Proceedings of the Fourth International Conference on Trends in Electronics and Informatics (ICOEI 2020)
IEEE Xplore Part Number: CFP20J32-ART; ISBN: 978-1-7281-5518-0

An Enhanced Adaptive Random Sequence (EARS) Based Test Case Prioritization Using K-Mediods Based Fuzzy Clustering

N.Gokilavani[1]
Research Scholar, Dept. Of computer science and engineering, Sathyabama institute of science and technology, Chennai, India
gokilavani_testing@yahoo.com

Dr.B.Bharathi[2]
Professor, Dept. Of computer science and engineering, Sathyabama institute of science and technology, Chennai, India
bharathi.cse@sathyabama.ac.in

***Abstract*—The efforts of prioritization method is to maximize the detection of fault rate by organizing the significant test cases which is operated in a sequence of regression tests. Generally, it is implemented to sort down the test cases based on the priorities former than those with minimum priority imparting to an estimated criteria. The faults which gives maximum impacts should be detected at earlier stages in testing practices. The adaptive random testing is implemented to execute arbitrary testing through input triggering clustering errors. It improves the detection ratio of regression testing in software based on object-oriented. In this proposal, an adaptive techniques of test case prioritization relied on fuzzy clustering is implemented. The adjacent matrices is generated and cluster head is chosen within the test cases. It is made by identity precise pairing. Then Enhanced Adaptive random sequence depending on prioritization of test cases detects the flaws which operates to categorize neighboring test cases as varied as possible. Hence the outcomes proved increased efficacy in earlier fault detection rate.**

Keywords—Adjacency Matrix; Fuzzy k-medoid; Adaptive Random Sequence (ARS); Average Percentage of Fault Detected.

I. INTRODUCTION

The growth in software development requires a sustainable regression to manage the high quality in the current competitive circumstances for testing operation. Nowadays, the prioritization in the test cases is the modern progressive method to compensate for the enhancement requirement of software testing efficacy. [1]This method is better for earlier fault identification, and so the debugging can be done at more initial stages. The present testing method decides implementation patterns of all test sets depending on the attention of either loop or coding on behalf of project and product capacity to eliminate the faults. It doesn't aim at the execution of data for prioritization of test cases.

A. Predicted issues

Even though there are various methods suggested for test case prioritization, but still, there are few difficulties when it is applied in a real-time environment. The imperfection in the software is occurred due to the sparse coding, which should be identified and rectified. The significance of the identical test cases should be recognized that denotes the functional dependencies. The active test case prioritization should enclose multiple criteria such as fitness values, fault data, code coverage, and historical execution of data.

B. Scope of the research

In the software development lifecycle, software testing is a mandatory process which should be performed effectively. A focused approach should be insisted to determine the precipitate optimum glitches by considering the test cases in a detected way. This method of generating test case is referred to test case prioritization. In this study, the adaptive clustering approach is developed that executes based on the statement level fuzzy optimization methods. The main objective of this research is to identify the bugs as the principal element as conceivable. The test case prioritization recognizes the similarities of evolution in regression testing by applying a fuzzy-based clustering method. Generally, software testing is a time-consuming task and expensive to be used. So, it is necessary to design the solution in optimal cost for declining the testing methods. It should be calculated how the priority wise test cases varies from the non-prioritize test methods. Because the test case of non-prioritize documents will run by test procedure and the further step would be the testing phase. In sequential test cases, the drawbacks of testing circumstances is fed as input for any of the prioritization algorithm. Conferring to the mentioned priorities, the testing case will be accomplished.

[2]The adaptive approach gives the required data concerning the current execution of a application at execution time. It computes the efficiency of the test cases, which collects the data flaws and organize the priority of every situation in subsiding order. [3]The adaptive approach selects the test case and performs once the output of the process is achieved when the priority of the test cases is insisted. One by another produces the real arrangement of the test case. After execution, the fresh order is created based on priority. The original test case order focused on the priority test case order, which is not matched. Since it has less faulty test cases which are marked as the first number of the sequences which is not possible by original test cases since it's prone to more attacks. There are several metrics which are evaluated with the aid of prioritization techniques (test case). [3]It is implemented for programming the order of test cases. The adaptive algorithm is processed to give the prioritization. In this research, the Enhanced Adaptive Random Sequence

(EARS) is implemented to organize the test cases. The programming of test cases is very noteworthy for completing the increased amount for test case. The main purpose of this research is to detect defective modules. In this proposed system, finding fault in test cases will be computed at the selection of test cases. Initially, chose a minimum test for implementation. When the first test case is completed, the performance will be stored as its output. The selection process is persistent until test cases are completed. Few statements are not selected because they will not get executed. The adaptive method is best strategies for test case prioritization. The novelty in this research is the enhanced adaptive algorithm is as follows,

- The adjacent matrix is produced and listed relied on the flaws of test cases. So the method selects the candidate who owns the increased faults as cluster head.
- The fuzzy clustering technique, along with integrated k-medoids, is implemented for creation of clusters with identical case attributes. It is because of the cases in the same group and is diverged in commencing of preceding auxiliary clusters.
- The enhanced adaptive random sequence is abbreviated as EARS used to enhance the failure identification by continually spreading the test input overall finishing the input province. The proposed algorithm adds up in the choosing of the ordered test case as divergent along the input region as possible.

This research paper is organized into several sections as below. Part II explains the representation of a related survey for clustering methods and test case prioritization methods. Section III elaborates the description of the proposed method in selection of optimal path and cluster head and a way for Test case prioritization. Section IV explains the results, discussion and comparison with other methods. Section V concludes the research with analysis on the future extension for the proposed method

II. REVIEW ON RELATED WORKSFGDGD

The different techniques implemented to rectify the shortcomings of test case prioritization is discussed in this section. The positive and negative aspect of various fuzzy clustering methods based on the test cases is deliberated to overcome its limitations.

A.Test case prioritization

This part reviews the existing methods developed in the past decade on test case prioritization methods. This method is very imperious in regression testing. Generally, it is used to create the order of the conforming test cases. [4]The existing test cases with high priorities are completed prior than the case with lesser priority. Comparing the optimal prioritization methods with nine sections such as no prioritization, optimal, random prioritization, branch-addtl, branch-total, FEP-addtl, FEP-total, stmt-addtl and stmt-total. The enhancement in every process concluded that improvised rate of fault detection. But limitations of research executes only with the base application rather than the modified version of an application. [5]As proposed by, the software license works on with five standard prioritization to utilize the chances of main problem recognition ratio of SPL test cases. The similarities and variations are used for computation purposes. The maximum value of the product variant is accounted as every product's minimum variation at a new system that needs an end to end retesting. [4]Revised all test case prioritization methods and factors based on client requirements, coverage, history and cost. [6]As discussed by, when there are any alterations in the client requirements, the impacts of fault are defined. Test cases are well organized according to the ethics of fault proneness and customer priorities. This work covers three optimal techniques with NSGA-II to extend both diversity and coverage for minimizing the test accomplishment duration. It is compared with coverage and justified the efficacy of the merged techniques. The disadvantages of this method eliminate the cost of unidentified faults. [7]Proposed a section of the program to execute test case prioritization as more effective. It is cost-effective, and the bugs are identified at an earlier stage. But it is not effective as it consumes more time for execution. Discovered the new method where fault detection is based on direct dependency and indirect dependency. But the size of the program is not portable, which influences the execution of test case prioritization. [8]Implemented three similarity features such as Hamming distance, edit distance and basic counting to increase the efficacy of the testing process. [9]Suggested a distinct approach consider the unselected test cases based on the recent nomination of the software's original version. The factors such as value, fp, coverage programming, and litigation and coverage granularities are improved in the prioritization methods. It gives that variable have required origin in task and test cases which should be tagged for every variable to give different clarifications and accessible attributes from variables essential for research. [10]Explained about the common causes encountered in the problem retrieval, that can localize the chaos of regression test prioritization such as the variance of documents organized between two test versions. [11]To improve the capability of earlier fault detection, proposed DIP to use the management of data and control dependence variables by sequences of collaboration but the length of the code gives complexity which should be resolved further

B. Cluster based prioritization of test case

This review deliberates the clustering-based test case prioritization methods. [12]Explained the similarity of cosine to cluster the similar values for cases (test) . The priority values are calculated by feeding the clusters as input. It is made simultaneously to limit the computation time and maximize error recognition. Proposed three different variables such as information of fault history, complexity measure and coverage of code is implemented to the clusters to build the prioritization model. [13]Deliberated to implement priorities to work on descending order with six factors of the test cases which primarily doesn't focus on clustering, but it operates on with cluster data.

C. Test case optimization techniques based on soft computing

[14]Explained a fuzzy approach to allocate the priority by various attributes that include coverage based parametric values, variety of event and collaboration. The priority of test cases are organized depending on fitness functions, and the pattern of the test suite is accomplished. The fuzzy method gives improved result other than existing methods. [15]Implemented a fuzzy rationality to control the prioritization of cases. It computed the defilement of a specified application assertion. The prior data on violation history is accounted for assessment when it is applied for the event formation. Suggested a prioritization method which is based on completed similar proceedings. The events are organized to the significance of their related structures, but it is not as efficient as the following bug location affects the performance of the further process. [16]It has implemented a fuzzy control approach for identifying the priority of cases through execution trees which are utilized in common languages to define the priorities. This method prevents the effortless results in the test by the symbolic execution of trees to manage where the test gives the most advantages to execution coverage.

The reviews, as mentioned earlier, stressed the need for the effective algorithm as the existing tools don't give remarkable efficacy in optimization and prioritization. Because some approaches are code-based or component-based, which is focused test case prioritization, it is cost-effective. So the combined methods of approximate analogous cases of clustering approaches and efficient prioritization algorithm for identification of faults as early as possible. The comprehensive methods lead in faster execution of the program.

III. PROPOSED SYSTEM

The detailed view of the proposed methodology by FCAR technique is integrated with k-medoids based on fuzzy clustering along with EARS relied on test case prioritization methods. It offers a great solution for test case prioritization methods. The overview of proposed cluster creation and discourse the merits and prioritization. The building of succeeding process to attain the test case prioritization to define the faults as early as possible is as structured below,

- Preprocessing
- Finding the literal pair
- Executing an adjacency matrix
- Selection of cluster head
- Computation of similarities
- Formation of clusters
- Enhanced adaptive random sequence prioritization

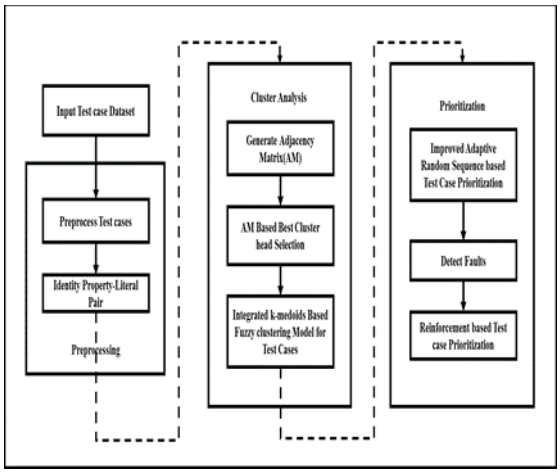

Fig 1. Overall flow of the proposed system

The fig.1 illustrates the workflow of the proposed framework of test case prioritization approach, which composed of chronological progressions. Predominantly, the test case dataset is fed as input to preprocessing block where the code and event dependences are explained in detail. Then a similar pair of test cases are developed, which is helpful to define the adjacency matrix. Then the adjacency matrix is implemented to considering every neighboring test case, where the cluster head is chosen. The assimilated cluster set may not be accounted as accurate as is a club of similar test cases. So, integrated k-medoids based fuzzy clustering techniques is executed for the creation of clusters with identical attributes of test cases. Then the enhanced adaptive random sequence prioritization method is applied to identify the faults in test cases as ordered.

A. Preprocessing and Pairing

The preprocessing of every test cases in the testing dataset is made to remove the duplicates and noises. Then the corresponding output is stored and aggregated in the testing database. The testing database load functions perform the tester delivers the testing criteria and store it in the table. The selection of test cases differs for each new product. The identity properties in the generation of test data include trail, loop, or description which is coupled for further use in the testing process. The pairing methods help in parallel processing which supports to enhance the presentation of the searches. Looping events are identified at the initial stage for the identification of the test data to be produced. The evaluation is mandatory for the recognition of diversity in the research programs. It could be reasonably multi-layered and may probably include extreme perceptions on the view of the developer and initiator of test data to define earlier possibilities in various programming scenarios. According to pairing, the test case is accounted to be useful when a pair of definition and variable usage is maximum.

B. Execution of adjacency matrix and selection of cluster head

The equivalent paring values are logged for each cases that indulges the requirements of test properties such as branch

coverage or statement. Generally, a test case is executed and checked on its pairing values. It is chosen and located in the adjacent matrix for every iteration, and subsequent loop or report is met by a test data. The execution of circuits and codes as predicted, if the primary value is zero, is increased by one in the generated matrix. It persists for every test cases when test cases meet the resultant report. The capacity of the model is based on generated paths. The matrix is built with several coverage and branches. The main focus of the adjacent pattern is to expose all the faults. The matrix halts when ensuring of total flaws are masked through it or not. If not, then chose the adjacent test cases that checks the remaining flaws and repetition on this whereas entire bugs have been encircled. When the whole bugs are surrounded, calculate an amount as the whole of errors enclosed using every test case placed in the adjacent matrix. This approach would guide a number of combining of test events known as coding structures that ensure that all software errors are handled. Then the test case that covers the highest faults is chosen as the cluster header.

C. Formation Of Cluster

It implemented the integrated k-medoids based on fuzzy clustering methods along with failure creating inputs. The clustering is the main progression which paved to favorable state of affair for ARS. [17]The adjacent dimensional matrix which is N*N is fed as input to a k-medoids algorithm that assigns every test case to one of C clusters to minimize the sum squares of the cluster.

$$\sum_{i=1}^{c} \sum_{k \in A_i} \| x_k - v_i \| 2 \qquad (1)$$

Here, the centre of the cluster is indicated as adjacent test cases to means of test cases in a single group, $V = \{V_i \in X | 1 \leq i \leq c\}$. A_i and v_i is the set of test cases and mean in the cluster i^{th}.

The test space in the matrix is discrete as each test case denotes the position of the adjacent matrix. The fuzzy K-medoids technique minimizes

$$J_m(V, X) = \sum_{i=1}^{n} \sum_{i=1}^{c} U_{ij}^{m} \, r(X_j, V_i) \qquad (2)$$

The minimization is attained with complete V in X, U_{ij}characterizes the fuzzy connection of X_j in cluster i. The relationship U_{ij} can be demarcated intended to increase the possibility of resolving the testing difficulties in various ways. The fuzzy membership design is represented as.

Where $m \in [1, \infty)$ is the fuzzifier.

$$U_{ij} = \frac{\left(\frac{1}{r(X_j, V_i)}\right)^{1/(m-1)}}{\sum_{k=1}^{c} \left(\frac{1}{r(X_j, V_i)}\right)^{1/(m-1)}} \qquad (3)$$

Algorithm 1 Fuzzy K − Medoids Algorithm

Assign the count of cluster c; Set i_ter = 0;
Randomly select the initial set of medoids:
$V = \{v_1, v_2, \ldots . v_n\}$ *from* X_c;
Repeat
for $i = 1$ *to c do* /* *Estimate memberships* */
for $j = 1$ *to n do*
Compute u_{ij} *by applying* (3), (4), (5) *or* (6)
end for
end for
Record the present medoids: $V^{old} = V$;
Calculate the new medoids:
for $i = 1$ *to c do*

$$q = argmin \sum_{j=1}^{n} U_{ij}^{m} \, r(X_k, X_j) \qquad 1 \leq k \leq n$$

$v_i = X_q$;
end for
$i\_ter = iter + 1$;
Until ($V^{old} = V$ *or* $i\_ter = MAX\_ITER$)

D. Algorithm for prioritization

EARS is being used to create ARS to organize a group of well-arranged test cases as a variant of the input province as conceivable. The primary objective of the proposed technique is to define the marginalization sections closer to each test case performed.

The arbitrary test cases that belong under the marginalization parts is not preferred as test cases. It acquires new test cases which are away from the operated test cases. The arbitral values in the selected cases inside the cluster are initially checked. If it meets the requirements for the test cases, then shift to the next matters. Hence the testing time is limited to the considered level and reduce the computation overhead by increasing the variability of test cases.

Algorithm 2 Improved Adaptive Random Sequence(IARS)

Set $m = 0$ *and* $S = \{\}$
Unsystamatically choose a test case, tc, *from the input suites based on uniform distribution.*
Increment n by 1. *If tc expose bugs, go to final Step;*
Else, record tc in S.
arbitrary form D
$= \{d_1, d_2 \ldots d_k\}$ *according to the uniform*

distribution

For each element in D, find its nearest neighbour in S.

In D, find which element, d_j, has the farthest distance to its nearest neighbor in S.

Let tc $= d_j$ and goto above increment step

Return m and t. EXIT.

IV. RESULTS AND DISCUSSION

The result of the existing methods and proposed techniques on prioritization is compared by evaluating the performance metrics such as fault detection rate and the average per cent of fault detected.

A. The Average Percentage Of Fault Detected (APFD)

It is used to predict the test case ordering rehearsals. If C is a test category closing in test cases, k then B will be a set of bugs p discovered by C. The initial test case index is CBi which organizes the C ' of cluster C and shows the problem i.

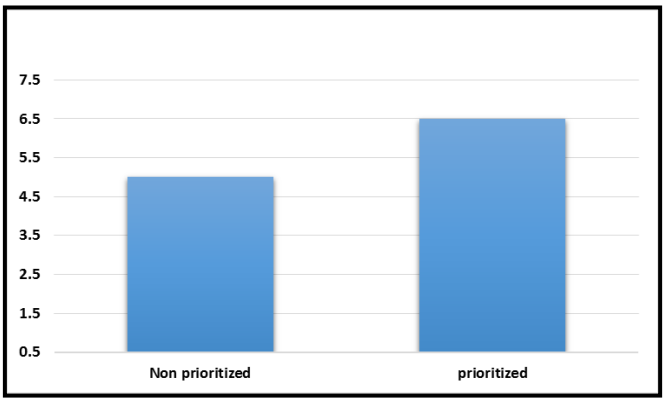

Fig 2. Prioritized VS Non Prioritized Test Cluster

The result of AFPD is associated for both prioritized as well as non-prioritized test cases. The result insists that prioritized cases deals earlier faults than one with non-prioritized cases.

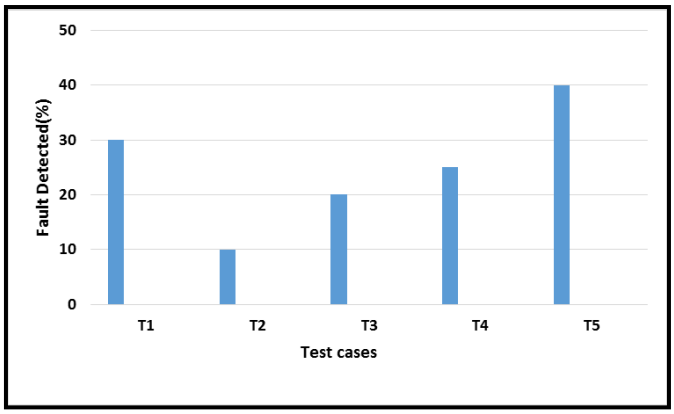

Fig 3. APFD of each test cases

The percentages for detection of faults is based on the evolution of given test order such as T4, T2, T1, T5 and T3 which gives the value of 0.65 whereas the prior arrangement shows 0.42

V. CONCLUSION AND FUTURE WORK

The proposed method is based on an improved cluster based fuzzy algorithm, which deliberates the understanding of medoids. The difficulty of this technique is O (n2) for every iteration at the time of imparting. The complexity narrates the positivity with the future fuzzy algorithm. The pairing idea offers coverage depends on the best path by the construction of the particular adjacent matrix for every iteration. The kind of test cases which demonstrates many faults is chosen as cluster head. The integrated k-medoids depend on on fuzzy clustering gives the preeminent test case when a similar error occurs. Then EARS algorithm detects the priorities to make organized and listed sequences. Hence the first result proved effective performance in software testing. In future, the proposed method is extended in high dimensional input to achieve increased performance metrics.

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g.” Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

[1] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," *IEEE transactions on software engineering,* vol. 28, pp. 159-182, 2002.

[2] J. Chen, L. Zhu, T. Y. Chen, D. Towey, F.-C. Kuo, R. Huang*, et al.*, "Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering," *Journal of Systems and Software,* vol. 135, pp. 107-125, 2018.

[3] B. Jiang, Z. Zhang, W. K. Chan, and T. Tse, "Adaptive random test case prioritization," in *2009 IEEE/ACM International Conference on Automated Software Engineering*, 2009, pp. 233-244.

[4] Z. Sultan, S. N. Bhatti, R. Abbas, and S. A. A. Shah, "Analytical review on test cases prioritization techniques: An empirical study," 2017.

[5] A. B. Sánchez, S. Segura, and A. Ruiz-Cortés, "A comparison of test case prioritization criteria for software product lines," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*, 2014, pp. 41-50.

[6] D. Mondal, H. Hemmati, and S. Durocher, "Exploring test suite diversification and code coverage in multi-objective test case selection," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, 2015, pp. 1-10.

[7] C. Fang, Z. Chen, K. Wu, and Z. Zhao, "Similarity-based test case prioritization using ordered sequences of program entities," *Software Quality Journal,* vol. 22, pp. 335-361, 2014.

[8] C. P. Rao and P. Govindarajulu, "Automatic Discovery of Dependency Structures for Test Case Prioritization," *IJCSNS,* vol. 15, p. 52, 2015.

[9] T. B. Noor and H. Hemmati, "A similarity-based approach for test case prioritization using historical failure data," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, 2015, pp. 58-68.

[10] D. Hao, L. Zhang, L. Zhang, G. Rothermel, and H. Mei, "A unified test case prioritization approach," *ACM Transactions on Software Engineering and Methodology (TOSEM),* vol. 24, pp. 1-31, 2014.

[11] C. N. A. Petrus, M. Razou, M. Rajeev, and M. Karthigesan, "Model-Based Test Case Minimization and Prioritization for Improved Early Fault Detection Capability," *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 2, pp. 205-210, 2013.

[12] R. Kanimozhi and J. Rbalakrishnan, "Cosine similarity based clustering for software testing using prioritization," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 16, pp. 75-80, 2014.

[13] M. H. DR, "Clustering approach to test case prioritization using code coverage metric," *International journal of engineering and computer science,* vol. 3, 2014.

[14] N. Chaudhary and O. Sangwan, "Assessment and Comparison of Fuzzy Based Test Suite Prioritization Method for GUI Based Software," *Assessment,* vol. 7, 2016.

[15] A. M. Alakeel, "Using fuzzy logic in test case prioritization for regression testing programs with assertions," *The Scientific World Journal,* vol. 2014, 2014.

[16] E. J. Rapos and J. Dingel, "Using Fuzzy Logic and Symbolic Execution to Prioritize UML-RT Test Cases," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, 2015, pp. 1-10.

[17] R. Krishnapuram, A. Joshi, and L. Yi, "A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering," in *FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No. 99CH36315)*, 1999, pp. 1281-1286.