Asian Journal of Applied Science and Technology (AJAST)
(Peer Reviewed Quarterly International Journal) Volume 2, Issue 4, Pages 103-112, Oct-Dec 2018

The Test Suite Selection in Performance Testing with the Test Case Prioritization by Modified ANN Method

Fanish Lal.S[1] and VR.Nagarajan[2]

[1]*Research Scholar, Department of Computer Science, Sree Narayana Guru College, Coimbatore - 641 105.*
[2]*Assistant Professor, PG & Research Department of Computer Science, Sree Narayana Guru College, Coimbatore - 641 105.*

Article Received: 25 May 2018 *Article Accepted: 27 September 2018* *Article Published: 08 October 2018*

ABSTRACT

The performance testing aims to identify the cases when the data suddenly demonstrates the aggravated features for definite integration of input data's. The performance testing needs to select a group of input data quickly repeatedly to analyze the performance blockage in various applications. The FOREPOST is emerged as a better solution for discovering the performance blockage by black box testing. This method is more effective and adaptive in finding the bottlenecks than random testing. The performance blockages can be determined by test cases written for the particular data. Even though the FOREPOST identifies the good and bad test cases it fails to discover the fault in the source program. The Kernel Fuzzy C-means Clustering (KFCM) algorithm is used to cluster the test cases to analyze these faults and to separate the appropriate and inappropriate test case which helps for prioritizing the test cases. The test case prioritization examines the maximum fault probability in the source program. The Modified Artificial Neural Network (ANN) classification algorithms are employed to prioritize the test cases whereas the Whale Optimization algorithm is utilized for weight optimization. The project work integrates the FOREPOST and modified ANN to identify the performance bottlenecks and faults in the source code.

Keywords: FOREPOST, KFCM, Modified ANN, Whale Optimization.

1. INTRODUCTION

The performance testing identifies the performance blockages, when the application is under test (AUT) suddenly displays the poor performance features for the particular input data. These types of performance problems can effectually determined by the situations where the AUT undergoes high response time or reduction in throughput unpredictably. The testing team is responsible for developing the test cases based on the performance which include certain actions (GUI objects interaction or the methods invoked for the interfaces exposed) and the input parameters of the GUI interface [11]. The effectual test cases are difficult to build in identifying the performance blockages in reduced time because the testing team requires testing more combination of input data and the actions for different applications.

The testing team as well as the developers requires some of the tools for performance analysis for discovering the performance bottlenecks to attain more performance while developing a product and to maintain the cost effectivity. The performance analysis is the first priority in many organizations and many studies have been illustrated that performance blockages are not quite simple to imitate and the developers have to spend more time to find them [8]. Moreover, the performance blockage fixing is tough and the efficient tools for placing and attaching the blockages need to be developed. Hence different group of teams have to work to identify the performance blockages on different tools. In performance testing, the input data has to be analyzed based on the requirements where the performances are determined regarding acceptance feature. Based on the acceptance criteria the test has to be planned and configure it. The test cases can be designed and implemented and hence the reports are taken to find the performance bottlenecks which is described in figure 1.

The prevailing tools in performance management help to organize and gather the data regarding the various applications and hence the shareholders can attain the performance insight. Unfortunately, none of these tools identifies performance bottlenecks automatically. The outlining of the source code is difficult and the level of high complexity turns out to be performance blockages where the loss of productivity occurs [6]. The different applications have 20% of the productivity loss

because of the downtime in application. The source program may not be existing for few components and testing team contemplate on black box testing on the performance of entire application than standalone component focus. The different application may have different characteristics based on the input data with respect to expenditure of the resource.

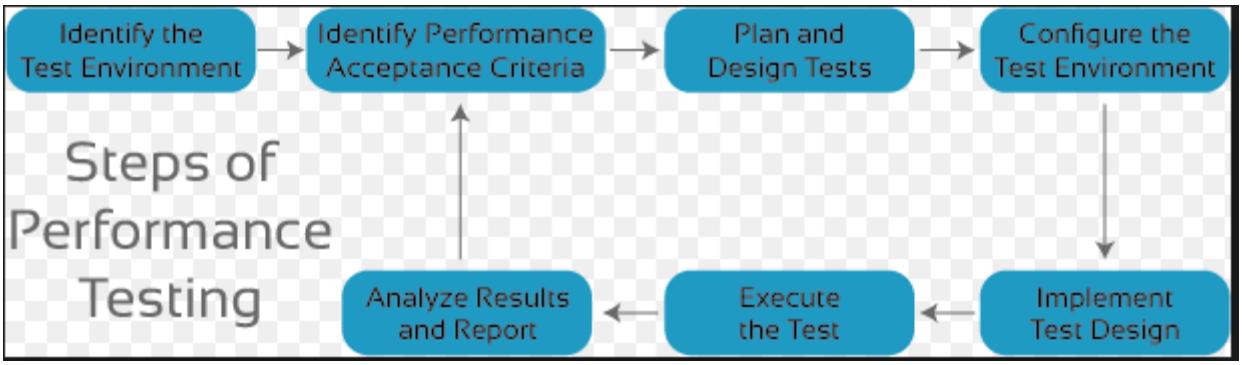

Figure No: 1 Steps in performance testing

These different characteristics may include intensive calculations which have more performance blockages and the testers are in the possession of summarizing the feature of AUT in terms of the source data. In this method, the input data can be selected which increases the resource expenditure significantly and also exposes the performance blockages. Identifying the sensitive rules that illustrate the properties of source data is extremely process in creativity which includes the deep knowledge in source domain [10]. There do some of the descriptive rules exist for opting the source data and this selection plays the major role because the descriptive rules estimate the AUT functionality. Sometimes the rules may become very simple to demonstrate the notations. These features may have more complexity that builds performance blockages in exposing the test cases.

Hence the better solution is afforded for analyzing the performance blockages termed as Feedback-Oriented Performance Software Testing (FOREPOST) by learning and the rules utilized can effectively demonstrates the group of source data which have more calculations. It is an adaptive and effective method of testing where the rules are learnt from the implementation of AUT outlines. This automatically selects the source data for performance testing. The more number of performance blockages can be identified by FOREPOST than random testing. The type of run time monitoring system is employed and this is done for short duration with the combination of machine learning and automated scripting. The performance associated measures can be minimized to low amount of descriptive rules that are assembled during AUT. This initiates performance insight in the testing source data and it also maximizes the load computation in applications [4]. The accuracy is evaluated by the FOREPOST method by identifying the injected and interpreted blockages.

The testing helps to determine the connected path of the program to determine the program accuracy. Performance testing is the feature in system to validate the modifications integrated with the proposed and they satisfy the requirements. Hence, performance testing includes the group of features that are exaggerated because of the alteration of a specific feature or the action. The alteration may happen when the bugs are resolved or by any improvements. The existing methods are not applied without adjusting straightly. The testing team will verify and validate the necessary characteristics of the development model. Test case prioritization is the major part in performance testing [1]. The test cases can be prioritized based on the importance, impact and the functionalities of the business. It helps to arrange the test cases for implementing in a way that maximizes the fault exposing likelihood while retesting the method.

The fault detection can be made early and the repeated updation may permit the developers to expose the problems while testing. The previous feedback method is afforded by prioritization which makes the debugging process easier for the developers. The probability of test case prioritization may be increased to run to significant test cases. The method for providing support to the test case prioritization is done on the particular criteria. This method helps to arrange the test cases by the test engineers and the test cases with more priority will be implemented first than lower one. The Test Case Prioritization (TCP) method does not eliminate the test cases but they just evade the disadvantages of test case minimization methods. There were many prioritization techniques used by the test cases that helps to illustrate and maximizes the effectiveness of improved rate of fault detection [3]. Hence the fault detection or localization method is required to enhance the level of performance testing. The performance bottlenecks can be identified by FOREPOST and test cases are clustered by Kernel Fuzzy C-means (KFCM) method to categorize the appropriate and inappropriate test cases. Then the test cases are prioritized with the help of Modified ANN algorithm by performing weight optimization.

The projected work affords the below mentioned contributions:

- FOREPOST is a simple method that assembles and employs the implementation outlines of the AUT to discover the rules that illustrate the computation power of the source data. The methods are very adaptive and effective because the implementation of AUT is based on the rules that are learnt newly.
- FOREPOST discovers the performance blockages where the AUT's performance is restricted by one or more components. The FOREPOST is applied to many applications and the performance bottlenecks can be found easily.
- Even though the FOREPOST identifies the good and bad test cases it fails to discover the fault in the source program. The Kernel Fuzzy C-means Clustering (KFCM) algorithm is used to cluster the test cases to analyze these faults and to separate the appropriate and inappropriate test case which helps for prioritizing the test cases. The test case prioritization examines the maximum fault probability in the source program. The Modified Artificial Neural Network (ANN) classification algorithms are employed to prioritize the test cases whereas the Whale Optimization algorithm is utilized for weight optimization. The project work integrates the FOREPOST and modified ANN to identify the performance bottlenecks and faults in the source code.

2. RELATED WORK

Schwartz et al. proposed the two Adaptive Test Prioritization (ATP) strategies that utilized the features of fuzzy Analytical Hierarchy Process (AHP) and the Weighted Sum Model (WSM). The comparative analysis of the ATP methods is presented. The methods are afforded by the researchers that are needed for performance testing and the accurate data to determine which method suits for the needs of the testing [13]. The methods described shows that it will enhance the effectiveness of the cost of performance testing. When there is a change in the software it must be sure that the other parts are not affected. The test cases that are prevailing are to be tested well and it can be resolved by test case prioritization.

Ansari et al. projected the optimized test case prioritization technique by the usage of Ant Colony Optimization (ACO) to minimize the effort, cost and the time needed to execute the performance testing and the revealed faults. Then the testing methods checks whether the developed software product satisfies the requirements. As a result, the quality assessment can be done on the basis of software development [14]. Huang et al. proposed the approach of test case prioritization by taking the historical data. The historical data is assembled and the projected genetic method is used to examine the effectiveness. The result shows that the method has enhanced the effectiveness of the fault detection [9].

Hettiarachchi et al. suggested the method to enhance the effectualness of the test case prioritization [15]. The requirement alteration features like complexity, status, security are taken as the risk factors and the method of fuzzy expert model assess the risks of the requirements. Pedemonte et al. proposed a Systolic Genetic Search (SGS) algorithm to resolve the Test Suite Minimization Problem (TSMP). The cost effectiveness of the testing can be further enhanced by prioritization and the popular method called "fixed-strength prioritization", helps to prioritize the test suite interaction by selecting the new test cases. The disadvantage is selecting a fixed strength than the multiple ones. The drawback can be solved by "aggregate-strength prioritization" which was given by Huang et al. It helps to integrate the interaction level of coverage at diverse strengths.

The remaining sections are organized as follows: section 3 describes the overview of performance testing; section 4 describes the overview of FOREPOST; section 5 describes the proposed work; section 6 describes the results and section 7 gives the references.

3. OVERVIEW OF THE PERFORMANCE TESTING

The performance testing is the method of producing the good test case. The performance testing helps the testing team to identify the test cases which has poor response time or poor throughput or latency. This part is attained by including more users to the AUT and this gives rise to the high computation and after identifying the source data where the AUT takes more number of resources and time to calculate the output. On the other hand, the type of poor test cases can be identified based on the utilization of limited resources which take few amount of time to implement while comparing to the effective test cases. The next method is to generate the rules for the input data selection after the detection of poor and effective test cases [5]. The system should have the capability of correcting on its own. This can be done by applying the learned rules that are selected on the input data on these rules and to validate the input data selection that gives expected performance outcomes. The probability can be maximized where the learned rules articulate the indisputable causation among the input data and performance associated measures.

The last thing is the performance testing is said to be incomplete when the enough clues are not afforded to the test engineers and in the AUT the difficulty hang about. The intention of the performance testing is to identify the performance issues for example, the minor issue may pull down the performance level of the whole application that is simple to discover by profilers. Nevertheless, the performance bottlenecks are difficult to identify because there are many approaches whose execution time is beyond the mentioned time that often occurs in large scale medium or applications [2]. The problem solved by this projected method is clustering of the input data into relevant and irrelevant teat cases that are prone to performance bottlenecks.

4. OVERVIEW OF FOREPOST

This section illustrates the two key factors on how the FOREPOST is constructed: The first thing is rule extraction from outline of the execution that explains the association between source data properties and testing performance workload that are implemented with this information. The second thing is bottleneck discovery by utilizing those rules.

4.1 Obtaining Rules

The AUT is instrumented and it is implemented by the few number of arbitrarily input data selection. The profile execution are gathered and clustered repeatedly into diverse groups that communally express the various performance AUT outcomes. There are two levels of two groups that are associated to the effective and poor test case performance. Its execution profiles are collected and automatically clustered into different groups that communally describe different performance results of the AUT.

The values defined for the AUT source data for both the good and bad test cases that denotes the source data to the classification method called Machine Learning. The rule learning method is selected by the FOREPOST termed as Repeated Incremental Pruning to Produce Error Reduction (RIPPER) to achieve the rules that direct the test source data selection written in test scripts [7].

RIPPER is called as a rule learning algorithm and it is customized version of the method called Incremental Reduced Error Pruning (IREP). The method combines the process of pre-pruning and post-pruning that is converted into learning phrase and a separate-and-conquer rule is followed. When it is created, every rule gets pruned and the procedure is quite similar to IREP. The difference is that it selects an alternate rule measure in the phase of pruning and it offers a new solution by optimizing the preliminary rule set that is achieved by IREP. The type of feedback loop is generated by the learned rules that are assisted and attained by classification algorithms of machine learning. The test scripts are used mechanically to supervise the input test data selection. The learned rules are newly generated and the mentioned test input data is segregated and the loop is repeated. The input data is selected based on the test scripts from the various segregated data by executing the AUT. The new rules generated can be learnt again from the gathered outlined implementation. When there are no rules to be learnt then the segregation of input test data is constant with higher probability level. Now the eradication of instrumentation can be performed and the testing is continued by selecting the input test data of the learned rules [12].

4.2 Identifying Bottlenecks

The testing team repeatedly discovers the performance bottlenecks when the method is called while execution may dangerously impinge on the entire performance of AUT. Consider the approach that is sporadically implemented by the thread and alterations of the some file has to be checked. This might be a bottleneck that is appealed in both good and worst test cases. Hence the part of resource expenditure is the important part of application method and it does not have any insights where the problem performance is resolved [17].

The second type of approach is considering the most important approaches that happen in effective test cases that are not called upon and not significant to poor test cases and the implication of the functional method of the resource expenditure is the trigger implementation [17]. The resource expenditure is termed as the standardized weighted sum of (a) count that the method is appealed, (b) the entire elapsed amount of time of its incantations minus the more number of elapsed time of all approaches that are appealed from this procedure, and lastly, (iii) the number of incantation methods that are generated from this approach. The performance blockages or bottlenecks can be discovered by the Independent Component Analysis (ICA) in this FOREPOST technique.

5. PROPOSED METHODOLOGY

The process of performance testing is to validate the enhancements that help to integrate the accurate work and to satisfy the requirements that are specified. Therefore, the performance testing involves the group of characteristics that have more impact on the feature alteration or any function. Recently, the test case prioritization difficulty has been included in organizing the test cases for performance testing to maximize the performance effectiveness. The already prevailing test case prioritization approaches are not capable of handling large amount of test input data and hence auxiliary cost effective outcomes cannot be attained by this approach. The above issue can be overcome by the proposed method. In the projected method, the set of test cases are produced. The generated test cases are clustered and classified as appropriate and inappropriate test cases. The clustering can be performed by the modified kernel fuzzy c means (MKFCM) algorithm and appropriate test cases that are

clustered are used for test case prioritization. The aim of this prioritization method is to examine the arrangement of the test cases that increases the fault identifying probability in the input data previously. The effective test case prioritization can be executed by MANN classification method. Then WOA is employed for weight optimization. Lastly, the final score is attained by the whale optimization method which helps to prioritize the test cases.

5.1 Test Case Prioritization

The test case prioritization is described by considering the following terms:

- T → Test case suite
- PT → The set of permutation functions of T
- F → function obtained from T (real numbers)

The problem is formulated by finding the T' ϵ PT where (∀T") (T" ϵ PT) (T"≠ T') | f(T')≥ f(T")|. Here,

- PT → Possible prioritization of the order T
- F → Used for the ordering to obtain award value

There are primary goals in attaining this type of prioritization. They are described below:

- The software engineers who involved in testing process may tend to maximize the fault detection rate which is called as likelihood of the test cases to disclose the faults while performing regression testing by the usage of those test suites.
- They may tend to maximize the code coverage rate at the faster level in the system which permits the code coverage measure to meet the needs priory in testing.
- The confidence level can be maximized while testing under reliable rate at the faster level.
- The likelihood of disclosing the faults can be maximized associated to the alterations of the particular change in the testing.

5.2 Kernel Fuzzy C-means Clustering

The appropriate and inappropriate test cases are prioritized and they are clustered by the kernel based FCM method. The different types of Various KFCM algorithmic methods expand the KFCM method with a diverse setting of kernel based learning. The projected method utilizes In this proposal, kernel fuzzy c means clustering procedure is occupied for clustering the already prevailing test cases depending on the coverage measure resemblance. Fuzzy c-means (FCM) is the clustering method that allows the information focused on the proximity of the group and it is used essentially in recognizing the pattern. The objective function of projected multiple kernel fuzzy c-means algorithm is effectually determined as,

$$F(M,A)=\sum_{i=1}^{N}\sum_{j=1}^{a}M_{ij}^{m}(1-K_{MK}(t_j,A_i))$$

The kernel based FCM is utilized to identify an error in advance. The end of the process is locating error where the source of the error is located. The classification of appropriate and inappropriate test cases is clustered by this method. The test cases are clustered and the appropriate test cases are taken for test prioritization. The intention of test case prioritization is to examine the test case arrangement that increases the level of probability to identify the faults in the input data.

5.3 MANN

The MANN algorithm is used for performance test case prioritization. In this approach, the former neural networks are altered by the whale optimization algorithm. Modified artificial neural networks function performs the following steps:

(1) The loads are fixed for every neuron other than the input layer neurons.

(2) The neural network is developed with the help of text as input data and HUa is hidden units and O as the output unit.

(3) The projected input layer Bias function is calculated as,

$$X = \beta + \sum_{n=0}^{H_U - 1} w_{(n)} T_1(n) + w_{(n)} T_2(n) + w_{(n)} T_3(n) + \ldots\ldots w_{(n)} T_m(n)$$

The classification method called MANN classification is used for test case prioritization. In the proposed modified artificial neural network, the weight optimization is performed by the Whale optimization algorithm.

5.4 Whale Optimization

The WOA is employed for the process of weight optimization. To conclude, the final score value is attained from the whale optimization algorithm and depending on the score value attained, the test case prioritization happens and it is figured in 2. The whale optimization includes three phases like encircling prey, bubble net attacking method and search for prey. Humpback whales are used for identifying the prey location and enclose them.

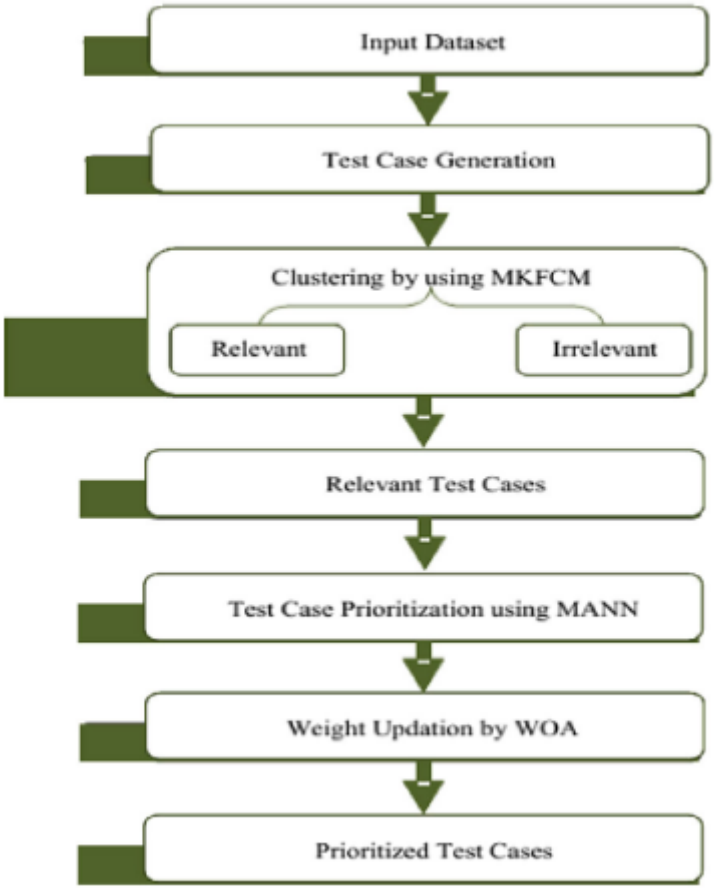

Figure No: 2 Test Case Prioritization Method

The best method can be obtained for the undetermined position of the most favorable design is the target prey or close proximity in the WOA algorithm. After the best agents are searched, then the updation regarding the position may occur. The similar kind of approach is used based on the discrepancy of a vector can be employed to investigate for prey (examination). The humpback whales explore randomly depending on the location or its position. Lastly, the WOA algorithm is eliminated by

the contentment of a termination method. Then the score value is attained by the whale optimization algorithm and depending on the score value the test case prioritization is obtained. The test suite prioritization is highly effectual in achieving the prioritization goal for P by the general test case prioritization and it becomes less effective over the progression of succeeding releases. To conclude, the difficulty of test case prioritization is utilized in the preliminary software testing.

6. RESULTS AND DISCUSSION

The comparative analysis of time and memory are analyzed by the existing FOREPOST and ANN whale method. The graphical representation of the comparison is shown in table no: 1 & 2, figure 3 & 4.

No. of Iterations	FOREPOST Method (in bytes)	Whale ANN Method (in bytes)
10	1,289,565	1,165,485
20	1,658,984	1,235,478
30	1,385,678	1,345,786
40	1,598,745	1,524,587

Table No: 1 Memory comparison

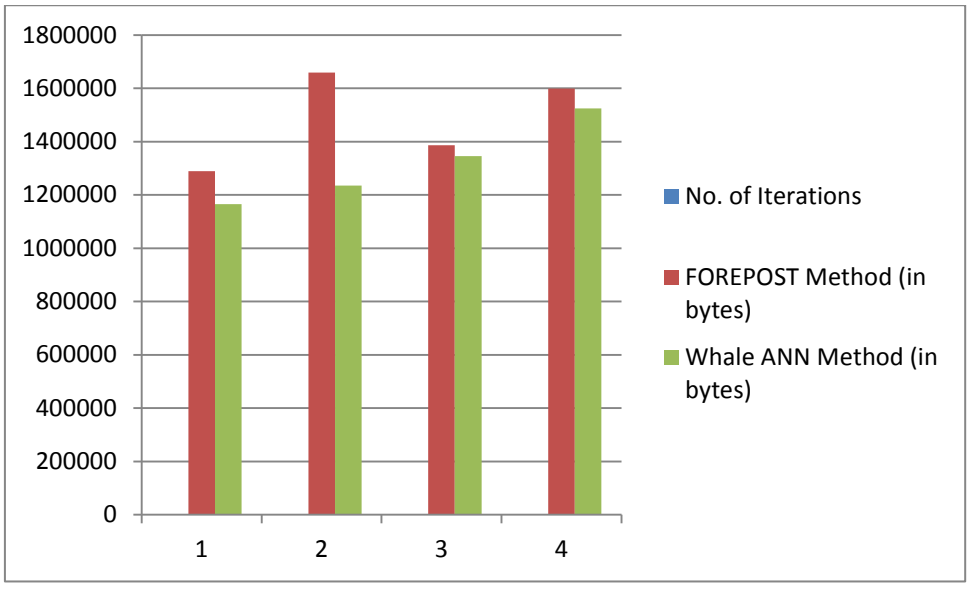

Figure No: 3 Memory comparisons

No. of Iterations	FOREPOST Method (ms)	Whale ANN Method (ms)
10	15542	12365
20	15695	14521
30	16573	16584
40	25475	21547

Table No: 2 Time comparison

Online ISSN: 2456-883X

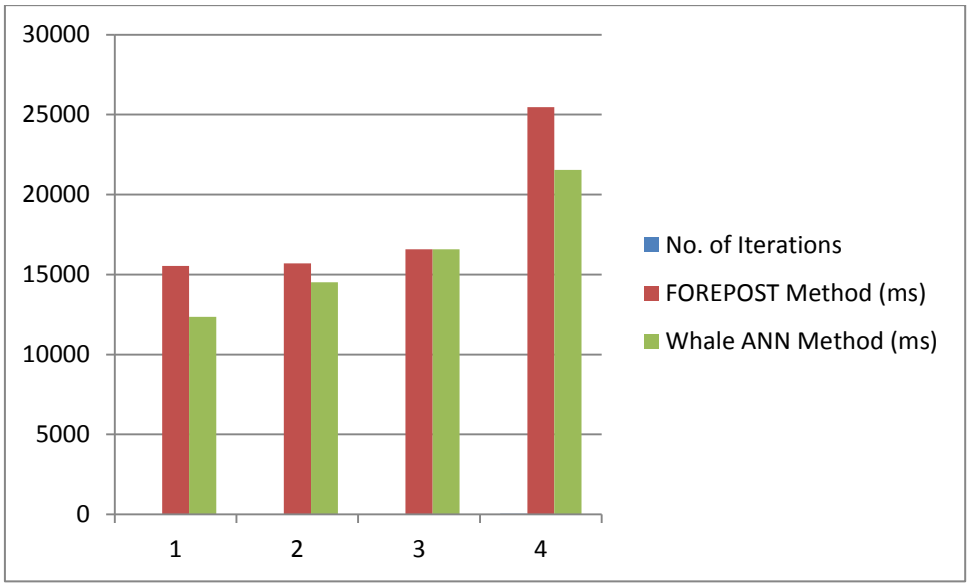

Figure No: 4 Time comparison

The FOREPOST is emerged as a better solution for discovering the performance blockage by black box testing. This method is more effective and adaptive in finding the bottlenecks than random testing. The comparative measures like time and memory are chosen and the result shows that the ANN whale method is very effective than FOREPOST method.

7. CONCLUSION

The test case prioritization technique helps to create the test cases that increase the cover for each criterion. The performance blockages can be determined by test cases written for the particular data. Even though the FOREPOST identifies the good and bad test cases it fails to discover the fault in the source program. The Kernel Fuzzy C-means Clustering (KFCM) algorithm is used to cluster the test cases to analyze these faults and to separate the appropriate and inappropriate test case which helps for prioritizing the test cases. The test case prioritization examines the maximum fault probability in the source program. The Modified Artificial Neural Network (ANN) classification algorithms are employed to prioritize the test cases whereas the Whale Optimization algorithm is utilized for weight optimization. The project work integrates the FOREPOST and modified ANN to identify the performance bottlenecks and faults in the source code.

REFERENCES

[1] Achenbach M, Ostermann K (2009) Engineering abstractions in model checking and testing. IEEE Intl Workshop SCAM:137–146

[2] Pedemonte, M., Luna, F., Alba, E.: A systolic genetic search for reducing the execution cost of regression testing. Proc. J. Appl. Soft Comput. 49, 1145–1161 (2016)

[3] Aguilera MK, Mogul JC, Wiener JL, Reynolds P, Muthitacharoen A (2003) Performance debugging for distributed systems of black boxes. In: SOSP, pp 74–89

[4] Ammann P, Offutt J (2008) Introduction to software testing. Cambridge University Press

[5] Ammons G, Choi JD, Gupta M, Swamy N (2004) Finding and removing performance bottlenecks in large systems. In: ECOOP, pp 170–194

[6] Bird DL, Munoz CU (1983) Automatic generation of random self-checking test cases. IBM Syst J 22:229–245

[7] Bishop CM (2006) Pattern Recognition and Machine Learning (Information Science and Statistics). Springer,Secaucus

[8] Briand LC, Labiche Y, Shousha M (2005) Stress testing real-time systems with genetic algorithms. In: Proceedings of the 7th annual conference on genetic and evolutionary computation, GECCO 05. ACM, USA, pp 1021–1028

[9] Huang, Y.-C., Peng, K.-L., Huang, C.-Y.: A history-based cost cognizant test case prioritization technique in regression testing. J.Syst. Softw. 85(3), 626–637 (2012)

[10] Grechanik M, Fu C, Xie Q (2012) Automatically finding performance problems with feedback-directed learning software testing. In: 34th international conference on software engineering (ICSE), pp 156–166

[11] Grindal M, Offutt J, Andler SF (2005) Combination testing strategies: a survey. Software Testing, Verification, and Reliability 15:167–199

[12] Group TY (2005) Enterprise application management survey. The Yankee Group Hamlet D (2006) When only random testing will do. In: Proceedings of the 1st international workshop on random testing, RT '06. ACM, USA, pp 1–9.

[13] Schwartz, A., Do, H.: Cost-effective regression testing through adaptive test prioritization strategies. Proc. J. Syst. Softw. 115, 61–81 (2016)

[14] Ansari, A., Khan, A., Khan, A., Mukadam, K.: Optimized regression test using test case prioritization. Proc. Comput. Sci. 79,152–160 (2016)

[15] Hettiarachchi, C., Do, H., Choi, B.: Risk-based test case prioritization using a fuzzy expert system. Proc. Inf. Softw. Technol. 69,1–15 (2016)

[16] Qu, X., Cohen, M.B., Woolf, K.M.: Combinatorial interaction regression testing: a study of test case generation and prioritization.In: The proceeding of IEEE International Conference on In Software Maintenance, ICSM, pp. 255–264 (2007)

[17] Do, H., Mirarab, S., Tahvildari, L., Rothermel, G.: The effects of time constraints on test case prioritization: a series of controlled experiments. Proc. IEEE Trans. Softw. Eng. 36(5), 593–617 (2010)