International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August 2019

Bi-Objective Constraint and Hybrid Optimizer for the Test Case Prioritization

K. Senthil Kumar, A. Muthukumaravel

Abstract: ***Regression testing is performed to make conformity that any changes in software program do not disturb the existing characteristics of the software. As the software improves, the test case tends to grow in size that makes it very costly to be executed, and thus the test cases are needed to be prioritized to select the effective test cases for software testing. In this paper, a test case prioritization technique in regression testing is proposed using a novel optimization algorithm known as Taylor series-based Jaya Optimization Algorithm (Taylor-JOA), which is the integration of Taylor series in Jaya Optimization Algorithm (JOA). The optimal test cases are selected based on the fitness function, modelled depending on the constraints, namely fault detection and branch coverage. The experimentation of the proposed Taylor-JOA is performed with the consideration of the evaluation metrics, namely Average Percentage of Fault Detected (APFD) and the Average Percentage of Branch Coverage (APBC). The APFD and the APBC of the proposed Taylor-JOA is 0.995, and 0.9917, respectively, which is high as compared to the existing methods that show the effectiveness of the proposed Taylor-JOA in the task of test case prioritization.***

Keywords: ***Prioritization, Regression testing, Jaya Optimization Algorithm, constraints, and branch coverage.***

I. INTRODUCTION

Software systems have spread everywhere in the world from the modern households to the space stations. Software is acting complex, in the presence of multifold functionalities for the facilitation of different services. In the same way, testing of these complex software systems is becoming very tedious. Thus, the requirement of the number of test item increases considerably that increases the cost of testing.

Testing of software is the main step in software development life cycle (SDLC), and the test case must be developed for the improvement of the quality of software. A testing process becomes successful, if it has the ability to find the as-yet-undiscovered error [10] [4]. The testing is a continuous process that continues till the maintenance of the project, after the coding phase of SDLC. The SDLC can be grouped as two activities, such as verification and validation.

Revised Manuscript Received on August 30, 2019.
* Correspondence Author
K. Senthil Kumar*, Research Scholar, Bharathiar University, Coimbatore, Tamilnadu,India.
A. Muthukumaravel, Professor & Head, Department of MCA Bharath University, Chennai (Tamilnadu)India.

Verification is the evaluation of the mediator products of the software to check the satisfaction of the conditions stated at starting phase, whereas validation is the evaluation of the end product to find whether the software satisfy the requirements that were specified [11] [4]. Testing involves the programs to find the bugs and faults in
software, and there are various methods of testing that the tester of software uses based on their needs, such as Mutation Testing, Regression Testing, security testing, stress testing, load testing, white Box Testing, and black box testing. Based on the type of testing, the tester develops certain number of test cases, which is termed as Test Suite [9]. During testing, the tester decides the number of test cases, and then implements it on software depending on the developed test cases and finally, verifies the results of the executions that were performed. Regression testing is a method of testing that is applied on the altered application with the use of pre-defined sets of test cases. In this type of testing, the test cases are prioritized in such a way to reuse the test cases that were generated newly and the existing test cases. Test case prioritization is performed with various methods, and in test case prioritization, all the test cases are organized in such an order to magnify the equitable characteristics. For the establishment of the priorities of test cases, some factors based on the needs are tested and selected, and then the preference is granted to each test case. The test case prioritization paves a way to lineup and executes the test cases with maximum priority to find the earlier faults [9]. It prioritizes the test cases based on business impact, importance, and frequently used functionalities [12], and attempts to order the test cases for execution in such a way to increase the likelihood to reveal the faults early at the process of retesting [13]. Prioritization increases the probability that if the testing ends prematurely, the most important test cases may run [14]. These prioritization methods make the testers to order their test cases in such a way that the test cases possessing higher priority are executed earlier as compared to the test cases possessing lower priority. As the TCP methods do not discard the test cases, they have the capability of avoiding the limitations of test case minimization methods [15], [16], [6]. However, there are number of test cases that make in use of the prioritization methods for increasing the efficiency of the fault detection rate [17]. Regression testing is an expensive and integral part in the software testing process, and in order to reduce its effort, test case prioritization methods were developed [6]. All the test case prioritization methods must address one or more objectives of test case prioritization, and the selection of the test cases may give the way to optimize the performance goals, like minimization of execution time and maximization of fault coverage.

Some of the optimization methods are important in increasing the software quality and reducing the testing process. There are various traditional and heuristic methods for test case optimization, such as greedy search and so on, but all are affected with the problem of local optima. The soft computing meta-heuristic application is effective in the testing phase of software in the development of software [18]. There are various soft computing methods applied in various phases of the software development cycle. Some of the beneficial approach is fuzzy logic, genetic algorithm, artificial intelligence and so on. Some of the advanced types of meta-heuristic algorithm were applied on test case prioritization, such as Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Cuckoo search (CS), and Harmonic Search (HS). All these algorithms were nature inspired and works in an optimized nature. The aim is to solve the multi-objective test case optimization that provides an accurate result and satisfy the objectives including less time, code coverage, and maximum fault coverage [7].

The main intention for the development of proposed concept is to prioritize the test cases by proposing a novel optimization algorithm. The test case to be selected for the detection of fault is decided based on two important constraints, namely branch coverage and fault. The searching criterion is developed as an optimization problem to obtain the optimal test cases based on the branch coverage and fault and the optimization exploits the fitness measure for determining the optimal test cases. The fitness function for optimal test case generation is performed by the consideration of the constraints. The developed optimization algorithm, known as Taylor-JOA is developed by the hybridization of Taylor series concept [25] in JOA [23]. Thus, the test case prioritization in the regression testing is performed using the proposed Taylor-JOA-based test case prioritization with the consideration of five subjects, namely print_tokens, schedule, print_tokens2, schedule2, and space [24].

The major contribution of the paper is:

II. BI-OBJECTIVE CONSTRAINT AND TAYLOR-JOA FOR THE TEST CASE PRIORITIZATION

The method of test case prioritization termed as Taylor-based Jaya Optimization Algorithm is proposed to find the fault with minimum time by considering the bi-objectives, such as branch coverage and fault detection.

The organization of the paper is described as: section 1 explains the introduction to the need of test case prioritization. Section 2 deals with the literature review and the drawbacks associated with the existing methods. Section 3 deals with the proposed system of test case prioritization using the Taylor-JOA. Results and discussion are discussed in section 4 and the section 5 concludes the paper.

A. Literature survey

The eight literatures related to the test case prioritization are discussed as: Yi Bian *et al.* [1] developed the Epistatic Test case Segment (ETS) based ACO, which had a considerable improvement to the origin regarding the efficiency. In addition, it overcame the conventional NSGA-II algorithm, but the external threats lowered the accuracy including the features under test and the objectives that were utilized. Jinfu Chen *et al.* [2] modelled the adaptive random sequence approach based on clustering, which was good in case of large scale object-oriented software (OOS) testing, but the sampling strategy was needed to be improved for the better optimization of the test case prioritization (TCP). Breno Miranda and Antonia Bertolino [3] designed the Scope-aided testing approach for test prioritization. This strategy, when applied to minimize the test suites lowered the impact in terms of reducing the in-scope fault detection capability, but was not suitable to maintain consistently in all studies. Soumen Nayak *et al.* [4] developed the test Case Prioritization Technique with the improvement in the Rate of Fault Detection. Maximum faults were detected with the consumption of less time using this method, but cannot be used in real-time situations. Deepak Panwar *et al.* [5] modelled the Meta-heuristic technique called ACO that was able to find the maximum faults with the execution of minimum numbers of test cases, but the theoretical analysis was difficult. S. K. Harikarthik *et al.* [6] developed the Modified Artificial Neural Network classification algorithms for test case prioritization that required very less time and memory requirements, but had increased computational burden. Sushant Kumar and Prabhat Ranjan [7] modelled the ACO based test case prioritization that required reduced effort and cost in case of regression testing operation, but took crucial time in the selection of test case. Aitor Arrieta *et al.* [8] developed the multi-objective test generation and prioritization approach that was very effective in finding the faults in test cases, but provided very less performance.

B. Challenges

There are various problems associated with the conventional methods of test case prioritization, which are stated as,

- Adaptive methods of test case prioritization suffer from financial and scalability problems [19].
- The conventional test case prioritization method utilized in [20] do not lower the count of generated test cases, and hence affected with the problems of time complexity in running the generated test cases.
- Risk-based prioritization of test cases with the use of a fuzzy expert model in [21] has reduced efficiency in computation.
- Input-based adaptive randomized prioritization utilized in [22] was not able to handle the input datasets of large size, and thus cost effective results are not possible to be obtained.
- The traditional methods of test case prioritization consider the fixed strength and not the multiple strengths [6], when selecting the test case. Thus, to solve those issues, an advanced method is needed to be developed.

III. PROPOSED TAYLOR SERIES-BASED JAYA OPTIMIZATION ALGORITHM IN TEST CASE PRIORITIZATION

The primary aim of this paper is to design a test case prioritization method in the regression testing by proposing a novel optimization algorithm. The test case to be selected is decided using a searching criterion based on two major constraints, such as fault and branch coverage. Accordingly, the effective searching criterion is formulated to find the optimal test cases depending on the constraints. The searching criterion is formulated as the problem of optimization, which is solved with an algorithm, named Taylor-JOA. The proposed Taylor-JOA is designed by introducing Taylor series concept in JOA and the fitness function for the optimal test case generation is carried out by considering bi-objectives, such as coverage and fault. Thus, the regression tester performs the test case prioritization in the regression testing using the proposed Taylor-JOA. Moreover, it considers five subjects, such as print_tokens, print_tokens2, schedule, schedule2, and space, for experimentation. The block diagram of the proposed Taylor-JOA based test case prioritization technique is depicted in figure 1.

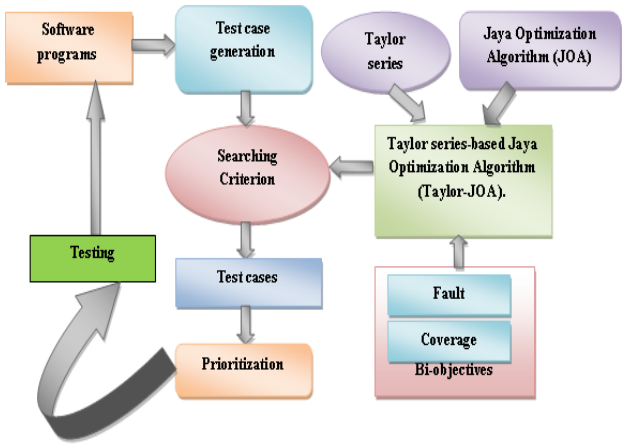

Fig. 1 Block diagram of the proposed Taylor-JOA in test case prioritization

A. Generation of test cases

The test cases are usually employed for detecting the fault with the loop by loop execution of the program while acquiring the minimal execution time. The execution time and cost is minimized when minimum test cases are chosen for software testing that aims at finding the fault present in a program. The test cases are generated with the consideration of two constraints, namely fault detection and branch coverage. The test cases with the maximal ability for detecting the faults and maximal branch coverage are chosen for the effective generation of the test cases. As an instance, let us consider that there are three attributes in each test cases such that the generated test cases are of size, $[3\times 3]$. The generated test cases are subjected to prioritization that defines the order of execution of test cases. In other words, the test case are executed in the order of their priorities with the test case with highest priority is said to be executed first.

B. Test case prioritization

The test cases obtained from the previous step are prioritized to manage the order of execution in an optimal way to find the fault in the software programs with minimum execution time using the proposed Taylor-JOA algorithm. The chosen test cases should cover the entire software program through checking each and every loop and branch for fault thereby, achieving maximal fault and branch coverage. The fitness measure is formulated based on the two objectives, termed as maximum fault detection and maximum branch coverage.

Solution encoding

Consider three test cases, $t_1, t_2,$ and t_3 and the order of execution of test cases is based on the priorities and the prioritization is enabled based on the constraints, such as fault detection and branch coverage. The fitness value of the test cases lies between 0 and 1, and the test case possessing minimum value of fitness is given the highest priority. Assume that $t_2 < t_1 < t_3$, then the test case t_2 is given first priority p_1, the test case t_1 is provided second priority p_2, and the test case t_3 is given the third priority p_3. Thus, the order of execution is given as $t_2, t_1,$ and t_3. Figure 2 shows the solution encoding of the proposed Taylor-JOA based test case prioritization method.

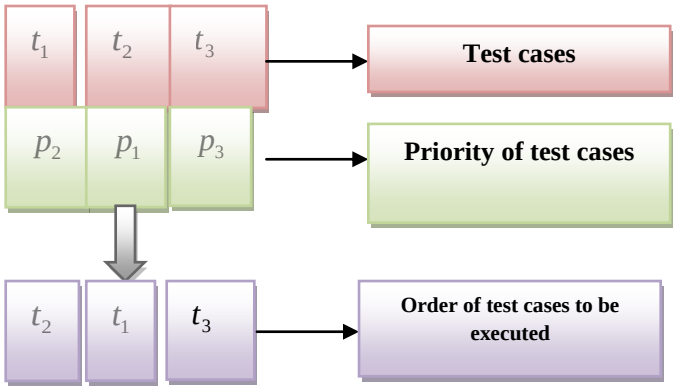

Fig. 2 Solution encoding of proposed Taylor-JOA

Fitness

The order of execution of the test case is decided depending on fitness measure, which relies on the constraints, such as fault detection and branch coverage. The relation for fitness is expressed as,

$$Fitness = \frac{APFD + APBC}{2} \quad (1)$$

a) Average Percentage of Fault Detected (APFD):

This metric indicates the weighted average of faults that are detected, and it is widely utilized in the evaluation of the test case prioritization methods. The APFD depends on the factors, such as order of execution of the faults, number of test cases and the number of faults, and it is expressed as,

$$APFD = 1 - \left[\frac{T_{f1} + T_{f2} + ... + T_{fu}}{ku} \right] \times \frac{1}{2u} \quad (2)$$

where, k is the number of test cases, u is the number of faults, T_{f1} is the position of initial test in the test suite that detects fault 1, T_{f2} is the position of initial test in the test suite that detects fault 2, and T_{fu} is the position of initial test in the test suite that detects the fault u. Consider an example with three test cases, such as $t_1, t_2,$ and t_3 with their respective priorities being $p_1, p_2,$ and p_3. Assume that the program consists of two faults $f_1,$ and f_2 to be detected. The test cases mentioned above finds both the faults $f_1,$ and f_2, and the factors, T_{f1} and T_{f2} depends on the test cases priority even if all the test cases detect the faults and the value for T_{f1} and T_{f2} is the order of their test cases detecting the faults.

b) Average Percentage of Branch Coverage (APBC):

This metric estimate the rate of prioritized test suite that covers the branches. The APBD depends on the factors, such as order of execution of the branches, number of test cases and the number of faults, and it is expressed as,

$$APBC = 1 - \left[\frac{T_{b1} + T_{b2} + ... + T_{bu}}{ku} \right] \times \frac{1}{2u} \quad (3)$$

Where, T_{b1} is the position of initial test in the test suite that covers the branch1, T_{b2} is the position of initial test in the test suite that covers the branch 2, and T_{bu} is the position of initial test in the test suite that covers the branch u. Consider an example with three test cases, such as $t_1, t_2,$ and t_3 with their respective priorities being $p_1, p_2,$ and p_3. Assume that the program consists of five branches $b_1,$ $b_2,$ $b_3, b_4,$ and b_5 that are needed to be covered. Let us assume that the test case t_1 and t_2 covers all the branches $b_1,$ $b_2,$ $b_3, b_4,$ and b_5, but the factors $T_{b1} + T_{b2} + ... + T_{bu}$ is based on the priority of the test cases t_1 and t_2. For a method to be effective in test case prioritization, the values of APFD and APBC must be maximal, and the fitness measure is necessary to be minimal. Thus, in the fitness constraints in equation (2) and (3), the constraints APFD and APBC are subtracted from one.

Algorithmic steps of the Taylor series-based Jaya Optimization Algorithm

The test case prioritization phenomenon enormously helps in finding the faults present in a program through the loop by loop execution and covering all the branches within the minimal execution time. A lot of test case prioritization methods are available to identify the faults, but these techniques are highly time-consuming. In addition, the detection of changes in the software programs and the affected parts of the program acts as the major challenges in the existing methods of test case prioritization. Hence, there is a need for an algorithm that requires less time and effective in the prioritization of the test case. JOA is an efficient optimization algorithm that has the ability to solve both the unconstrained and constrained problems of optimization.

All the test cases are given as input to the proposed Taylor-JOA approach that optimizes the rules and produces enhanced results. The advantage of JOA over other optimization algorithms is that all the observations are utilized for both the training and validation process, and each observation is used for the validation exactly only once. One of the main reasons for the enhanced performance of the JOA is that it always moves away from worst solution and converges to best solution. The drawback seen in JOA is that it is a deterministic method and not a heuristic method and does not use a random function that leads to poor scalability. In order to overcome the limitations associated with the JOA, and to inherit the advantages of both Taylor series and the JOA, the concept of Taylor series is integrated in JOA. The Taylor series prediction (TSP) criterion is used as it increases the scalability, and able to rectify the drawbacks associated with the JOA. In addition, it provides more accurate prediction on current test cases even with greatly varying access patterns. The TSP can be compared using the first order, second order, and the higher order predictions, whereas the higher order predictions offer more accurate results.

i) Initialization:

The first step in the Taylor-JOA is the initialization of the parameters. The design variables p, candidate solutions q, and the termination conditions till the l^{th} iteration are initialized in this step.

ii) Fitness evaluation:

The fitness of the method is evaluated based on the constraints, namely fault detection and branch coverage. For a system to be effective, the fitness measure must be minimal, indicating maximum fault detection and branch coverage. The fitness value is estimated based on the relation expressed in equation (1).

iii) Generation of random variables:

The random variables for all g, where $g = 1,2,...,p$ were selected.

iv) Estimation of best solution:

The solution possessing least fitness value is selected optimally using the proposed Taylor-JOA optimization method, which indicates the order of test case prioritization. The standard expression for the JOA is expressed in equation (4) as,

$$J_{g,h,l}^{s+1} = J_{g,h,l}^{s} + m_{1.g,l}\left(J_{g,best,l} - \left|J_{g,h,l}^{s}\right|\right) - m_{2.g,l}\left(J_{g,worst,l} - \left|J_{g,h,l}^{s}\right|\right) \quad (4)$$

where, $J_{g,best,l}$ indicates the value of variable g for best candidate, $J_{g,worst,l}$ indicates the value of variable g for worst candidate, $\left|J_{g,h,l}^{s}\right|$ is the absolute value, $J_{g,h,l}^{s+1}$ is the updated value of $J_{g,h,l}$, $m_{1.g,l}$ and $m_{2.g,l}$ are the random numbers for h^{th} variable at the l^{th} variable between 0 and 1, l is the iteration number, g is the design variable varying from 1 to p, and h is the candidate solutions varying from 1 to q. The standard equation of JOA is given as equation (4), which is modified using the Taylor series. In order to inherit the advantages of Taylor series in JOA, and to overcome the drawback associated with the JOA, Taylor series concept is integrated with the JOA. The Taylor series uses the best solutions of the previous iterations or the historical solutions to obtain the current best solution. The priority order of the test case to be executed for finding the fault in a program is selected using the expression obtained from the Taylor integrated JOA. The standard equation of the Taylor series is expressed as,

$$\begin{aligned} J_{g,h,l}^{s+1} = {} & 0.5J_{g,h,l}^{s} + 1.3591J_{g,h,l}^{s-1} - 1.3590J_{g,h,l}^{s-2} + 0.6795J_{g,h,l}^{s-3} - 0.2259J_{g,h,l}^{s-4} + 0.0555J_{g,h,l}^{s-5} \\ & - 0.0104J_{g,h,l}^{s-6} + 1.38e^{-3}J_{g,h,l}^{s-7} - 9.92e^{-5}J_{g,h,l}^{s-8} \end{aligned} \quad (5)$$

where, $J^{s}, J^{s-1}, \ldots, J^{s-8}$ is the time series, J^{s-8} is the solution in the $(s-8)^{th}$ iteration, J^{s+1} is solution at the $(s+1)^{th}$ iteration.

$$J_{g,h,l}^{s} = \frac{1}{0.5}\left[\begin{array}{l} J_{g,h,l}^{s+1} - 1.3591J_{g,h,l}^{s-1} + 1.3590J_{g,h,l}^{s-2} - 0.6795J_{g,h,l}^{s-3} + 0.2259J_{g,h,l}^{s-4} - 0.0555J_{g,h,l}^{s-5} \\ \quad + 0.0104J_{g,h,l}^{s-6} - 1.38e^{-3}J_{g,h,l}^{s-7} + 9.92e^{-5}J_{g,h,l}^{s-8} \end{array}\right] \quad (6)$$

$$J_{g,h,l}^{s} = \left[\begin{array}{l} 2J_{g,h,l}^{s+1} - 2.7182J_{g,h,l}^{s-1} + 2.718J_{g,h,l}^{s-2} - 1.359J_{g,h,l}^{s-3} + 0.4518J_{g,h,l}^{s-4} - 0.111J_{g,h,l}^{s-5} \\ \quad + 0.0208J_{g,h,l}^{s-6} - 2.76e^{-3}J_{g,h,l}^{s-7} + 19.76e^{-5}J_{g,h,l}^{s-8} \end{array}\right] \quad (7)$$

Equation (7) is the simplified expression of the Taylor series. Substitute equation (7) in equation (4) for the integration of the Taylor series in JOA,

$$\begin{aligned} J_{g,h,l}^{s+1} = {} & 2J_{g,h,l}^{s+1} - 2.7182J_{g,h,l}^{s-1} + 2.718J_{g,h,l}^{s-2} - 1.359J_{g,h,l}^{s-3} + 0.4518J_{g,h,l}^{s-4} \\ & - 0.111J_{g,h,l}^{s-5} + 0.0208J_{g,h,l}^{s-6} - 2.76e^{-3}J_{g,h,l}^{s-7} + 19.76e^{-5}J_{g,h,l}^{s-8} \\ & + m_{1.g,l}\left(J_{g,best,l} - \left|J_{g,h,l}^{s}\right|\right) - m_{2.g,l}\left(J_{g,worst,l} - \left|J_{g,h,l}^{s}\right|\right) \end{aligned} \quad (8)$$

$$\begin{aligned} J_{g,h,l}^{s+1} = {} & 2.7182J_{g,h,l}^{s-1} - 2.718J_{g,h,l}^{s-2} + 1.359J_{g,h,l}^{s-3} - 0.4518J_{g,h,l}^{s-4} \\ & + 0.111J_{g,h,l}^{s-5} - 0.0208J_{g,h,l}^{s-6} + 2.76e^{-3}J_{g,h,l}^{s-7} - 19.76e^{-5}J_{g,h,l}^{s-8} \\ & - m_{1.g,l}\left(J_{g,best,l} - \left|J_{g,h,l}^{s}\right|\right) + m_{2.g,l}\left(J_{g,worst,l} - \left|J_{g,h,l}^{s}\right|\right) \end{aligned} \quad (9)$$

Equation (9) is the standard expression of the Taylor-JOA algorithm that involves in the prioritization of the test cases. This expression is the update equation of the proposed Taylor-JOA, which performs the test case prioritization such that the test cases possessing higher priorities enable in finding the faults that are present in a program with the loop by loop execution covering the entire branches with minimum execution time.

v) Termination:

The process is continued, till the prioritization of the test cases with the completion of all iterations. The Pseudocode of the proposed Taylor-JOA is shown in algorithm 1.

Algorithm. 1 Pseudocode of proposed Taylor-JOA algorithm

	Proposed Taylor-JOA Algorithm
1	**Input:** Design variables, candidate solutions, stopping condition
2	**Output:** Best solution $\rightarrow J_{g,best,l}$
3	**Start**
4	**Initialization**
5	Initialize Design variables, candidate solutions, stopping condition.
6	Repeat till l^{th} iteration
7	Choose the solution with maximum fitness measure as the worst solution
8	Choose the solution with minimum fitness measure as the best solution
9	For all g
10	Generate random variables
11	For all k
12	Modify the solution using equation (9)
13	Update the best solution based on fitness measure
14	end
15	end
16	**Stop**

IV. RESULTS AND DISCUSSIONS

In this section, the outcomes produced using the proposed Taylor-JOA based test case prioritization model and the conventional test case prioritization methods are explained. The performance of the proposed Taylor-JOA model is analyzed using two measures, such as APFD and APBC.

A. Evaluation metrics

The effectiveness of the proposed Taylor-JOA based test case prioritization model is analyzed using two measures, such as APFD and APBC and the metrics are described in the section 3.2.2 in detail. The software's [26] used in the development of the proposed Taylor-JOA based test case prioritization model is described as below,

Print_tokens: It is a C language of Ad Hoc test type with seeded fault type of size 726 LOC, 18 Procedures (20,934,932 total bytes) developed originally by Tom Ostrand and the colleagues at the Siemens Corporate Research.

Print_tokens 2: It is a C language of test type named Ad Hoc with seeded fault type of size 570 LOC, 19 Procedures (20,088,511 total bytes) developed by Tom Ostrand and the colleagues at the Siemens Corporate Research.

Schedule: Similar to print_tokens, it is a C language of Ad Hoc test type with seeded fault type of size 412 LOC, 18 Procedures (15,692,125 total bytes) designed by Tom Ostrand and the colleagues at the Siemens Corporate Research.

Schedule 2: The schedule2 differs from schedule only in size, and the size of schedule2 is 374 LOC, 16 Procedures (17,836,149 total bytes).

Space: It is a C language of test type named Ad Hoc with real fault type of size 6199 LOC, 136 Procedures (268,950,867 total bytes).

B. Comparative methods of test case prioritization

The comparative methods considered are, random sequence approach [2], Whale optimization algorithm (WOA) [6], Ant Colony Optimization (ACO) [5], Jaya Optimization Algorithm (JOA) [23], and the proposed Taylor-JOA method. The performance of the proposed Taylor-JOA method is analysed and are evaluated using two metrics, namely APFD and APBC.

Comparative analysis of the methods of test case prioritization based on print_tokens

The comparative analysis of methods involved in the test case prioritization using print_tokens based on APFD and APBC for various population sizes are depicted in figure 3. Figure 3.a shows the analysis based on APFD for various population sizes. For the population size of 15, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.9717, 0.985, 0.9683, 0.995, and 0.995, respectively. Similarly, when the population size is 20, the APFD for random sequence is 0.975, WOA is 0.9816, ACO is 0.955, JOA is 0.985, and the proposed Taylor-JOA 0.995. When the population size is 25, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.9817, 0.9717, 0.965, 0.965, and 0.9917, respectively. Figure 3.b shows the analysis based on APBC for various population sizes. For the population size of 15, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.925, 0.925, 0.895, 0.945, and 0.947, respectively. Similarly, when the population size is 20, the APBC for random sequence is 0.935, WOA is 0.941, ACO is 0.919, JOA is 0.937, and the proposed Taylor-JOA is 0.951. When the population size is 25, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.949, 0.955, 0.873, 0.953, and 0.967, respectively.

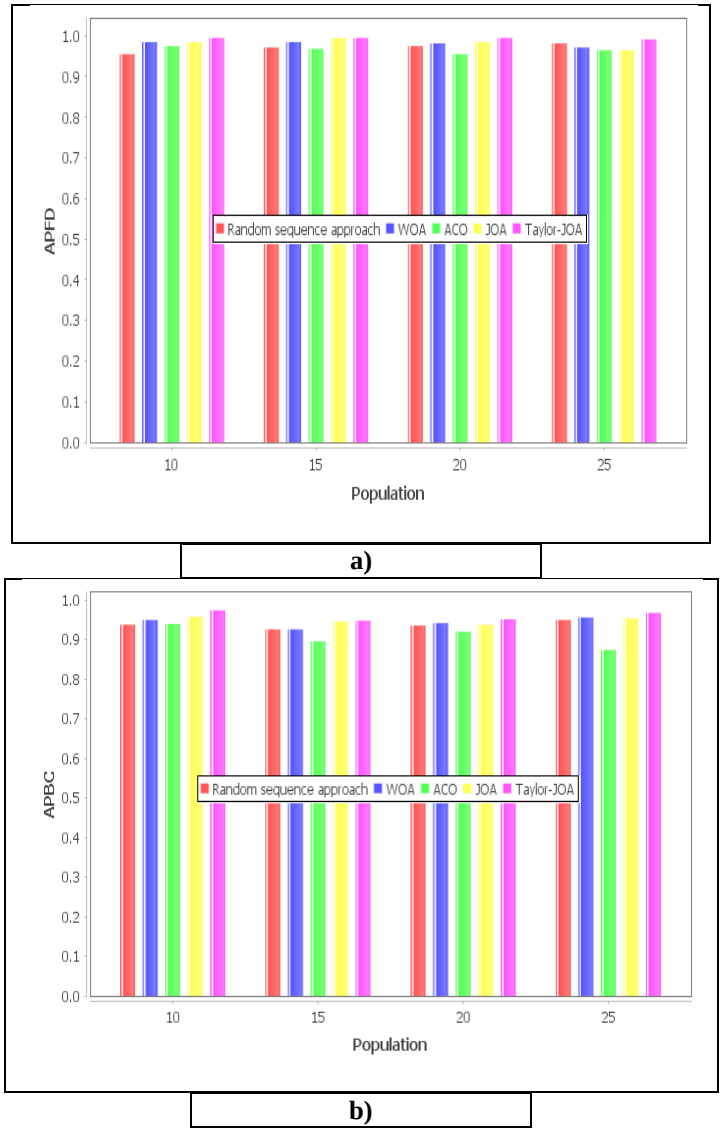

Fig. 3 Comparative analysis using print_tokens based on a) APFD, and b) APBC

Comparative analysis of the methods of test case prioritization based on print_tokens 2

The comparative analysis of the methods involved in the test case prioritization using print_tokens2 based on APFD and APBC for various population sizes are depicted in figure 4. Figure 4.a shows the analysis based on APFD for various population sizes. For the population size of 15, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.97, 0.96, 0.96, 0.975, and 0.99, respectively. Similarly, when the population size is 20, the APFD for random sequence is 0.96, WOA is 0.98, ACO is 0.915, JOA is 0.975, and the proposed Taylor-JOA is 0.985. When the population size is 25, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.93, 0.965, 0.915, 0.925, and 0.985, respectively. Figure 4.b shows the analysis based on APBC for various population sizes. For the population size of 15, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.99, 0.995, 0.97, 0.98, and 0.995, respectively. When the population size is 20, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.965, 0.995, 0.995, 0.995, and 0.995, respectively. Similarly, when the population size is 25, the APBC for random sequence is 0.995, WOA is 0.99, ACO is 0.995, JOA is 0.99, and the proposed Taylor-JOA is 0.995.

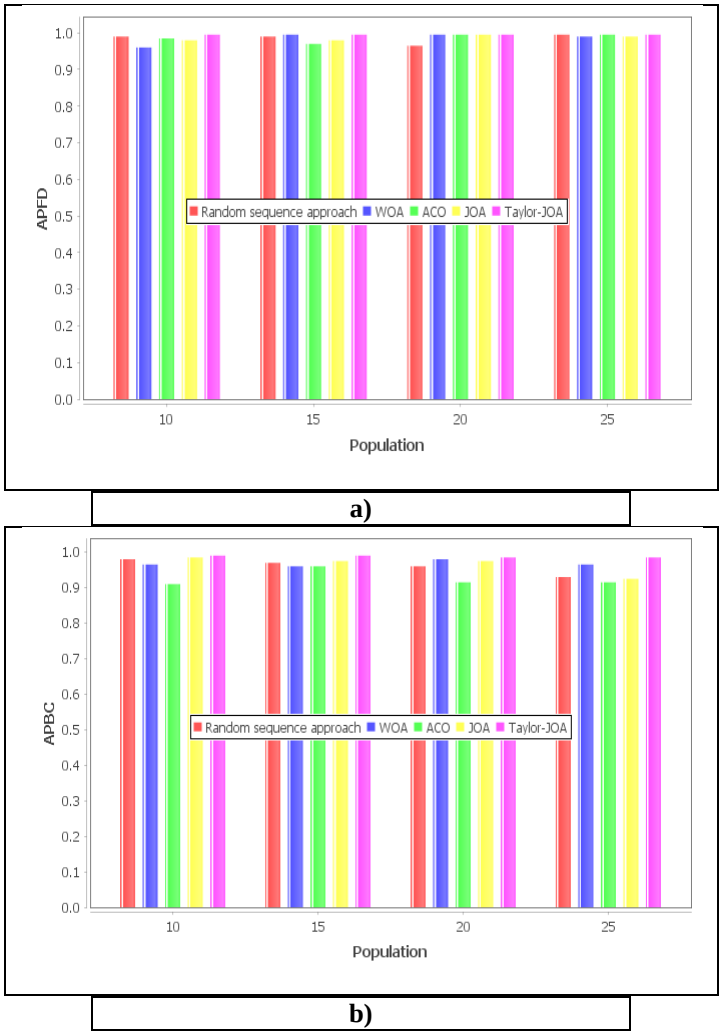

Fig. 4 Comparative analysis using print_tokens 2 based on a) APFD, and b) APBC

Comparative analysis of the methods of test case prioritization based on schedule

The comparative analysis of methods involved in the test case prioritization using schedule based on APFD and APBC for various population sizes are depicted in figure 5. Figure 5.a shows the analysis based on APFD for various population sizes. For the population size of 15, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.985, 0.9817, 0.965, 0.9783, and 0.9883, respectively. When the population size is 20, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.9917, 0.9983, 0.9717, 0.985, and 0.9983, respectively. Similarly, when the population size is 25, the APFD for random sequence is 0.975, WOA is 0.9717, ACO is 0.9483, JOA is 0.985, and the proposed Taylor-JOA is 0.9917. Figure 5.b shows the analysis based on APBC for the variation in the population size. For the population size of 15, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.963, 0.953, 0.937, 0.967, and 0.981, respectively. When the population size is 20, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.959, 0.927, 0.881, 0.965, and 0.979, respectively. Similarly, when the population size is 25, the APBC for random sequence is 0.957, WOA is 0.951, ACO is 0.901, JOA is 0.949, and the proposed Taylor-JOA is 0.967.

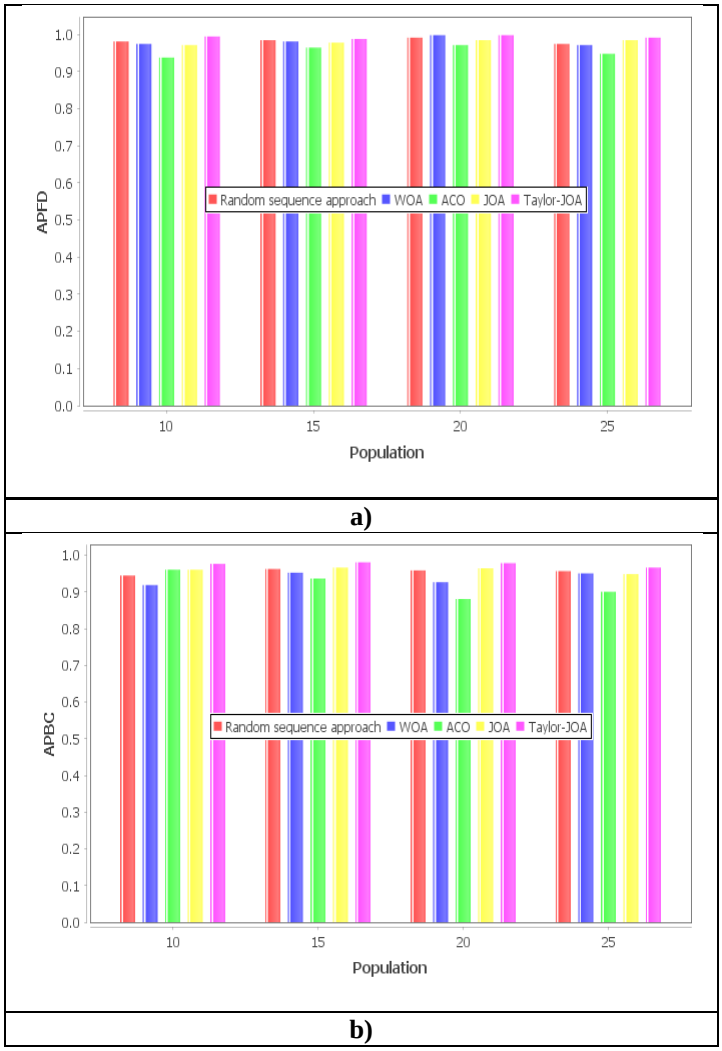

Fig. 5 Comparative analysis using schedule based on a) APFD, and b) APBC

Comparative analysis of the methods of test case prioritization based on schedule 2

The comparative analysis of methods involved in test case prioritization using schedule2 on the basis of APFD and APBC for various population sizes are depicted in figure 6. Figure 6.a shows the analysis based on APFD for various population sizes. For the population size of 15, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.9817, 0.9917, 0.9583, 0.9917, and 0.9917, respectively. Similarly, when the population size is 20, the APFD for random sequence is 0.985, WOA is 0.9917, ACO is 0.945, JOA is 0.995, and the proposed Taylor-JOA is 0.995. When the population size is 25, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.985, 0.9883, 0.9417, 0.9583, and 0.9917, respectively. Figure 6.a shows the analysis based on APBC for various population sizes. For the population size of 15, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.9683, 0.9733, 0.9667, 0.9783, and 0.9783, respectively. When the population size is 20, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.9683, 0.975, 0.96, 0.9683, and 0.9833, respectively. Similarly, when the population size is 20, the APBC for random sequence is 0.9733, WOA is 0.9667, ACO is 0.96, JOA is 0.96, and the proposed Taylor-JOA is 0.9817.

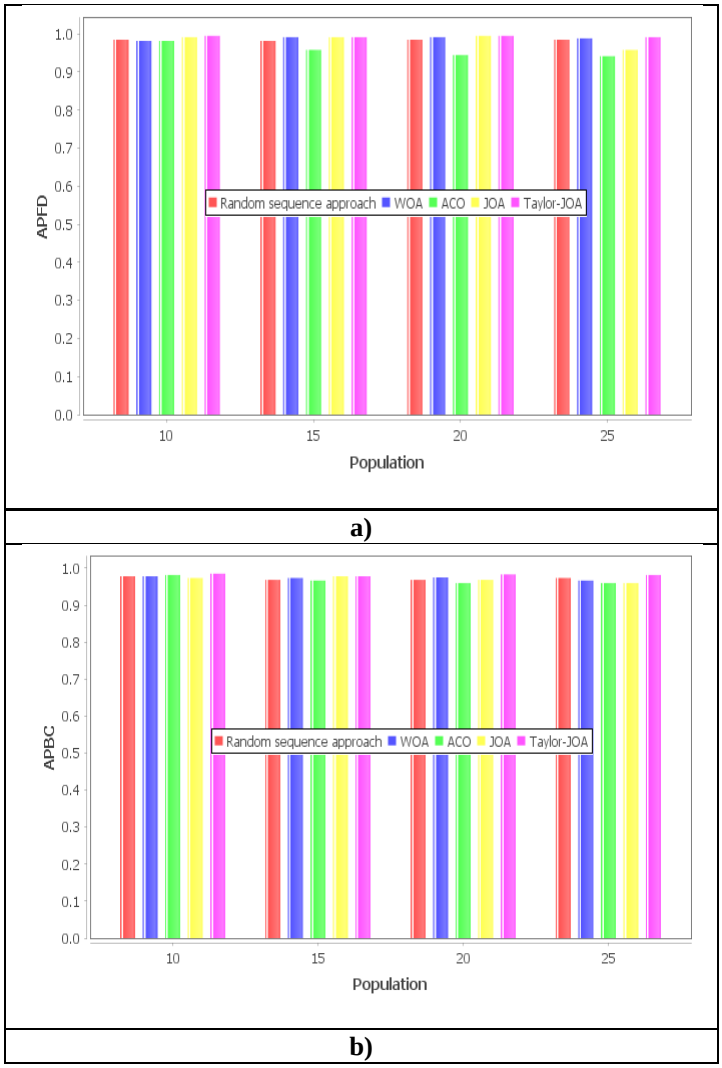

Fig. 6 Comparative analysis using schedule 2 based on a) APFD, and b) APBC

Comparative analysis of the methods of test case prioritization based on space

The comparative analysis of methods involved in the test case prioritization using space depending on APFD and APBC for various population sizes are depicted in figure 7. Figure 7.a shows the analysis based on APFD for various population sizes. For the population size of 15, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.99, 0.985, 0.99, 0.985, and 0.995, respectively. When the population size is 20, the APFD for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.98, 0.97, 0.945, 0.995, and 0.995, respectively. Similarly, when the population size is 25, the APFD for random sequence is 0.995, WOA is 0.99, ACO is 0.96, JOA is 0.98, and the proposed Taylor-JOA is 0.995. Figure 7.a shows the analysis based on APBC for the various population sizes. For the population size of 15, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.985, 0.9817, 0.9783, 0.9683, and 0.9917, respectively. When the population size is 20, the APBC for the methods, such as random sequence, WOA, ACO, JOA, and the proposed Taylor-JOA is 0.9583, 0.985, 0.9883, 0.9683, and 0.9917, respectively. Similarly, when the population size is 25, the APBC for random sequence is 0.9883, WOA is 0.9617, ACO is 0.9817, JOA is 0.985, and the proposed Taylor-JOA is 0.9917.

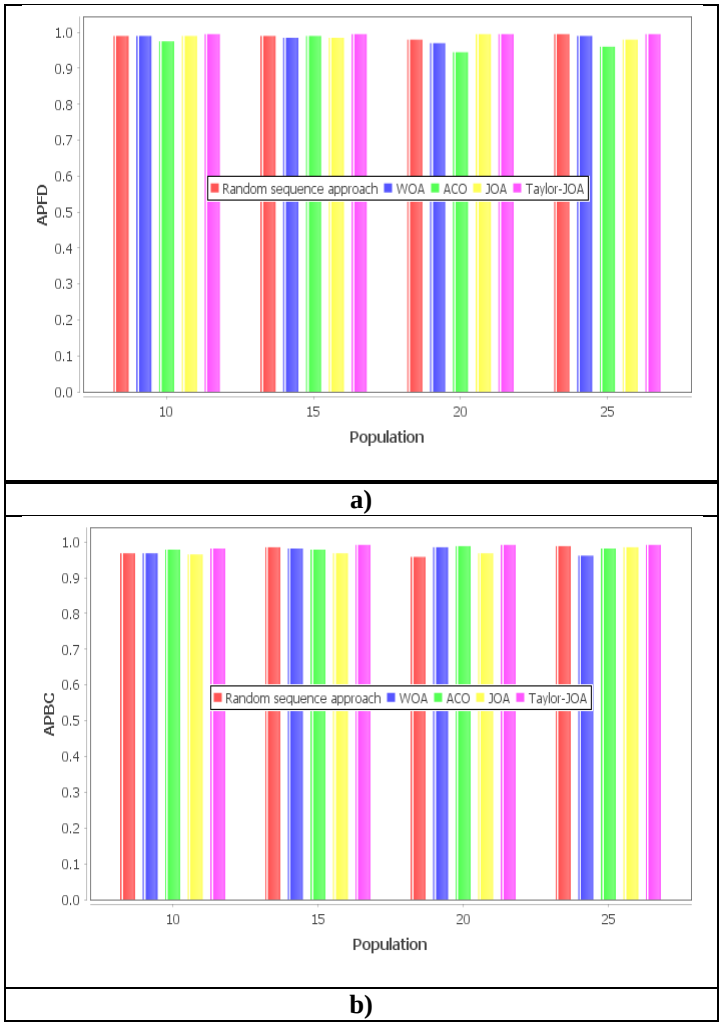

Fig. 7 Comparative analysis using space based on a) APFD, and b) APBC

C. Comparative Discussion

Table 1 depicts the comparative discussion of methods involved in test case prioritization in terms of evaluation metrics, namely APFD and APBC. The APDF of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject print_tokens is 0.975, 0.9817, 0.955, 0.985, and 0.995, respectively. The APBC of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject print_tokens is 0.937, 0.949, 0.939, 0.957, and 0.973, respectively. Similarly, the APDF of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject print_tokens2 0.99, 0.96, 0.985, 0.98, and 0.995, respectively. The APBC of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject print_tokens2 is 0.97, 0.96, 0.96, 0.975, and 0.99, respectively. In the same way, the APDF of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject schedule is 0.975, 0.9717, 0.9483, 0.985, and 0.9917, respectively. The APBC of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject schedule is 0.963, 0.953, 0.937, 0.967, and 0.981, respectively. The APDF of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject schedule2 is 0.985, 0.9817, 0.9817, 0.9917, and 0.995, respectively. The APBC of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject schedule2 is 0.9733, 0.9667, 0.96, 0.96, and 0.9817, respectively. The APDF of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject Space is 0.99, 0.99, 0.975, 0.99, and 0.995, respectively. The APBC of the methods, such as random sequence approach, WOA, ACO, JOA, and the proposed Taylor-JOA using the subject Space is 0.985, 0.9817, 0.9783, 0.9683, and 0.9917, respectively.

Table. 1 Comparative discussion of the methods test case prioritization

Subjects	Metrics	Methods				
		Random sequence	WOA	ACO	JOA	proposed Taylor-JOA
print_tokens	APFD	0.975	0.9817	0.955	0.985	**0.995**
	APBC	0.937	0.949	0.939	0.957	**0.973**
print_tokens2	APFD	0.99	0.96	0.985	0.98	**0.995**
	APBC	0.97	0.96	0.96	0.975	**0.99**
schedule	APFD	0.975	0.9717	0.9483	0.9850	**0.9917**
	APBC	0.963	0.953	0.937	0.967	**0.981**
Schedule2	APFD	0.985	0.9817	0.9817	0.9917	**0.995**
	APBC	0.9733	0.9667	0.96	0.96	**0.9817**
Space	APFD	0.99	0.99	0.975	0.99	**0.995**
	APBC	0.9850	0.9817	0.9783	0.9683	**0.9917**

V. CONCLUSION

Regression is termed as the retesting of unchanged parts of an application, where the test cases are executed again so as to analyze whether the earlier action of the application is acting well and the new modifications does not create any faults. This paper proposes a test case prioritization method in the regression testing using an optimization algorithm called as Taylor series-based Jaya Optimization Algorithm (Taylor-JOA), which is the hybridization of Taylor series in Jaya Optimization Algorithm (JOA). The test optimal cases are chosen on the basis of two constraints, termed as fault detection and branch coverage, and the fitness function is formulated in terms of these constraints. The performance of the proposed Taylor-JOA is analyzed based on the evaluation metrics, such as Average Percentage of Fault Detected (APFD) and the Average Percentage of Branch Coverage (APBC). The proposed Taylor-JOA produced the APFD of 0.995, and the APBC of 0.9917 that shows the superiority of the proposed Taylor-JOA. In future, the performance will be enhanced through the application of any other hybrid optimizations and with the inclusion of additional objective constraints.

REFERENCES

1. Yi Bian, Zheng Li, Ruilian Zhao, and Dunwei Gong, "Epistasis Based ACO for Regression Test Case Prioritization," IEEE Transactions on Emerging Topics In Computational Intelligence, vol. 1, no. 3, pp. 213-223, June 2017.
2. Jinfu Chen, Lili Zhu, Tsong Yueh Chen, Dave Towey, Fei-Ching Kuo, Rubing Huang, and Yuchi Guo, "Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering," Journal of Systems and Software, vol. 135, pp. 107-125, January 2018.
3. Breno Miranda and Antonia Bertolino, "Scope-aided test prioritization, selection and minimization for software reuse," Journal of Systems and Software, vol. 131, pp. 528-549, September 2017.
4. Soumen Nayak, Chiranjeev Kumar, and Sachin Tripathi, "," Arabian Journal for Science and Engineering, vol. 42, no. 8, pp. 3307–3323, August 2017.
5. Deepak Panwar, Pradeep Tomar, Harshvardhan Harsh and Mohammad Husnain Siddique, "Improved Meta-Heuristic Technique for Test Case Prioritization," Soft Computing: Theories and Applications, pp 647-664, 2017.
6. S. K. Harikarthik, V. Palanisamy, and P. Ramanathan, "Optimal test suite selection in regression testing with test case prioritization using modified Ann and Whale optimization algorithm," Cluster Computing, pp 1–10, 2017.
7. Sushant Kumar and Prabhat Ranjan, "ACO based test case prioritization for fault detection in maintenance phase," International Journal of Applied Engineering Research, vol. 12, Nno. 16, pp. 5578-5586, 2017.
8. Aitor Arrieta, Shuai Wang, Urtzi Markiegi, Goiuria Sagardui, and Leire Etxeberria, "Employing Multi-Objective Search to Enhance Reactive TestCase Generation and Prioritization for Testing Industrial Cyber Physical Systems," IEEE Transactions on Industrial Informatics, vol. 14, no. 3, pp. 1055 - 1066, March 2018.
9. Zainab Sultan, Shahid Nazir Bhatti, Rabiya Abbas, and S. Asim Ali Shah, "Analytical Review on Test Cases Prioritization Techniques: An Empirical Study," (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 8, no. 2, pp. 293-302, 2017.
10. Glenford J. Myers, "The Art of Software Testing," Wiley, New York, 1979.
11. IEEE Standard Glossary of Software Engineering Terminology, IEEE standards collection, 1990 Accessed from, http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf.
12. Thillaikarasi Muthusamy and Dr. Seetharaman.K, "Effectiveness of test case prioritization techniques based on regression testing," International Journal of Software Engineering & Applications (IJSEA), vol.5, no.6, pp. 113–123, November 2014.
13. Qi Zhang, Ludmila Cherkasova, Ningfang Mi, and Evgenia Smirni, "A regression-based analytic model for capacity planning of multi-tier applications," Cluster Computing, vol. 11, no. 3, pp. 197–211, 2008.
14. Hyunsook Do,Siavash Mirarab, Ladan Tahvildari, and Gregg Rothermel, "The effects of time constraints on test case prioritization: a series of controlled experiments," IEEE Transaction on Software Engineering, vol. 36, no. 5, pp. 593–617, 2010.
15. R.Krishnamoorthi and S.A.Sahaaya Arul Mary, "Factor oriented requirement coverage based system test case prioritization of new and regression test cases," Information and Software Technology, vol. 51, no. 4, pp. 799–808, 2009.
16. Sreedevi Sampath, Renee Bryce, and Atif M Memon, "A uniform representation of hybrid criteria for regression testing," IEEE Transaction on Software Engineering, vol. 39, no. 10, pp. 1326–1344, 2013.
17. C.P.Indumathi and K.Selvamani, "Test cases prioritization using open dependency structure algorithm," Procedia Computer Science, vol. 48, pp. 250– 255, 2015.
18. Luciano S.de Souza, Ricardo B.C.Prudencio, Flavio de A.Barros, and Eduardo H.da S.Aranha, "Search based contrained test case selection using execution effort," Expert systems with applications ,vol. 40, no. 12, pp. 4887- 4896, 2013.
19. Amanda Schwartz and Hyunsook Do, "Cost-effective regression testing through adaptive test prioritization strategies," Journal of Systems and Software, vol. 115, pp. 61–81, 2016.
20. Ahlam Ansari, Anam Khan, Alisha Khan, and Konain Mukadam, "Optimized regression test using test case prioritization," Procedia Computer Science, vol. 79, pp. 152–160, 2016.

21. Charitha Hettiarachchi, Hyunsook Do, and Byoungju Choi, "Risk-based test case prioritization using a fuzzy expert system," Information and Software Technology, vol. 69, pp. 1–15, 2016.
22. BoJiang and W.K.Chan, " Input-based adaptive randomized test case prioritization: a local beam search approach," Journal of Systems and Software, vol. 105, pp. 91–106, 2015.
23. R. Venkata Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems", International Journal of Industrial Engineering Computations, Vol. 7, No. 1, pp. 19-34, 2016.
24. Zheng Li, Mark Harman, and Robert M. Hierons, "Search Algorithms for Regression Test Case Prioritization", IEEE Transactions on Software Engineering, vol. 33, no. 4, pp. 225-237, April 2007.
25. S. Alamelu Mangai, B Ravi Sankar, K. Alagarsamy, "Taylor Series Prediction of Time Series Data with Error Propagated by Artificial Neural Network", International Journal of Computer Applications, vol. 89, no. 1, February 2014.
26. Softwares, printtokens, printtokens2, schedule, schedule2,andspace "https://sir.csc.ncsu.edu/php/showfiles.php?lang%5B%5D=C&name= &min_ver_cnt=&max_ver_cnt=&min_src_size=&max_src_size=&min_unit_cnt=&max_unit_cnt=&display=Display"accessed on march 2019.