, the goal of this present research would not be towards discovering a single approach that can substitute the other testing approaches/techniques in all aspects; but to propose an approach the which can assist us to optimize the regression test suite by satisfying multiple objective functions of optimization like minimized repetitions in transitions, enhance path coverage with minimized test suite size . Most of the research in the field of regression test case optimization has focused only towards enhancing the code coverage, path coverage and number of faults detected[11–23]. Negligible research work has been carried out to develop multi-objective regression test case optimization technique to enhance path coverage with minimum test suite and minimum transition repetitions. In this regard, a novel technique "BA-TPF24" ("Bat Algorithm-Test Path Fitness") has been proposed which uses bat algorithm for test sequence generation and then applies test path fitness criteria for prioritizing these test sequence to achieve this multi-objective optimization. The next section describes the comparative evaluation of the proposed "BA-TPF" technique against other techniques[24].

The proposed "BA-TPF" technique for regression test case optimization has been experimentally evaluated using four case studies namely online shopping system, Order Management System, ATM system, Telephone System[8]. This study carries out the comparative performance evaluation of "BA-TPF" approach for test case optimization using three metrics:

- TCP (Test Suite Percentage required for Complete Path Coverage)
- PRT (Percentage of Repeated Transitions)
- APPC (Average Percentage of Path Covered)
- TCP (Test Suite Percentage required for Complete Path Coverage) Metric. A test sequence with lower value of TCP metric is considered as best and cost-effective because it needs lesser test cases to be executed without compromising the path coverage capability. Minimum value of TCP implies achieving complete path coverage with minimum time and minimum efforts as minimum test cases execution is required.

$$TCP = \frac{\text{No.Of Test Cases Required for Complete Path Coverage}}{\text{Total Number of Test Case}} \times 100 \quad (1)$$

- PRT metric calculates the percentage of the redundant transitions in a test sequence. Notion for Calculation of PRT can be defined as:

$$PRT = \frac{\text{Total No.Of Repeated Transitions in a Test Sequence that achieve complete Path coverage}}{\text{Total Number of Test Cases required for Complete Path Coverage}} \times 100 \quad (2)$$

A test sequence with lower value of PRT metric is considered as better as it possesses the lesser redundancy in comparison to the other test sequences. Minimum value of PRT implies achieving complete path coverage with minimum repetitions and redundancy.

- APPC (Average Percentage of Path Covered) Metric[4]: This metric proposed by basically finds the average percentage of path or transitions covered by an approach. Higher value of APPC is solicited for better test case optimization technique.

$$APPC = \left[1 - \frac{TP1 + TP2 + TP3 \ldots\ldots TPm}{mn}\right] + \frac{1}{2n} \quad (3)$$

where, n=no. of test cases

m=no. of test paths covered

TP_m= Position of test case that covers the m^{th} test path first

The performance evaluation of the proposed "BA-TPF" approach has been carried out against contemporary approaches like Bat Algorithm (BA), Genetic

Algorithm (GA), Ant Colony Optimization (ACO), and Bee Colony Optimization (BCO) to measure the effectiveness of the proposed approach.

2.1 ATM System-Case Study 1

The proposed "BA-TPF" technique was experimentally evaluated against other contemporary approaches on "ATM System" case study. The results obtained were measured based on three metrics values as shown in Figure 3.

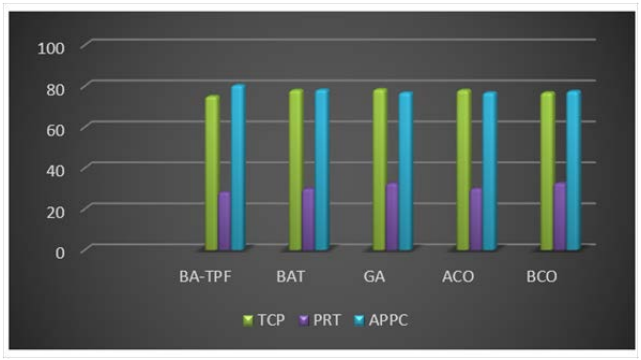

Figure 3. Comparative analysis of proposed "BA-TPF" technique-case study1.

2.2 Online Shopping-Case Study 2

The proposed "BA-TPF" technique was experimentally evaluated against other contemporary approaches on "Online Shopping" case study. The results obtained were measured based on three metrics values as shown in Figure 4.

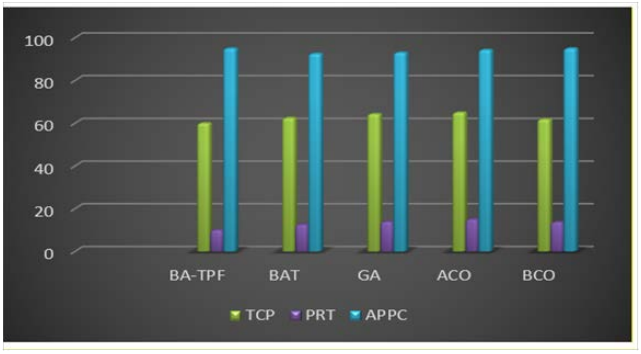

Figure 4. Comparative analysis of proposed "BA-TPF" technique-case study 2.

2.3 Order Management-Case Study 3

The proposed "BA-TPF" technique was experimentally evaluated against other contemporary approaches on "Order Management" case study. The results obtained were measured based on three metrics values as shown in Figure 5.

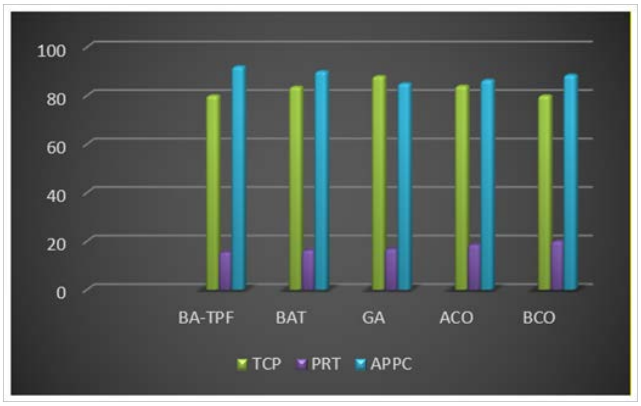

Figure 5. Comparative analysis of proposed "BA-TPF" technique-case study 3.

2.4 Telephone System-Case Study 4

The proposed "BA-TPF" technique was experimentally evaluated against other contemporary approaches on "Telephone System" case study. The results obtained were measured based on three metrics values as shown in Figure 6.

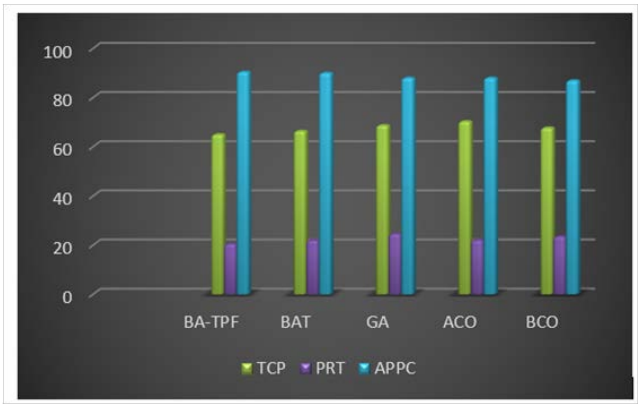

Figure 6. Comparative analysis of proposed "BA-TPF" technique-case study 4.

3. Comparative Metrics Performance Evaluation

This section makes a comparative evaluation of the proposed technique based on individual parameters or metrics namely TCP, PRT, APCC. The comparison truly depicts the efficiency and efficacy of the proposed technique in terms of its ability to attain higher path coverage, minimum transition repetitions and minimum test suite size for execution.

3.1 Comparative TCP Values

This section makes a comparative performance evaluation of the proposed "BA-TPF" technique in terms of TCP values against other contemporary approaches. The results obtained are shown in Figure 7.

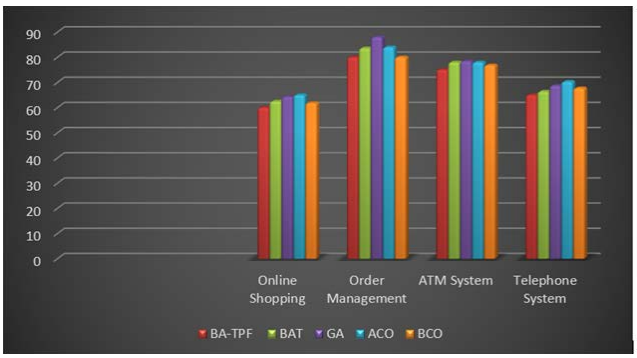

Figure 7. Comparison of TCP values.

3.2 Comparative PRT Values

This section makes a comparative performance evaluation of the proposed "BA-TPF" technique in terms of PRT values against other contemporary approaches. The results obtained are shown in Figure 8.

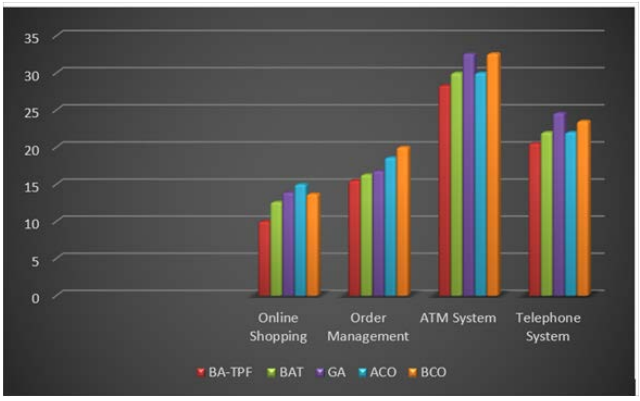

Figure 8. Comparison of PRT values.

3.3 Comparative APPC Values

This section makes a comparative performance evaluation of the proposed "BA-TPF" technique in terms of APCC values against other contemporary approaches. The results obtained are shown in Figure 9.

(Figure 10) depicts the comparative analysis of all the case studies. It is evident from the comparative performance evaluation of the proposed "BA-TPF" technique that the proposed techniques outperform the other techniques on all fronts collectively on all parameters considered. Individual parameter performance comparison also reveals that proposed technique has the potential of achieving higher path coverage (higher APCC) with minimized repetitions (lower PRT) and minimum test suite execution size (lower TCP).

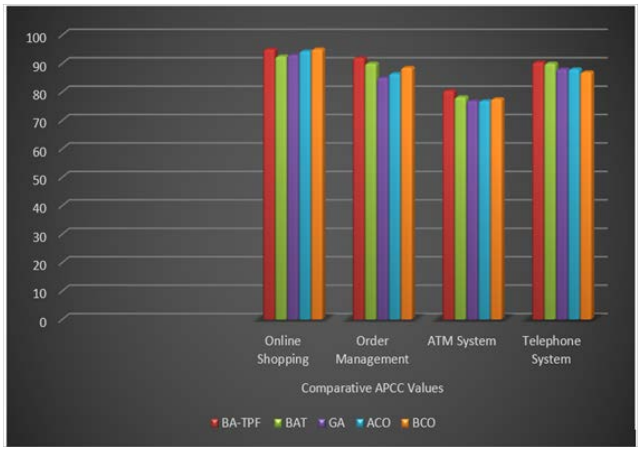

Figure 9. Comparison of APCC values.

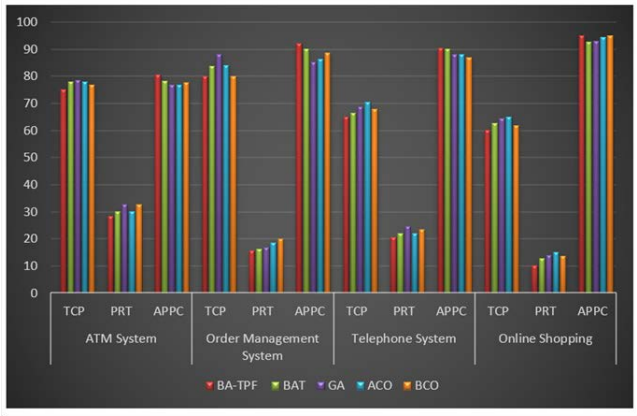

Figure 10. Comparative analysis of proposed "BA-TPF" technique using all case studies.

4. Conclusion

This study presented the comparative performance evaluation of recently proposed "BA-TPF" technique against other approaches like GA, BCO, ACO and BA. It has been observed that the proposed technique has the potential of achieving higher path coverage (higher APCC) with minimized repetitions (lower PRT) and minimum test suite execution size (lower TCP). The future research is to focus on measuring the effectiveness of the proposed m-ACO technique on some other parameters using diverse case studies.

5. References

1. Beizer B. Software testing techniques. 2nd Edition. Dreamtech Press: India; 2003.
2. Catal C, Mishra D. Test case prioritization: A Systematic study. Software Quality Journal. 2013; 21(2):445–78. https://doi.org/10.1007/s11219-012-9181-z
3. Chandu PMSS, Sasikala T. Implementation of regression testing of test case prioritization. Indian Journal of Science and Technology. 2015 Apr; 8(8):290–3. https://doi.org/10.17485/ijst/2015/v8iS8/61922
4. Li Z, Harman M, Hierons RM. Search algorithms for regression test case prioritization. IEEE Transactions on Software Engineering. 2007; 33(4):225–37. https://doi.org/10.1109/TSE.2007.38
5. Leung H, White L. Insights into Regression testing. Proceedings of the IEEE International Conference on Software Maintenance; 1989 Oct. p. 60–9. PMid:2671494
6. Elbaum S, Malishevsky A, Rothermel G. Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering. 2002; 28(2):159–82. https://doi.org/10.1109/32.988497
7. Elbaum S, Rothermel G, Kanduri S, Malishevsky AG. Selecting a cost-effective test case prioritization technique. Software Quality Journal. 2004; 12(3):185–210. https://doi.org/10.1023/B:SQJO.0000034708.84524.22
8. Srivastava PR. Test case prioritization. Journal of Theoritical and Applied Information Technology. 2008; 4(3):178–81.
9. 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. https://doi.org/10.17485/ijst/2015/v8i35/72881
10. Solanki K, Singh Y. An empirical literature study of various test case prioritization techniques. Software Engineering and Technology. 2014; 6(6):169–73.
11. Solanki K, Singh Y, Dalal S, Srivastava PR. Test case prioritization: An approach based on modified Ant Colony Optimization. Emerging Research in Computing, Information, Communication and Applications; 2016. p. 213–23.
12. 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):2786–95.
13. 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. https://doi.org/10.1145/1811226.1811238
14. Baudry B, Fleurey F, Jézéquel JM, Le Traon Y. Automatic test case optimization: A bacteriologic algorithm. IEEE Software. 2005, 22(2):76–82. https://doi.org/10.1109/MS.2005.30
15. Solanki K, Singh Y, Dalal S. Test case prioritization: An approach based on modified ant colony optimization. International Conference on Computer, Communication and Control (IC4); 2015. p. 1–6.
16. Singh Y, Kaur A, Suri B. Test case prioritization using ant colony optimization. ACM SIGSOFT Software Engineering Notes. 2010; 35(4):1–7. https://doi.org/10.1145/1811226.1811238
17. Solanki K, Singh Y, Dalal S. A comparative evaluation of «m-ACO» technique for test suite prioritization. Indian Journal of science and technology. 2016; 9(30):1–10.
18. Solanki K, Singh Y, Dalal S. Experimental analysis of m-ACO technique for regression testing. Indian Journal of Science and Technology. 2016; 9(30):1–7. https://doi.org/10.17485/ijst/2016/v9i30/86588
19. Solanki K, Singh Y. Importance of selecting test cases for regression testing. IOSR Journal of Computer Engineering. 2014; 16(4):43–51. https://doi.org/10.9790/0661-16444351
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. 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.
22. Musa S, Sultan AB, Ghani AB, Baharom S. Software regression test case prioritization for object-oriented programs using genetic algorithm with reduced-fitness severity. Indian Journal of Science and Technology. 2015 Nov; 8(30):1–9. https://doi.org/10.17485/ijst/2015/v8i30/86661
23. Maheswari RU, JeyaMala D. Combined genetic and simulated annealing approach for test case prioritization. Indian Journal of Science and Technology. 2015 Dec; 8(35):1–5.
24. Sudhir DS. Experimental analysis of "BA-TPF" for regression test case optimization. International Journal of Engineering and Technology. 2018; 7(4):1–7.