International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I1P202
Volume 69 Issue 1, 5-16, January 2021

An Efficient Requirement-based Test Case Prioritization Technique using Optimized TFC-SVM Approach

Omdev Dahiya[#1], Kamna Solanki[*2]

[#]*Department of Computer Science and Engineering, University Institute of Engineering and Technology, Maharshi Dayanand University Rohtak, India*

[1]Omdahiya21792@gmail.com, [2]Kamna.mdurohtak@gmail.com

Abstract *- Software testing is an essential and challenging part of the SDLC (Software Development Life Cycle). Requirement-based TCP (Test case prioritization) is a method to optimize the execution time, cost, and effort as an essential part of the regression testing. It is a technique used to arrange the test cases (TCs), and sorting of the TC's is based on some criteria. It is established to execute the high priority test cases initially to reduce the execution time, efforts, and cost during the software testing. Thus, conventional TCP (Test Case Prioritization) is motivated to design for testing the software to enhance prioritization efficiency. TCP permits the testers to classify the test cases as the priority for performing the test execution. It helps in enhancing software quality. In the existing research, the authors had developed a method to prioritize the optimal test cases using the firefly approach. They used the firefly algorithm to optimize the ordering of the test cases and fitness value (FV), defined through the same distance model, to have better performance. The firefly's approach may be more efficient in determining fault proneness problems, which is intensely required in security-critical schemes. Thus, the proposed research deals with the processing of the non-linear approach that provides high classification rates. A TFC-SVM algorithm is a novel approach deal with the CUCKOO optimization in collaboration with SVM, used to achieve higher classification rates in terms of high mean, median, and low minimum value. Afterward, training and testing modules are considered through the classification approach and processed the requirements-based in TCP. The proposed model has resolved the existing issues such as error rates, high priorities, and maximum execution time to prioritize processed requirement-based on test case prioritization. The proposed parameters are evaluated using computation time, APFD, Mean, Standard Deviation, Min, and Max values through which the performance metrics can be achieved for the robust proposed system.*

Keywords — *Requirement-based Test Case Prioritization, TFC-SVM method, FA (Firefly Algorithm), and APFD metric.*

I. INTRODUCTION

Testing is an activity performed to uncover the errors in a software system. Testing reduces the rate of uncertainty about the software quality system [1]. When an application is tested, a test suite is built to improve the functionality. Testers reserve the test suite for future usage [2]. When the modifications are done in the system, then the pre-defined test suites are applied by testers to assure that no new faults or errors are familiarized with the code that has been tested. When modifications take place in the system, then each test is re-executed for every module after the relevant modifications [3]. Also, it is a costly method to execute complete test cases once modifications are done. Thus, to reduce the regression testing cost, requirements-based test case prioritization (TCP) has been introduced by the researchers. In requirement-based TCP, whole test cases (TC) are organized to strengthen some good performance [4]. Moreover, to develop priorities of the test case, definite factors depend on the requirement to be selected, and priority is assigned to test cases [5]. The main goal of the requirement-based TCP is to improve the probability that if the TCs are prioritized, it may meet a specific goal within stipulated time and cost [6]. Requirement-based TCP addressed a wide variety of goals such as: (i) The software programmer or tester intends to improve the fault detection rate (ii) Early-stage detection of the high-risk faults in the Test life cycle (TLC) (iii) To improve the probability of regression faults related to substantial code modifications early in the testing procedure (iv) To improve the code coverage (CC) program at a fast rate. (v) To build more reliable software [7]. The test case prioritization (TCP) method includes the test case selection that exposes maximum faults in software components and assigns a high priority to test cases with less execution time [8]. The execution of the prioritized test cases is more appropriate for testing the functionalities of software at minimum time. Therefore, Test case prioritization aims to decrease testing costs. Test Case Prioritization methods are random prioritization, complete branch coverage, and additional

coverage prioritization. Different algorithms for TCP have been established by different researchers [9]. The algorithms are GA (greedy algorithm), AGA (additional greedy algorithm), GA (genetic approach), and ACO (ant colony optimization), etc. The analysis of the current work is to implement and authorize a requirement-based system-level TCP method to expose errors or faults at the initial phase and to enhance the customer perceived Software Quality (SQ). The last view shows the following components are measured to pattern the user has assigned a novel requirement-based TCP method such as (i) Priority of needs (ii) Software developer perceived program implementation convolution (iii) Needs Modification and (iv) Error impact.

II. SUMMARY OF CONTRIBUTION

The summary of the research work contribution is described as follows: (i) initially, related work is systematically written, and analysis of regression test case prioritization (TCP) methods used for software development is performed. (ii) The comparative analysis of numerous TCP methods to verify the advantages and disadvantages of various approaches is done. (iii) Design and development of the novel requirement-based TCP method for regression testing (RT) using the Nature-Inspired Cuckoo-Search Optimized SVM (Support Vector Machine), which is called a TFC-SVM method, is performed. (iv) The proposed requirement-based TCP approach is evaluated and analyzed with the Firefly algorithm (FA) in [17]. The proposed model has improved the average percentage of fault detection (APFD) rate and reduces the execution time compared to the firefly algorithm.

Sections are described as follows: Section 1 explained the overview of the requirement-based TCP methods, and a survey of various articles is done in section 2. The firefly optimization (FA) algorithm is described in section 3. The proposed work is elaborated in section 4. Experimental result analysis is done in section 5, including the data set description and performance parameters are analyzed for the proposed and existing methods. Then, the conclusion and future scope are defined in section 6.

III. RELATED WORK

This section reviews several articles on requirements-based test case prioritization methods. It includes several techniques, tools, parameters, advantages, and disadvantages. It also includes the numerical results observed by the respective authors by comparing their proposed work with the previous methods. In the surveyed papers, most previous methods prioritize the test cases using some coverage data gathered with significantly additional efforts. The most widely used performance parameters in the surveyed papers are the average percentage of fault detected (APFD). *Wang et al., (2019)* developed a location-based TCP approach using the gravitational technique [10]. This research used a better approach to develop a new position in the software surrounded by mobile devices using the gravitational rule technique. Initially, the test gravitation was discussed, relating to the concept of worldwide gravitation. After that, a unique computational model of the test case gravitation was considered for the smart mall situation. A method to generate a fault test case set was developed using pseudo code. Then, location-based TCP using gravitation law was developed using test case data, fault data, location data to prioritize the test cases. Experimental results demonstrated that the novel test case prioritization method had demonstrated better performance than conventional test case prioritization methods. Moreover, local data and device level was a significant factor that influenced prioritization efficiency. Experiment outcomes defined that the average percentage of fault detection rate (APFD) of location-based test case prioritization method was 78.67 percent, which was higher than the base-line techniques. *Dhiman et al., (2019)* proposed research on manual and automated slicing for TCP to detect large faults from the scheme, where the modifications were done from the novel version announcement [11]. Thus, slicing was the method that separated the complete function-wise and identified the connected function. Hence, the performance of the present and current algorithm in the model was analyzed by considering the ten projects. Every project comprised seven functions and four modifications, which were defined in RT (regression testing). The simulation outcomes observed that the average percentage of fault detection value was improved. The execution was decreased with the automatic TCP execution compared to the manual TCP technique in RT (regression testing). The proposed model achieved an accuracy rate of 89 percent compared to the manual slicing (81%) method in test case prioritization. *Mohd-Shafie et al., (2020)* implemented a model-based TCP that recovered the fault recognition, assessment of RT (regression-testing) [12]. It associates the development of two current survey models, whereas integrated an extra arrangement principle enhances the prioritization rate. Empirical research was done to compute and compare the developed technique's result with the specific models from the survey using the APFD metric. Thus, three web-based presentations were used as the research object to achieve the mandatory tests that comprised the prioritized tests. It was observed from the outcomes that the APFD metric has better performance as compared to previous models that were 91.67 percent, 86.5 percent, and 91.2 percent for three web-services. Hence, it indicates that the developed method was more efficient in estimating the early faults during testing. It demonstrated that the planned techniques improve the fault detection performance of regression testing. *Xiao et al., (2020)* developed a new-TCP method using LSTM (long short-term memory) based DL (deep learning) to acquire reliable regression testing for fixed software on continuous integration [13]. Long short-term memory was the time-sequence prediction approach. It may

forecast the possibility of every TC detection, fault in the subsequent cycle following test data of existing continuous integration cycles. Thus, the priority of the test case was acquired vigorously under the guidelines of possibility. Experiment analysis was done on two industrial data sets. Numerical results identified that the proposed model was compared with the departing test case prioritization models. The proposed models have better performance for embedded software as :(i) Enhance prioritization efficiency. (ii) Improve the fault detection value in a continuous integration environment and (iii) Reduce the test execution time by automatic deduction of out-dated test cases. *Afzal et al., (2019)* worked on the requirement-based TCP techniques. TCP arranges the test cases (TC's) to perceive maximum faults irrespective of a minimum test suite or selection of test cases [14]. They developed a method that used route density and division coverage to order the TCs based on the theory that multipart coding had maximum faults. Halstead's parameters were used to compute the route complication of the TC's. Thus, the main goal of the projected method was to improve the APFD of the test-suite. They analyzed that the average priority fault detection of the planned method was more beneficial than the branch reporting-based prioritization method. Thus, the change among the average percentage fault detection of code coverage and complication depends on analysis ranges from 2.7 to 42 percent. *Srikanth and Williams (2005),* considered a prioritization requirement for the test (PORT 1.0) method that represented the efficiency of the TCP at the phase of evaluating the four factors [15]. The test cases were ordered that depends on the requirement priority, which was achieved by assessing the factors: (i) Client Priority, (ii) Execution Complexity, (iii) Fault Proneness, and (iv) Requirement Unpredictability for every constraint. Test cases were mapped to requirements with a maximum priority that was arranged previously for the execution. They showed that the efficiency of the PORT 1.0 method for the four large sequencers showed the improvement value of the fault recognition rate and test efficiency. *Srikanth et al., (2016)* investigated the two aspects and applied prioritization based on the feature in various domains [16]. They aimed to present efficient prioritization methods that experts may develop with less effort. The proposed method involved analyzing and assigning rates to every requirement depending on the essential aspects, client's priority, and fault proneness. The TC's for the highest requirements were arranged previously for the execution. They implemented two requirements-based TCP methods that used risk data on the system. Numerical analysis was done on the enterprise cloud application to measure the fault recognition of various test suites, prioritized based on the client priority and fault proneness. *Khatibsyarbini et al., (2019)* proposed TCP using the firefly (FA) optimization approach [17]. They applied a firefly algorithm with the defined fitness function method (FFM) for optimizing the arrangement of the test cases that were defined using the same distance model. The experimental analysis defined that the firefly optimization (FA) approach has a better average percentage fault detected (APFD) scores than other prioritization methods. It also demonstrated that the firefly algorithm (FA) has better LBS (Local Beam Search) in average execution time. Results obtained from the average percentage fault detected identified that the firefly algorithm may be more efficient in inventing the fault proneness problems required in security and critical situations. It can be said that for systems dealing with the computation of a huge amount of data, very efficient systems are required. As systems are running using software-based applications, they must be reliable and quality-oriented, which can only be ensured by testing the software systems [24-33]. Table 1 defines the comparison based on different parameters like existing methods, problems, benefits, and performance metrics and simulation tools used in the TCP (test case prioritization).

Table 1. Comparative Analysis of existing methods on TCP (Test Case Prioritization)

Author Name, (Year), Reference number	**Technique Name**	**Advantages**	**Problems**	**Tools /Performance Parameters**
Wang *et al.*, (2019), [10]	Location-based TCP using the law - gravitation	Impact of occurred errors to prioritize test cases in test rounds.	TCP (Test Case Prioritization) Problem	Average Percentage Fault Detection Rate (APFD)
Dhiman and Chopra, (2019), [11]	Ant Colony Optimization	Maximum no. of faults detected from the project	A hard-combinatorial optimization problem	MATLAB, Accuracy, Precision, Execution Time, Recall, and Fault Detection

Mohd-Shafie, *et al.*, (2020), [12]	Model-based TCP with (SESOC)	Execution of the system model is faster Improve the test cases	Maximum Time consumption and cost	Average Percentage Fault Detection Rate (APFD)
Xiao *et al.*, (2020), [13]	Long Short-Term Memory (LSTM)	Decrease the faults and time cost	Execute the high detection rate	Accuracy, Precision, Recall, and APFD
Afzal *et al.*, (2019), [14]	Coverage based prioritization method	High factors of verifying the defects Quick processing	Large no. of faults or issues in the program	Average Percentage Fault Detected (APFD), Fault detected.
Srikanth and Williams, (2005), [15]	System-Level TCP and Multi-faceted TCP approach	Commercial benefits	Costly and time-consuming	TCP tool /ASPD (Avg. Severity of Fault Detection) and PFV (Prioritization Factor Value)
Srikant *et al.*, (2016) [16]	Coverage Based, Operational Profile Based and requirements-based prioritization (TCP)	Cost advantages of customers and quality management	High complexity	ReBaTe/ CP and FP (Customer Priority and Fault Proneness)
Muhammad *et al.*, (2019), [17]	Firefly Optimization and TD-IDF	Regression Testing might be proven benefits.	Difficult to predict which test cases will real reveal errors.	APFD, Mean, Median, Min, Max, SD, and Time execution / UNIX Programs.

IV. ALGORITHM USED IN TCP (TEST CASE PRIORITIZATION)

TCP is an approach that aims to arrange the test case so that high priority test cases as per some fitness values are executed earlier to uncover the maximum number of errors in the minimum time. The prioritization procedure presents criteria to plan test cases so that the maximum number of faults may be detected earlier. This section elaborated on the algorithm for test case prioritization to improve the fault detection rate for software schemes.

A. Firefly Optimization Algorithm Used in TCP

The light flashes from the firefly map out for fireflies. The light flash may be designed by linking them with the SF (selective function) to be enhanced, making it probable to design a novel optimization method. The assumptions in the firefly approach are described as follows [18]:

(i). The real fireflies are unisexual. Each firefly (FF) may attract to each other fireflies.

- Complete Fireflies (FF) may be attracted to other fireflies without any discriminant.
- For instance, there are five fireflies, and everyone gets attracted to each firefly available when connected.

(ii) The attractive nature of the firefly (FF) is directly proportional to the enchantment of FF.

- When a firefly is attracting another, the vibrant nature of the firefly becomes the priority between them to grade the enchantment or attractiveness.

(iii) Fireflies may transfer randomly if they do not search for more allure fireflies in adjoining areas.

- When more than two fireflies are having similar, reflecting light (brightness), the firefly may randomly move towards either direction.

Firefly algorithm is observed in spatial regions comprising diverse sizes with encouraging consistency and supremacy over other methods [19]. Firefly is a meta-heuristic method that estimates that the optimization issue is programmed as an agent's position, whereas the SF is fixed as light intensity. Hence, the central two deliberations in the firefly algorithm are; change of intensity termed as brightness and the discovery of interference between fireflies. It is estimated that the firefly's vision is selected through brightness that, in turn, is associated with encoded specific features. Hence, the attractive nature of fireflies in the search region is compared to the SF rate of fireflies. The firefly (FA) approach has been applied in Requirement based TCP (test case prioritization), the concern of

assumption explained previously in figure 1. The algorithm started with selective function origin. Besides, the computation of the adjacency of DM (distance matrix) among firefly representatives and its brightness is encoded to identify every firefly's enchantment. The succeeding motion of the firefly depends on the rate of brightness.

Thus, the motion ends once, when complete fireflies have been examined. However, the sweeping motions are recorded. All the motions are verified. Lastly, the best pattern of the fireflies is selected that depends on a shorter distance. The given table 2 represents the FF algorithm in Requirement based TCP (test case prioritization). The particular function defined the attractive nature of the firefly that demonstrates the test case resemblance weight and distinctiveness. The sweeping motion of the firefly is stored eventually. The suitable route is ultimately specified as the best test suite sequence. Using table 2, the demonstration of firefly modules is shown in figure 1. The 5 test cases are prioritized; the possibility of receiving the best priority organization is unique over five factorials (1/5)!. Every test case may work as a firefly agent, whereas the distance among every test case represents the attractive function between firefly agents. Hence, the firefly algorithm is used with the fitness function to search for the best priority organization.

Table 2. Component mapping of Firefly Algorithm

Firefly Module	TCP Module	Explanation
Firefly Representative (FA1, FA2,…)	Test case (TC1, TC2,…)	The firefly algorithm demonstrates a test case. The motion route of the firefly algorithm for other firefly algorithms may be stored as the test case arrangement in test case prioritization.
Firefly Attractiveness (Light and Distance)	Test case resemblance and variation weights and distinctiveness test case distance.	Firefly algorithm motion depends on the attractiveness that is parallel to the same weight and distinctiveness.
Spaces among Firefly	Test case distance	Firefly algorithm brightness is the reduction that depends on the moved distances.

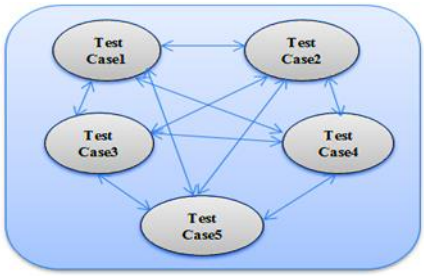

Fig. 1 The Scheme Representation of Test Case Distance Adjacency

The development of the FF algorithm in the TCP demonstrates the flow diagram of the firefly algorithm.

In the given figure 1, the information is interpreted from the criterion program, and test cases are extracted from the database. After that, the computation of test case distance is performed using ED (edit-distance) and SM (string-metric). Prioritization is initiated with the firefly algorithm's motion and the brightest non-located TC unless it reached the final FF in the search area. Therefore, the complete moved route demonstrates the prioritized TC arrangement. Hence, the short distance of the test case's complete arrangement is measured as the most acceptable route.

The main features of fireflies are mapped out numerically for a better performance rate. These are brightness, intensity rate, and attractiveness [20]. Hence, the attraction is identified using the brightness function. The selective function of the FF algorithm is the brightness function that is given in equation (i). Hence, the brightness rate of every FF for their motion in a solution area to optimize the traverse route.

$$F\left(n_{jk}\right) = \frac{(10*GF)}{((CYC_{\,l})*rand())} \ldots\ldots\ldots (1)$$

Hence, GF: Guidance Factor and CYC: input program. In eq (1), random value *rand ()* is used to create a random number between [0, 1], as it is a standard scale. It is a type

of off-set rate. The random number may be searched in a standard survey [21]. Such values are utilized by the selective function of the FF approach. Thus, the function used the scale metrics to distinguish the sizes with the moves of the firefly.

Firefly algorithm has two main advantages: (i) one is the automated sub-division and capability of dealing with the multi-modals [22]. (ii) Firefly algorithm depends on the attractiveness reduces with the distance. The whole population may sub-divide into the subgroups automatically, and every group may swarm around every mode. Thus, the best global solution is found from all the models. The subdivision permits the fireflies to search the complete optima if the population dimension is adequately maximum than the amount of the modes.

Table 3. Performance metrics with Firefly Algorithm

Parameters	**Firefly Algorithm**
Processing Time	Less
Classification	Population and Attractiveness based algorithm
Algorithm basis	Flash behavior of swarm and firefly
Performance	Better

V. PROPOSED WORK

In the proposed work, novelty deals with the nonlinear model's processing; this provides high classification rates. Also, the Cuckoo and SVM hybridization is used with the term frequencies for processing requirements-based TCP that is processed in the proposed method and shows the novelty of the proposed work by achieving the highest classification rates in terms of high mean, median, and low standard deviations. The proposed model steps are described below:

Step I: Firstly, the data set will be uploaded using GUI. The GUI is an essential task that is useful in the man-machine interface. The GUI used is the user interface tools built-in MATLAB for useful information gathering and easy visualization of the data processing.

Step II: Then, the pre-processing will be performed using data mining to get useful information. Data mining includes the normalization of the information, which is to be occurrences of the requirements in the data, which shows the significance of the requirement and the priority of the process to be executed in the minimum execution.

Step III: Then, we will perform the extraction of the term frequency features through which we will get the feature vectors. The feature vector will extract the characteristics in the form of the frequencies of the request to be executed in a test case to achieve high priorities of the processed requirements from the data.

Step IV: Then, the instance selection process will be performed through cuckoo search optimization. As the data have always been in the form of instances, the data's selection is a significant part of the processed requirement-based test case prioritization. This instance selection will perform the optimization in terms of optimizations weights and distance. We come to know the weightage of the processed data to be processed sequentially, which will reduce the redundancy of the data.

Step V: After instance selection, we will perform the training and testing in which the classification will take place. The classification will process the test cases, a priority level, which gives us classified priorities on the test cases. The classified output will generate the trained model for the high and low priorities for the processed test cases.

Step VI: Then, eventually, the performance will be evaluated as per the different parameters. The parameters are evaluated using computation time, APFD, Mean, Standard Deviation, Min, and Max values through which the performance evaluations can be achieved for the robust proposed system.

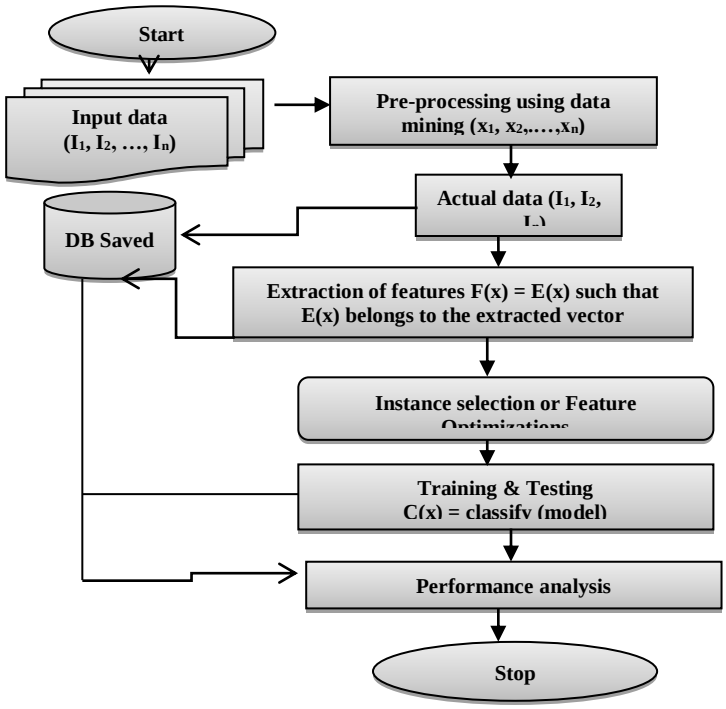

Fig. 2 The flow chart of the proposed Work

Proposed Work - Pseudo Code:

Step 1: Generate Data X_1, X_2, ……, X_n, such that x = data input sequence.

Step 2: Split X_n in such a way that S_x = {Split (X_n)} for all X_n.

Step 3: Tokenization of the S_x such that T → T_1, T_2, …….. T_n.

Step 4: Implement BoG (Bag of Words) with TFW ← Term Frequency Weights.

Step 5: Perform TFIDF← Term Frequency Inverse Document Frequency.

Step 6: Generate Population P such that P = T_1, T_2..... T_n.

Where T_n = Term Frequencies.

Step 7: While (t<MaxGen)

Evaluate quality Fitness

If (F_i > F_j)

Update New Solution

End if

Rank the solutions

Update with current fitness.

Until all process complete

Repeat

End while

Step 8: Generate optimize weight (wt.) values such that w = w_1, w_2, ……. W_n.

Step 9: For I =1: L (OptWt.)

Check Priorities based on T(x)

where T(x) = Min & Max values of test priorities

P(l) = T(x) for the current Level

end for

Step 10: Perform C_x = f{x← classify }, and evaluate the performance.

Step 11: Repeat Steps 1 and 10 until all iterations and processes are completed.

VI. EXPERIMENTAL RESULT ANALYSIS

This section involves a requirement-based TCP in regression testing of the dataset. It includes the no. of files and interrelated files. The used parameters to get the desired result are; average percentage fault detected, execution time, Min, Max, Mean, Median, and Standard deviation.

➢ *Dataset Description*

The Dataset taken is "Data files for Mahtab: Phase-wise Acceleration of Regression Testing for C" [23].

➢ *Performance Metrics*

The result performance metrics are described as below:

- **Average Percentage of Fault Detection Rate**

It is the measurement of the percentage of the detected fault for a test suite. The values range from zero to a hundred, and it has a better fault detection rate.

$$APFD = 1-\frac{Tf1+Tf2+\cdots Tfn}{n.m} + \frac{1}{2.n} \quad \ldots\ldots\ldots\ldots\ldots (3)$$

Here eq (3), t is a test suite, m is the number of faults detected at the test suite execution, n is the total amount of test cases, Tf1 is the start test position in the test-suite T, which identifies j.

- **Average Time Execution**

It is defined as the time taken to execute all the instructions for the test case.

$$\text{Average Time Execution: } \frac{T_{X*}N_J}{T_S} \quad \ldots\ldots\ldots (4)$$

Here eq (4) is defined as the time taken to execute all the instructions and total test cases.

- **Standard Deviation (SD)**

It is the measurement that demonstrates the variation from the mean. The standard deviation identifies the complex variation from the mean. It is the measurement of variability because of the return to real units of measurement of data.

$$SD\ (\sigma) = \sqrt{\frac{\sum_{j=1}^{n}(x_j - x')^2}{n}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Here eq (5), x determined unique value of the population represents the mean of all values; n is the total amount of values.

- **Median**

It is the sample of the middle point of the array when the total observation is odd.

$$\text{Median} = L+ \left(\frac{\frac{n}{2}-c.f}{f}\right) \text{ x c } \ldots\ldots\ldots (6)$$

Here eq (6), L is less limit of the median, N is total frequency, Cf is cumulative frequency, C is the class median interval, and F is median frequency class.

➢ *Simulation Analysis*

This section shows the experiment result in analysis with the research model (TFC-SVM) and the existing Firefly algorithm. It evaluated the performance metrics with all parameters such as Median, Mean, SD, Execution Time, APFD, Min, and Max compared with other methods

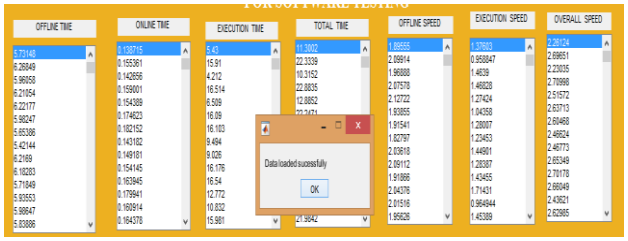

Fig. 3 Processed Requirement-based dataset uploading

The above figure 3 shows that the uploading process, which deals with the test case, includes offline time, online time, elapsed time, execution time, speed of the execution of the machine based on the load for the test suite. This is the training data through which the processing will be performed, and the evaluation will be done based on the extractions.

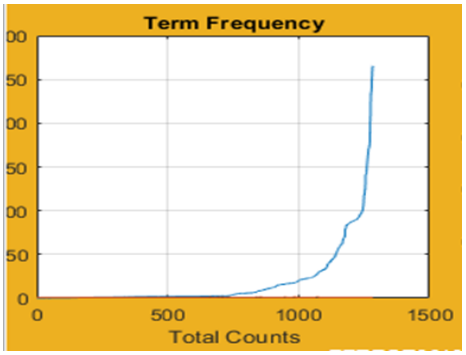

Fig. 4 Term Frequencies

The above figure 4 shows that the term frequencies, which act as a feature evaluation. Term Frequencies is one

of the significant parameters through which the occurrences of the errors in the test suits take place, and it can be noticed through these feature evaluations. This is the most crucial step, which tells us the frequencies and how vital it that test case errors to be resolved efficiently and also a key component of telling the relevance of the test case to be examined.

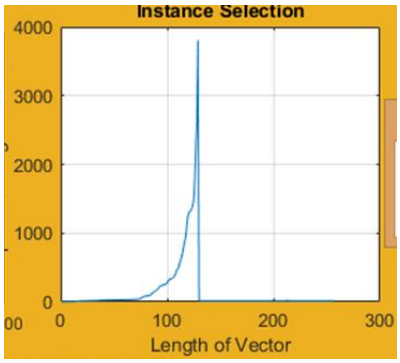

Fig. 5 Instance Selection using Cuckoo Search

The above figure 5 shows the instance selections process through the cuckoo optimization process, which shows the selections of the relevant features for each test case used to classify the priority. This is one of the crucial steps which will reduce the redundancy of the data and also reduces the execution time for the classification of the priority levels. This part helps prioritize the test case in an optimal way to have accurate prioritization of the test case.

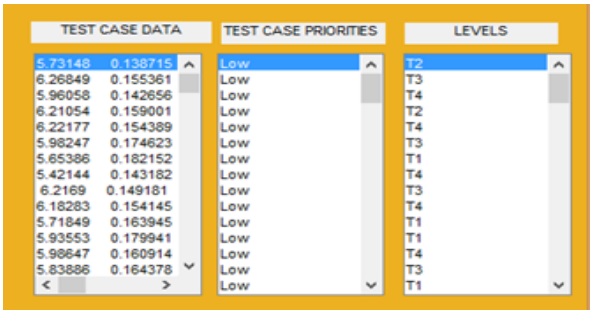

Fig. 6 Classifies the priorities

The above figure shows the classification, which is done using the SVM classification, and shows the different types of priorities of the test case to be resolved according to the priorities. The proposed approach can achieve efficient priority classification for low error rates and standard deviations. The T1 to T8 are the levels of the priorities used for the test cases to achieve severity of the test case to be resolved for the future performance on which the decisions will be made and through which the evaluations will be controlled to solve the priority of the test case.

Table 4. Proposed Performance with TFC-SVM Algorithm

Parameters	Values
APFD (%)	90.0232
Execution Time (ms)	108.4456
Max Value	0.994366
Min Value	0.187267
Mean	0.810716
Median	0.999526
SD	0.057313

Table 4 shows the performance of the proposed system, which are statistical terms and through which the performance will be measured to an extent for the test case priorities. The above table shows that the proposed approach can achieve high performance than the base approach in the overall development of the test case prioritization system.

Below, figure 7 shows the comparison of the mean between the FA and the proposed (TFC-SVM) approach, which is a normal distribution process. This must be high for low error rates, which will produce less variance from the mean (Stable) classification to have efficient priority classifications.

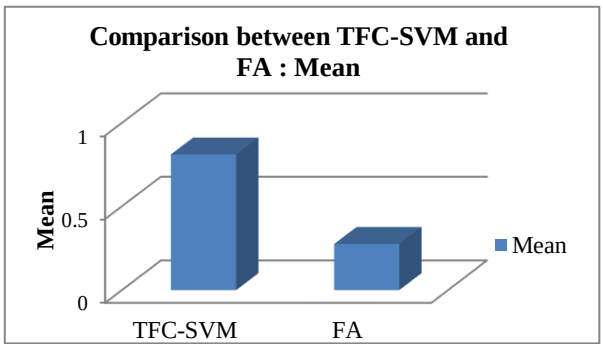

Fig. 7 Mean: Comparison between proposed (TFC-SVM) and Existing (Firefly Algorithm)

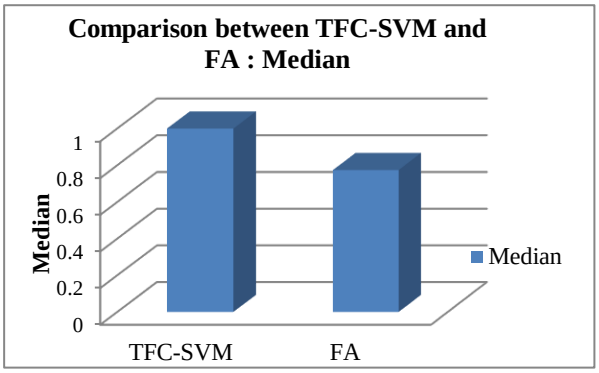

Fig. 8 Median: Comparison between proposed (TFC-SVM) and Existing (Firefly Algorithm)

The above figure 8 shows the comparison of the FA approach with the proposed (TFC-SVM) approach in terms of the median, which is one of the significant performance parameters. The median must be high and is useful in terms of stability to perform the classification of the priority levels of the test suits. Median evaluation is essential to signify how much high priority is required for each test case during the classification process. If the median is low, then the system classification error rates will be increased.

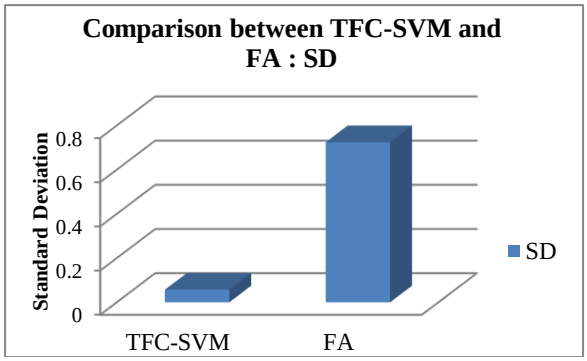

Fig. 9 Standard Deviation: Comparison between proposed (TFC-SVM) and Existing (Firefly Algorithm)

The above figure 9 shows the standard deviation performance comparison between the FA and proposed approach (TFC-SVM), which shows that the proposed approach is well efficient to achieve low deviations from the mean distribution through which the classification accuracy will be high to prioritize the levels.

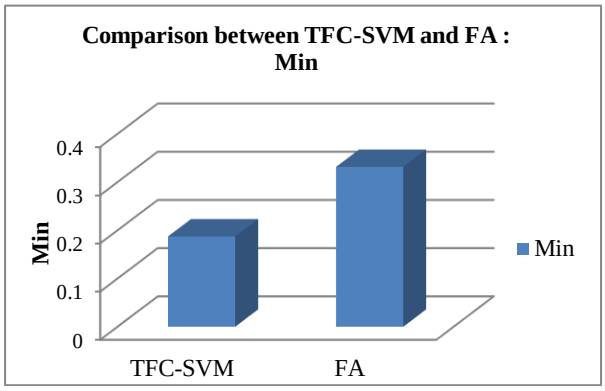

Fig. 10 Min: Comparison between proposed (TFC-SVM) and Existing (Firefly Algorithm)

The above figure 10 shows the comparison of the FA approach with the proposed (TFC-SVM) approach in terms of the min value, which must be low. If the min value is low, the system can detect early faults during the classifications to prioritize the test cases. It also minimizes the diversities among the classification of the levels of the priorities.

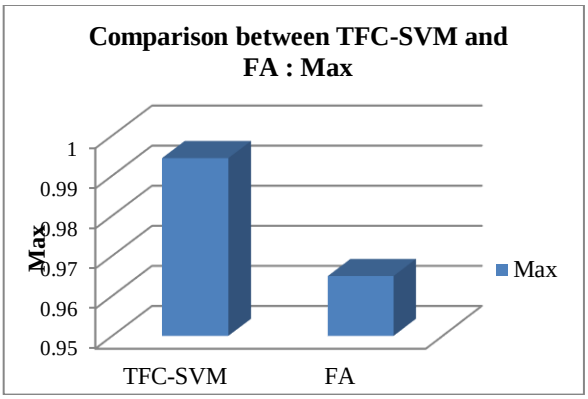

Fig. 11 Max: Comparison between proposed (TFC-SVM) and Existing (Firefly Algorithm)

The above figure 11 shows the comparison of the FA approach with the proposed (TFC-SVM) approach in terms of the max value, which must be high. It also maximizes the diversities among the classification of the levels of the priorities.

Below, figure 12 shows the execution time. It shows that the proposed approach can achieve low execution time. The existing approach (FA) is well-suited to achieve high computation time for the classification rate.

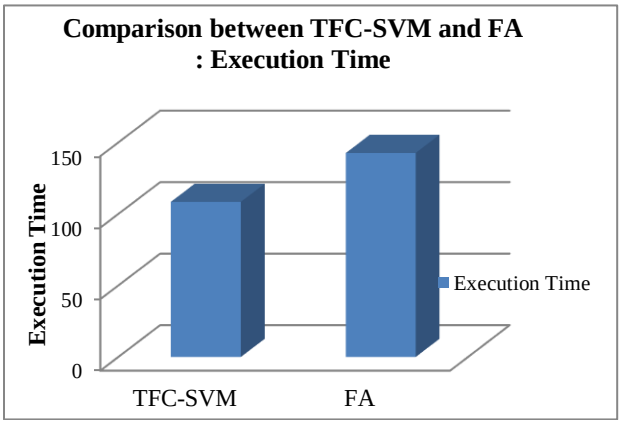

Fig. 12 Execution Time: Comparison between proposed (TFC-SVM) and Existing (Firefly Algorithm)

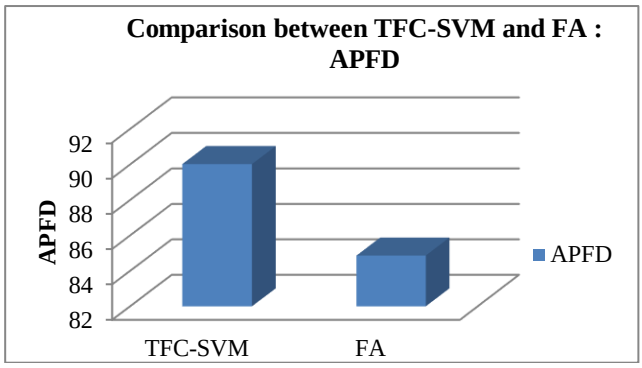

Fig. 13 APFD: Comparison between proposed (TFC-SVM) and Existing (Firefly Algorithm)

The above figure 13 shows the fault detection rate is one of the significant parameters which detects the proposed approach detection of faults in terms of performance parameters. It should be high for high-performance evaluation to detect more faults. If the system has a high AFPD value, then there will be high chances of the particular test case to be evaluated for the system to achieve high fault detections for the test case priorities.

Table 5. Comparison Analysis with various performance metrics

Parameters	TFC-SVM	Firefly Algorithm
APFD (%)	90.0232	84.868
Execution Time (ms)	108.44564	142.6667
Max Value	0.994366	0.964934
Min Value	0.1872674	0.331598
Mean	0.810716	0.275212
Median	0.999526	0.773856
SD	0.057313	0.724788

Table 5 shows the comparative analysis with performance evaluations for the proposed work with the Firefly algorithm and shows that the proposed work can achieve high performance for classifications efficiently to achieve low loss functions in case of priority levels with existing methods.

VI. CONCLUSION AND FUTURE SCOPE

It is concluded that the requirement-based test case prioritization prioritized the TC's, optimized the test implementation, saving time and cost. Requirement-based TCP is a technique of arranging the TC execution per some specific goals. Moreover, the requirement-based TCP goal is to improve the probability and fault detection rate (FDR). Requirement-based TCP may address a diversity of objectives; testers improve the FDR, increase the possibility of identifying faults before regression testing, testers improve the risk of faults detection and find the faults previously during the testing phase, testers improve the test speed for testing their coverage of coverable coding in the organization. This research has implemented the TFC-SVM model with an instance selection model using an improved cuckoo search optimization algorithm. This novel model has been used to attain a high classification rate.

The training and testing modules are done with the TFC-SVM Classification method using machine learning. The existing research issues are resolved using TFC-SVM methods and evaluate the performance metrics such as Min, Max, Mean, Median, Standard Deviation (SD), APFD, and Execution Time. An existing model using Firefly Algorithm has attained results such as APFD value 84.868 percent, Execution Time value is 142.66 ms, Max value is 0.964, Min Value is 0.331, Mean value 0.275, Median Value 0.773, and SD value 0.724. The proposed model has achieved a high APFD value of 90.02 percent, and execution time is 108.44 ms; Max, Min, Mean, Median, and SD Values are 0.994, 0.187, 0.810, 0.999, and 0.057. All the performance parameters are compared with the existing methods.

Overall, the APFD (average percentage fault detection) outcomes showed that the TFC-SVM method might become a high challenger in the test case prioritization field. The APFD outcomes define that the TFC-SVM algorithm may be efficient in determining FP (fault proneness) problems mandatory in security-critical schemes.

The upcoming directions can be: (i). It can introduce a novel optimum selection and prioritization method. (ii) It would be stimulating to search for probable improvement on this nature-inspired algorithm concentrating on coverage efficiency. (iii) It can implement a WOLF optimization with NN (Neural Network) algorithm to improve the different parameters like error and precision rate.

REFERENCES

[1] M. Khatibsyarbini, M. A. Isa, and D. N. A. Jawawi, Test case prioritization approach in regression testing: A systematic literature review.

[2] Information and Software Technology, 93(2)(2018) 74-93.

[3] G. J. Myers, T. Badgett, T. M. Thomas, C. Sandler, The art of software testing.. Chichester: John Wiley & Sons, 2,(2004).

[4] G. Rothermel, R. H. Untch, C. Chu and M. J. Harrold, Test case prioritization: An empirical study. In Proceedings IEEE International Conference on Software Maintenance-1999 (ICSM'99).' Software Maintenance for Business Change'(Cat. No. 99CB36360), IEEE. (1999) 179-188.

[5] D. Hao, L. Zhang, and H. Mei, Test-case prioritization: achievements and challenges. Frontiers of Computer Science 10(5) (2016) 769-777.

[6] O. Dahiya & K. Solanki, A systematic literature study of regression test case prioritization approaches, International Journal of Engineering & Technology, 7(4)(2018) 2184-2191.

[7] O. Dahiya, K. Solanki, S. Dalal, and A. Dhankhar, Regression Testing: Analysis of its Techniques for Test Effectiveness, International Journal of advanced trends in computer science and engineering, 9(1)(2020) 737-744.

[8] A. Ansari, A. Khan, A. Khan, and K. Mukadam, Optimized regression test using test case prioritization. Procedia Computer Science, 79(3)(2016) 152-160.

[9] J. A. P. Lima, & S. R. Vergilio, Test Case Prioritization in Continuous Integration environments: A systematic mapping study. Information and Software Technology,121(1)(2020) 106268.

[10] G. Duggal, & B. Suri. Understanding regression testing techniques. In Proceedings of 2nd National Conference on Challenges and Opportunities in Information Technology. (2008).

[11] X. Wang, H. Zeng, H. Gao, H. Miao, and W. Lin. Location-based test case prioritization for software embedded in mobile devices using the law of gravitation, Mobile Information Systems, (2019) 1-15.

[12] R. Dhiman, & V. Chopra,Novel Approach for Test Case Prioritization Using ACO Algorithm, In 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT), IEEE, 2(3)(2019) 292-295.

[13] M. L. Mohd-Shafie, W. M. N. Wan-Kadir, M. Khatibsyarbini and M. A. Isa, "Model-based test case prioritization using selective and even-spread count-based methods with scrutinized ordering criterion", Plos one, 15(2)(2020) 1-27.

[14] L. Xiao, H. Miao, T. Shi, and Y. Hong, LSTM based deep learning for spatial-temporal software testing, Distributed and Parallel Databases, 38(7)(2020) 687–712.

[15] T. Afzal, A. Nadeem, and M. Sindhu, Test Case Prioritization Based on Path Complexity, In 2019 International Conference on Frontiers of Information Technology (FIT), IEEE, (2019)363-3635.

[16] H. Srikanth, & L. Williams, On the economics of requirements-based test case prioritization.ACM SIGSOFT Software Engineering Notes, 30(4)(2005) 1-3.

[17] H. Srikanth, C. Hettiarachchi, and H. Do, Requirements based test prioritization using risk factors: An industrial study. Information and Software Technology, 69(2016) 371-83.

[18] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, HNA Hamed, and M.D.M. Suffian, Test Case Prioritization Using Firefly Algorithm for Software Testing, IEEE Access, 7(3)(2019) 132360-132373.

[19] P. R. Srivatsava, B. Mallikarjun, and X. S. Yang, Optimal test sequence generation using firefly algorithm, Swarm and Evolutionary Computation, 8(3)(2013) 44-53.

[20] N. Iqbal, K. Zafar, and W. Zyad, Multi-objective optimization of test sequence generation using multi-objective firefly algorithm (MOFA). In 2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE), IEEE, 3(4)(2014) 214-220.

[21] J. Nayak, B. Naik, and H. S. Behera, A novel nature inspired firefly algorithm with higher order neural network: performance analysis, Engineering Science and Technology, an International Journal. 19(1)(2016) 197-211.

[22] D. P. Mohapatra,Firefly optimization technique-based test scenario generation and prioritization, Journal of applied research and technology, 16(6)(2018) 466-483.

[23] X. S. Yang, & X. He. Firefly algorithm: recent advances and applications, International Journal of swarm intelligence, 1(1)(2013) 36-50.

[24] S. Mondal, Data files for Mahtab: Phase-wise Acceleration of Regression Testing for C, Mendeley Data, V2, DOI: 10.17632/7fvwj88jvm.2, (2019).

[25] O. Dahiya, K. Solanki, and A. Dhankhar, Risk-Based Testing: Identifying, Assessing, Mitigating & Managing Risks Efficiently In Software Testing, International Journal of advanced research in engineering and technology, 11(3)(2020) 192-203.

[26] K. Solanki, Y. Singh, & S. Dalal, A Comparative Evaluation of "m-ACO" Technique for Test Suite Prioritization, Indian Journal of science and technology, 9(30) 1-10.

[27] K. Solanki, Y. Singh, and S. Dalal, Experimental analysis of m-ACO technique for regression testing, Indian Journal of Science and Technology, 9(30) 1-7.

[28] O. Dahiya, & K. Solanki, Prevailing Standards in Requirement-Based Test Case Prioritization: An Overview, ICT Analysis and Applications, (2020) 467-474.

[29] O. Dahiya & K. Solanki, A Study on Identification of Issues and Challenges Encountered in Software Testing, International Conference on Communication & Artificial Intelligence (ICCAI-2020) 17th-18th September (2020).

[30] Palak, & P. Gulia, Ant Colony Optimization Based Test Case Selection for Component Based Software. Int. J. Eng. Technol, 7(4) (2018) 2743-2745.

[31] S. Dalal, & V. Dahiya, A novel technique-absolute high utility itemset mining (Ahuim) algorithm for big data, International Journal of advanced trends in computer science and engineering, 9(5) (2020) 7451–7460.

[32] A. Dhankhar, K. Solanki, A. Rathee and Ashish, Predicting Student's Performance by using Classification Methods, International Journal of advanced trends in computer science and engineering, 8(4) (2019).

[33] Hemlata & P. Gulia,Novel Algorithm for PPDM of Vertically Partitioned Data. International Journal of Applied Engineering Research, 12(12)(2017) 3090-3096.

[34] A. Chahal, & P. Gulia, Deep Learning: A Predictive IoT Data Analytics Method, International Journal of Engineering Trends and Technology, 68(7)(2020) 25-33.