Journal of Engineering and Applied Sciences 15 (5): 1067-1073, 2020
ISSN: 1816-949X

Towards the Prioritization of Test Case by using NDBSC-FFNN

N. Gokilavani and B. Bharathi
Sathyabama University, Chennai, India

Abstract: The technique of ordering the test cases in the test suite in order to enhance the effectiveness of testing process is known as the test case prioritization. According to some criterion the test cases are accomplished in an order that are having the maximum priority than the one with lower priority by using the prioritization technique. This prioritization technique will be employed to sort the test cases in an order. This in turn increases the rate of fault detection. The measure to detect the fault rate increases will improves the performance of the running system and the makes the software testing process an efficient one. There may be a problem of occurrence of faults due to some error in coding. Hence, it is necessary to detect the fault rate to make the system an efficient one. In this proposed work an input test case dataset are preprocessed followed by the generation of Adjacency Matrix (AM). The features are then extracted by the use of adjacency matrix. A novel Density Based Spatial Clustering of Application with Noise (DBSCAN) is presented from which the optimization of test case using the Feed Forward Neural Network (FFNN) is carried out to obtain the optimized results. Then the faults in the test case are predicted which is then prioritized according to the maximization of test case with the help of bubble sort algorithm. This in turn sorts the test case and swaps them as per the priority provided for the test case. The performance analysis was made for the proposed system (NDBSC-FFNN) and are compared with the existing methodologies. From, this analysis it is clear that the proposed method performs well than the existing methodologies.

Key words: Feed forward neural network, DBSCAN, adjacency matrix, preprocessing, optimized results, existing methodologies

INTRODUCTION

The software testing was an indispensable one for the entire software development. Likewise, manual testing is the expensive one and are labor intensive. Hence, the automation process plays a significant role in the reduction of cost and enhancing the software testing effectiveness. Nowadays, a wide variety of software testing tools have been developed and are provided in marketplace. In these, the test case generation is the most challenging and a demanding one which shows an extensive impact on the efficiency and effectiveness of the entire process of testing.

Due to the development of software field in recent years, the prioritization as per the requirements shows a vital role. Several factors may have an adverse effect on the decision to be made to give priority for the documentation to implement and to arrange them orderly. Several number of techniques for the (Khatibsyarbini *et al*., 2018) generation of test case are implemented and analyzed. There are several types of testing process according to which the tester can adopt the software (Dave and Agrawal, 2017) testing process as per the requirements. In test case prioritization, the arrangements of test cases are made to make the preferences for test case and to identify the faults.

On improving the detection of fault the system under feedback will be capable of providing quick response and the faults could be corrected effectively. This has been implemented to meet the user needs with the use of (Krishnamoorthi and Mary, 2009) quality of software's thereby enhancing the rate of fault detection.

Since, there were several techniques that have been implemented for the process of test case (Pathy and Baboo, 2016) prioritization there are some limitations and challenges to overcome the faults occurred due to some defects of coding. Thus, to overcome this issue test case prioritization should be implemented. The main intention of this proposed method are as follows:

- To identify the faults occurred due to defects of coding process
- Usually, the software testing is the time consuming factor and are expensive, implementing the tool with the minimum cost is an essential process
- To categorize the test case according to the priority that should be given initially
- To generate the appropriate order of test case
- To schedule the order of execution for the test cases cluster the applications with noise by using density based spatial technique
- To optimize the test case with the help of feed forward neural networks
- To detect the faults occurred in the test case
- To prioritize the test case by using bubble sort technique

Corresponding Author: N. Gokilavani, Sathyabama University, Chennai, India

Literature review: In recent days, the development of software system have been extensively increasing. This may cause some defects in the system during the process of coding. Thus, it was an essential factor for the engineers to accomplish (Arafeen and Do, 2013) testing regression in order to ensure the quality of the software systems. In this the requirements based clustering approach was analyzed which in turn incorporates the systematic analysis of code thereby enhancing the test case prioritization efficiency. From this it was evident that the use of this type would be a beneficial factor. There were some threats which could be overcome by the utilization of several code metrics and the various cluster size.

A system level value driven approach was developed for the purpose of test case prioritization (Srikanth *et al*., 2014) termed as prioritization of requirements for test. This in turn analyzed and assign the values depending on the volatility needs, implementation complexity, proneness of faults and the priority of customer. This enhances the detection of fault rate thereby decreasing the several types of failures. However, there were some limitations in this method which has to be faced.

A new technique was presented to prioritize the test (Sharma *et al*., 2014) case on discovering the difficult clustering paths that were critical with the utilization of genetic algorithm. Once, the requirement of software have been change then there was a need to alter and retesting of the software system. The limitation of this work was that there is a need to develop an approach for the application into UML diagram such as sequence diagram which also utilizes the white box and the object oriented testing.

The field of software testing was having the several number of problems such as effort, cost and time of the testing. Various methodologies and techniques were proposed to solve (Sharma *et al*., 2013) this difficulties. For many investigators, the utilization of evolutionary algorithms to automatic test generation was the area of interest. The Genetic algorithm was the one such evolutionary algorithm kinds. Furthermore, the genetic algorithm was enhanced for the regressive testing and this GA has to be utilized with the other kind of software methodologies like neural networks and fuzzy logic for the generation of test case from the UML diagram.

The test case prioritization was performed to arrange the test case, thus, to improve the testing effectiveness. The coverage of code has been utilized as a factor in the test case prioritization. The ordered (Fang *et al*., 2014) sequence of entity programs have been utilized to enhance the test case prioritization effectiveness. Various test case prioritization methods based on similarity have been presented to enhance the detection rate of faults present in the loops. Further analysis have been made to check the number of mutation faults and the number of test cases which affects the test case prioritization. The testing may be utilized to support the risk analysis and the risk analysis have been used to aid testing process. The identification by Erdogan *et al*. (2014) sufficient approach have been utilized. Anyhow, there were some shortcomings of this work that should be overcome by some modifications.

For the automatic generation of test cases an orchestrated survey of the most (Anand *et al*., 2013) important methods have been presented which includes the symbolic execution of structural testing. Combinatorial testing, search dependent methods and a model dependent testing. There was a need to reformulate the test objectives, thus, offering the avenues.

A novel technique was proposed to increment knowledge acquisition and also (Tonella *et al*., 2013) to overcome the different constraints. In this approach the requirements prioritization was based on the hierarchical collection of requirements which put up numerous kinds of priorities and restraints. The choices should be made by the comparative prominence of requirements or else there was a need that a specified implementation order must be an option to a human having the complicated knowledge during requirements prioritization. The effectiveness, efficiency and robustness was improved and there was some limitations in this work for investigating conceivable stopping rules for IGA, for instance, stopping rule ides must be accepted that well-defined in IAHP.

An innovative method for software testing was one of the crucial task in software development. Various existing approaches were (Srivatsava *et al*., 2013) reviewed but the self-governing test path could not be made in that. Top overcome these issues metaheuristic firefly algorithm was realized to optimize the test paths through suitable objective functions. Additionally in this method for traversing the graph, guidance matrix was introduced. Thus, the test paths generated here was optimal path. Further analysis have been made to fine tune the algorithm for the wider range of applications.

The test analysis algorithm was performed to estimate the each test case functionality (Thomas *et al*., 2014) to offer high priority for testing different functionalities. The proposed technique compared with existing one to analyze the performance of the techniques. The static black box TCP technique was performed better comparatively than other techniques. Anyhow there were some limitations like distance maximization algorithms which was not considered in this method.

The new proposed algorithm of proportion-oriented randomized for the test case prioritized it have the viable solution still the existing based algorithm in less effective (Jiang *et al*., 2015) suggested to shown the test case

prioritization methods effectiveness. The PORA optimized the distance between the prioritized the hierarchy of the distributed data as input. Therefore they compared the additional greedy algorithm and other techniques of ART methods if not more effective to compare those algorithms. However, the additional system utilized the ART procedures to cover the information usage of code. Furthermore these methods were constant in efficiency.

In regression testing some faults were detected and the test cases method increased in the test case prioritization (Hao *et al*., 2014) offered the unified test case prioritization it included the two models as extended and basic also it ranged with the purely additional and purely total methods. They evaluated the approach and it performed the impact of external three factors and internal three factors. So, they finalized the techniques of unified teat prioritization applied in the Java have unit tests and the C have the system test.

They maximized the fault detection of the test localization prioritization methods. Before they step the one fault to detect and the other fault were fixed (Yoo *et al*., 2013) presented that the new techniques of fault localization prioritization difficult which combined the localization and prioritization. Also the three factors evaluated the techniques as Fault Localization using Information Technology (FLINT) and Test Case Prioritization (TCP) are previously standard work. From the techniques it improved the efficacy from the fault localization also tried to reduce the entropy of locality faults.

The modified test complement is enhanced by means of Cuckoo Search (CS) beside combinatorial method and alteration testing is utilized to seed diverse (Ahmed, 2015) responsibilities to the software-under-test in addition to filter the test cases established on the sensed faults. To extent the efficiency of the method an experiential revision is led on a software organism. The method shows its efficiency over the accompanied case study. However, there were some limitations of this method that could be faced.

The foremost intention of the (Petrus *et al*., 2013) software testing was to identify the failures of software that might be corrected or discovered. It utilizes the extended state of final machine. This in turn enhances the rate of fault detection. The optimization process was carried out in order to reduce the cost. Some testing includes an amount of test cases together identified as test suite. When test suites turn into too large, they can be hard to accomplish and exclusive to route. They must need more calculation assets and implementation time.

The software development and maintenance activities have complained the most thing of the regression testing. Here the test case prioritization were used for the profitable case of the appropriate test cases be existed. (Gupta and Yadav, 2013) presented the algorithm test case to prioritize cases based from fault impact and fault detection. Then they determined the efficacy of new tests have the arrangements of APFD (Average Percentage of Faults Detected) were used. So, the software suggested the improvement of the detection and effective as the version of software.

Solanki *et al*. (2016) evaluation of an innovative "m-ACO" ("Modified Ant Colony Optimization") technique was handled for regression test. By varying the food source election criteria, the technique of "m-ACO" can prioritize the test to enrich the fault diversity. "m-ACO" technique is comparatively evaluated with GA, BCO and ACO for prioritization of test cases. Effectiveness of the scheme was measured by utilized the subsequent measures:

APFD ("Average Percentage of Faults Detected"). PTR ("Percentage of Test suite Required for complete fault coverage"). Efficiency of the scheme was proved and the evaluation can be handled in future by utilizing the open source software. From this analysis it is clear that the existing methodologies are having some limitations that can be overcome by the proposed methodologies.

MATERIALS AND METHODS

Proposed work: This section describes the proposed methodology (NDBSC-FFNN) of the test case prioritization. Figure 1 represents the flow of proposed system in which the input test case is preprocessed and then the analysis of clustering is performed by the generation of adjacency matrix. The features are then extracted from the AM. A novel density based clustering process is carried out followed by the optimization process with the feed forward neural networks for the identification of faults then prioritize the test case by using bubble sort algorithm. The flow of the proposed system is constructed by the following parameters:

- Pre-processing
- Generation of adjacency matrix
- Density based clustering of application with noise
- Optimization of the test case by feed forward neural networks
- Detection of faults
- Prioritization of test case using bubble sort algorithm

Pre-processing: The input test case data set are preprocessed in order to remove the noise present initially in the input dataset. The preprocessed input are then kept in the training database. The generation of test case data involves the future use of considering the testing like the description, trial and loop. This is known as the pairing

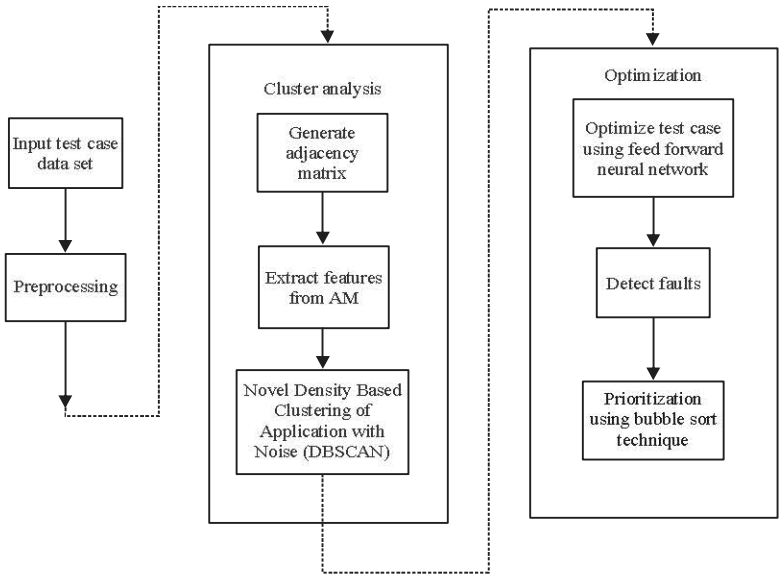

Fig. 1: Flow of the proposed system

process. This in turn helps the search presentation to increase. This type of test case shows an effective result.

Adjacency matrix generation: After, the process of pre-processing and pairing values are selected, the adjacency matrix will be generated. By anticipating the implementation incidence of loops or codes initially the value of zero in the matrix will be increased to 1 at that matrix. The size of that matrix will be dependent on the path that are generated. Generally, the major intention of this generation of adjacency matrix is to uncover the faults that are in the test case. If the whole bugs are concealed then the matrix will stops the checking process. Otherwise, the test cases that are adjacent will be checked for errors and the whole bugs are covered. The entire value of errors that are enclosed for each test case will be estimated including the adjacent matrix. Thus this process encloses the entire errors and faults present in the test case.

Feature extraction: Once, the generation of adjacency matrix is performed, the features are extracted. The features are extracted from the adjacency matrix that are formed previously. The features that were extracted will undergoes clustering process.

Clustering: A novel density based spatial clustering of application with the noise is proposed for the process of clustering. The clustering algorithm was used in the metric thus to divide the clusters that are leaving along with the un-clustered buckets test. This might be carried out by the density-based spatial clustering of applications with the noise. This method is capable of detecting the cluster numbers depending on the distribution density of the offered data points.

It is usually utilized to cluster the pixels in a different form to a group of clusters. It does not needs the number of clusters that are pre-defined on the other hand it stores the map of cluster with the cluster ID as a key and setting the value of various pixels regarding that clusters.

The clustering process will be carried out after the extraction of features from the adjacency matrix. The novel density based spatial clustering algorithm is represented below. In this the extracted features from the dataset is provided as input by setting the value of cluster initially as zero. Then the neighborhood function is set followed by epsilon and the point from the dense region. Then the values of j and k will be provided as 1. The distance and the size was computed followed by the clustering process. Thus, finally, the output will be a clustered output.

Algorithm 1; Novel Density Based Spatial Clustering of Applications with Noise (DBSCAN):

Input: Extracted features from dataset Ext_{ds}
Output: Clusters Cl_{cl}
Procedure:
Set Initial Cluster $Cl_{cl} = 0$
Set Neighbors as Nei_{nr}
Set Epsilon as eps_{ep}
Set points to form dense region pt_{min}
For j =1 to Ext_{ds} (size)
if distance (Ext_{ds}, j)≤eps_{ep}{
$Nei_{nr} = Nei_{nr}+(Ext_{ds})$ j
if $Nei_{nr}<pt_{min}$
label (j) = Noise`
$clu_{lb} = clu_{lb}+1$
$Seed_{ss} = Nei_{nr}$\j
For k = 1 to $Seed_{ss}$ (size)
If label (j) = Noise
label (k) = clu_{lb}
If $Nei_{nr}>pt_{min}$
$Seed_{ss} = Seed_{ss}+Nei_{nr}$

Optimization: The test case is optimized by using the feed forward neural networks. The clustered output undergoes the optimization process. The feed forward neural network plays an important role in the classification scenario. The process of supervised learning having the input data needs the objective function and its usage thus to access the output values that are predicted. As it is the fastest algorithm and are convergence. The adjustment can be done during the phase of training in the neural network. If the error is occurred then the test data is used for the testing phase. This in turn causes the minimization of output error.

The clustered output is optimized by using the feed forward neural network which can be represented as follows. In this the initialization of network is performed and the hidden, output layers are taken as $netw_{hid}$ and $netw_{out}$. Then the three functions are taken as activate, forward propagate and predict. The activation process is taken place followed by the forward propagation of neuron layers. Finally, the predict function predicts the results that are having the maximum optimization.

Algorithm 2; Optimized feed forward neural network:

Initialize network
$netw_{ini}$ be the initialized network
$netw_{hid}$ be the hidden layer
$netw_{out}$ be the output layer
$netw_{hid}$←for i =1 to len(TW_{tws})+1
for i = 1 to $netw_{hid}$
$netw_{ini}$←$netw_{hid}$
End For
End For
$netw_{out}$← for i = 1 to len($netw_{hid}$)+1
for i = 1 to $netw_{out}$
$netw_{ini}$←$netw_{hid}$
End For
End For
Activate
Let neu_{wei} be weights of neuron
Let neu_{net} be activation of network
for i = 1 to len (neu_{wei})
neu_{net}+ = neu_{wei}*$netw_{ini}$
End for
Forward propagate
neu_{inp} be neuron inputs
neu_{out} be neuron outputs
for layer in neu_{net}
for neuron in layer:
neu_{net} = activate (neuron (neu_{wei},))
neu_{inp}←neu_{out}
End For
Predict
$feat_{pre}$ be predicted results
$feat_{pre}$ = Forward propagate(network, row)
$feat_{pre}$ = max ($feat_{pre}$)

Thus, the optimization results were predicted by using this feed forward neural network algorithm. From this the results of fault will be detected.

Prioritization using bubble sort algorithm: The test case are checked for faults, after the detection of faults the test case undergoes some sorting process in order to arrange the test cases according to priority. For this process of sorting the bubble sort algorithm is used. It continuously works repeatedly by stepping through the list that are to be sorted. The comparison of each pair in the adjacent items are done followed by the process of swapping if they are in an incorrect order.

The bubble sort process is capable of utilizing the comparison of pair-wise function. In this sorting algorithm the decision maker is capable of deciding the needs among which one is important the need of lower one is most important than the one in the top and the swapping of their position is performed. Thus, according to this the priority is given for the one having higher probability and the other will moves to swapping process and this may continue till the end of process. By this the prioritization to the test case is given.

RESULTS AND DISCUSSION

Performance analysis: This section provides the performance estimation of the proposed system (NDBSC-FFNN) and the comparative analysis of proposed system with the existing methodologies.

Test case and the fault detection rate: The average percentage of the fault that are identified is generally constructed for evaluating the test case arrangement functions. Let D be the group of test that are enclosing the test cases M, B is represented as faults or bugs. This can be depicted as follows in a mathematical way:

$$APFD = 1-\frac{DB1+DB2+..., DBp}{mp}+\left(\frac{1}{2m}\right) \quad (1)$$

Table 1 depicts the comparison of test case for the both proposed method and the existing methods. By this analysis it is evident that the performance analysis of proposed method (NDBSC-FFNN) yields a better efficiency on comparing the other existing methodologies. Figure 2 depicts the performance analysis of the test case of prioritization and a non-prioritization for the both existing and proposed methods. From this analysis it is shown that the proposed method (NDBSC-FFNN) performs well in both prioritization and non-prioritization case.

Table 2 provides the fault detection rate of the existing and proposed system for various test cases. The

Table 1: Test case prioritization

Techniques	Prioritized	Non prioritized
Existing	5	8
Proposed	7	9

Table 2: Fault detection rate

Test case	T1	T2	T3	T4	T5
Existing	30	10	20	25	40
Proposed	35	20	25	35	50

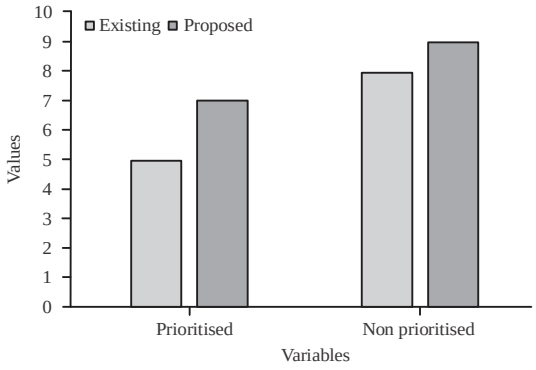

Fig. 2: Performance analysis of test case

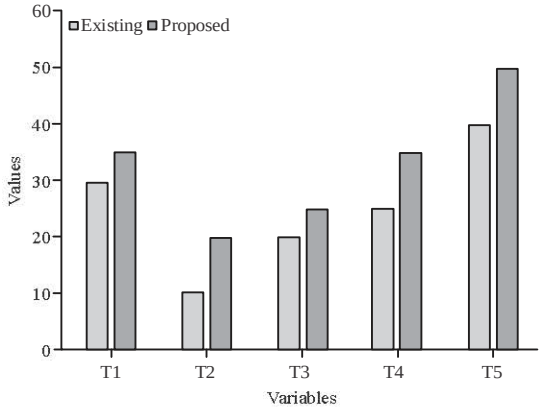

Fig. 3: Performance analysis of fault detection rate

proposed method (NDBSC-FFNN) shows a better result than the existing methodology. Figure 3 depicts the performance analysis of the fault detection rate for the both existing and proposed methods at various test cases. From this analysis it is shown that the proposed method (NDBSC-FFNN) performs well in various test cases.

CONCLUSION

The test case prioritization technique was employed to order the test case as per the priority provided by them. In the proposed work, the preprocessing of the input dataset was done. Then the generation of adjacency matrix was made to cluster the output of adjacent matrix by using a Novel Density Based Spatial Clustering process (NDBSCAN).

The features were extracted by this adjacency matrix. Then the optimization of test case with the utilization of feed forward neural networks. Finally, the faults were predicted and then the priority was given to the test case using the sorting techniques like bubble sort algorithm. This in turn arranges the test case according to the priority and swaps the remaining test case which will be continued till the end of process. The performance analysis was made followed by the comparison of test cases and the fault detection rate which shows that the proposed method achieves better result than the other existing methods in case of detecting the faults rate.

REFERENCES

Ahmed, B.S., 2015. Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. Eng. Sci. Technol. Intl. J., 19: 737-753.

Anand, S., E.K. Burke, T.Y. Chen, J. Clark and M.B. Cohen *et al.*, 2013. An orchestrated survey of methodologies for automated software test case generation. J. Syst. Software, 86: 1978-2001.

Arafeen, M.J. and H. Do, 2013. Test case prioritization using requirements-based clustering. Proceedings of the 2013 IEEE 6th International Conference on Software Testing, Verification and Validation, March 18-22, 2018, IEEE, Luxembourg, Europe, pp: 312-321.

Dave, M. and R. Agrawal, 2017. Mutual information gain based test suite reduction. Int. J. Comput. Appl., Vol. 168, No. 4. 10.5120/ijca2017914358

Erdogan, G., Y. Li, R.K. Runde, F. Seehusen and K. Stolen, 2014. Approaches for the combined use of risk analysis and testing: A systematic literature review. Int. J. Software Tools Technol. Transfer, 16: 627-642.

Fang, C., Z. Chen, K. Wu and Z. Zhao, 2014. Similarity-based test case prioritization using ordered sequences of program entities. Software Quality J., 22: 335-361.

Gupta, R. and A.K. Yadav, 2013. Study of test case prioritization technique using APFD. Int. J. Comput. Technol., 10: 1475-1481.

Hao, D., L. Zhang, L. Zhang, G. Rothermel and H. Mei, 2014. A unified test case prioritization approach. ACM. Trans. Software Eng. Method. (TOSEM), Vol. 24, No. 10. 10.1145/2685614

Jiang, B., W.K. Chan and T.H. Tse, 2015. PORA: Proportion-oriented randomized algorithm for test case prioritization. Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security, August 3-5, 2015, IEEE, Vancouver, Canada, pp, 131-140.

Khatibsyarbini, M., M.A. Isa, D.N.A. Jawawi and R. Tumeng, 2018. Test case prioritization approaches in regression testing: A systematic literature review. Inf. Software Technol., 93: 74-93.

Krishnamoorthi, R. and S.A.S.A. Mary, 2009. Requirement based system test case prioritization of new and regression test cases. Int. J. Software Eng. Knowl. Eng., 19: 453-475.

Pathy, S. and S. Baboo, 2016. Analysis of code coverage metrics using eCobertura and EclEmma: A case study for sorting programs. Int. J. Adv. Res. Comput. Sci. Manage. Stud., 4: 121-130.

Petrus, C.N.A., M.S. Razou, M. Rajeev and M. Karthigesan, 2013. Model-based test case minimization and prioritization for improved early fault detection capability. Int. J. Innovative Technol. Exploring Eng. (IJITEE), 2: 205-210.

Sharma, C., S. Sabharwal and R. Sibal, 2013. A survey on software testing techniques using genetic algorithm. IJCSI. Int. J. Comput. Sci. Issues, 10: 381-393.

Sharma, C., S. Sabharwal and R. Sibal, 2014. Applying genetic algorithm for prioritization of test case scenarios derived from UML diagrams. Intl. J. Comput. Sci., 8: 433-444.

Solanki, K., Y. Singh and S. Dalal, 2016. A comparative evaluation of "m-ACO" technique for test suite prioritization. Indian J. Sci. Technol., Vol. 9, No. 30. 10.17485/ijst/2016/v9i30/86423

Srikanth, H., S. Banerjee, L. Williams and J. Osborne, 2014. Towards the prioritization of system test cases. Software Testing Verif. Reliab., 24: 320-337.

Srivatsava, P.R., B. Mallikarjun and X.S. Yang, 2013. Optimal test sequence generation using firefly algorithm. Swarm Evol. Comput., 8: 44-53.

Thomas, S.W., H. Hemmati, A.E. Hassan and D. Blostein, 2014. Static test case prioritization using topic models. Empirical Software Eng., 19: 182-212.

Tonella, P., A. Susi and F. Palma, 2013. Interactive requirements prioritization using a genetic algorithm. Inf. Software Technol., 55: 173-187.

Yoo, S., M. Harman and D. Clark, 2013. Fault localization prioritization: Comparing information-theoretic and coverage-based approaches. ACM. Trans. Software Eng. Method. (TOSEM), Vol. 22, No. 3. 10.1145/2491509.2491513