International Journal of Applied Science and Engineering

Analysing a novel multi-objective prioritization model using improved fuzzy c mean clustering

Sarika Chaudhary[*], **Aman Jatain**

Department of Computer Science and Engineering, Amity University, Gurugram, India

ABSTRACT

Consistent regression testing (RT) is an abstract class, that considered indispensable for assuring the quality of software systems but it is too expensive. To minimize the computational cost of RT, test case prioritization (TCP) is the most adopted methodology in literature. The implementation of TCP process, performed using various hard clustering techniques but fuzzy clustering, one of the most sought clustering technique for selecting appropriate test cases had not been discover at a wider platform. Therefore, the proposed work discusses a novel density based fuzzy c- mean (NDB-FCM) algorithm with newly derived initialize membership function for prioritizing the test cases. It first, generates optimal number of cluster (Copt) using a density based algorithm, which in turn minimizes the search criteria to find the 'Copt', especially in cases where a given data set does not follow the empirical rule. Then, creates an initial fuzzy partition matrix based upon newly suggested initial membership method. In addition, a novel multi-objective prioritization model (NDS-FCMPM) proposed to achieve the performance goal of enhanced fault recognition. Initially, feature extraction carried out by exploiting the dependencies between test cases, and then test cases are clustered using proposed fuzzy clustering approach, which finally, prioritized using a newly developed prioritization algorithm. To validate the performance of suggested fuzzy clustering algorithm two-performance measure namely "Fuzzy Rand Index" and "Run Time" exercised and for prioritization algorithm "APFD" metrics is analysed. The proposed model is assessed using eclipse data extracted from Github Repository. Inferences generated depict that NDB-FCM clustering provide more stable results in terms of classification accuracy, run time and quick convergence when compared with other state-of-the-art techniques. Also, it is verified that NDS-FCMPM observes an improved rate of fault identification at early stage.

Keywords: APFD, Customer requirements, Fuzzy clustering, Feature extraction, Prioritization, Regression testing.

OPEN ACCESS

Received: March 19, 2021

Accepted: July 1, 2021

Corresponding Author:
Sarika Chaudhary
schaudhary@ggn.amity.edu

Copyright: The Author(s). This is an open access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0), which permits unrestricted distribution provided the original author and source are cited.

Publisher:
Chaoyang University of Technology
ISSN: 1727-2394 (Print)
ISSN: 1727-7841 (Online)

1. INTRODUCTION

Set of actions carried out on a software, once it is deliver for use, known as software maintenance. These activities (actions) are required to accommodate the changes that are usually vital during this phase of SDLC. In order to validate that these continuous changes are precise and pose no impact on the remaining functionality software systems, re-testing of product is necessary (Yoo et al., 2009) RT is a validation process of frequently performed activities to retest the modified versions of software, and thus owe to 50% cost of overall maintenance cost (Chaudhary 2018). Due to restraint resources and time, it is not advisable to attempt re-execution of all test cases. Furthermore, test cases play a significant role to automate testing (Mani and Prasanna, 2017). Significance of RT can be viewed by the fact that the only critical and expensive defect in past have been uncovered by it. Various activities of RT are selection, minimization and test cases prioritization (Yoo and Harman, 2010). First two, accounts for reducing the expense of testing process by selecting the relevant subset and by minimizing test suite to a subset, satisfying the prior coverage criteria respectively. Prioritization organize and rank test cases in a way that aims to improve code coverage efficiency and, thus deal efficiently with early detection of faults (Miranda et al., 2018). Besides, it provides faster feedbacks, thereby allowing developers to debug as early as possible. It also enhance the probability of execution of important tests, in cases where testing ends abruptly.

Since a variety of techniques for TCP are suggested (Pang et al., 2017; Hasan et al., 2017; Carlson et al., 2011) which demonstrate the usefulness of increasing fault detection rate. To a large extent, many of these techniques exploit statement coverage and hard clustering approaches for prioritization of test suites. Limited focus has been given to soft/fuzzy clustering methods in prioritization in past (Chaudhary and Jatain, 2020). Clustering is a predominant method to optimize TCP techniques as it minimize the count of pair wise comparison in the test cases (Garg et al., 2013). It can divided into two categories hard clustering and fuzzy clustering. In hard clustering a data element "x" belong either completely to a cluster or not at all. The value of data are crisp and can be either 0 or 1, it cannot lie in between values of 0 and 1. In contrast , in fuzzy clustering a data element "x" is assigned with a membership function "m", where "m" represents the degree to which "x" belong to different clusters at the same time and range of "x" lies from $0 < x < 1$ (Hüllermeier and Rifqi, 2009). In real time applications fuzzy clustering behaves more naturally than hard clustering because the object that lie near to boundary of clusters are not forced to belong to a single cluster specifically and more accurate classifications results can be drawn. The main intention of our proposed work is to implement a multi-objective prioritization model to prioritize test case under fuzzy clustering category. The suggested method developed to increase the rate of average percentage of fault identified in software systems. The planned approach has used code coverage and customer requirements factors for test case prioritization. The rest of the document structured as: Section 2 described the related work. Core contributions of the proposed methodology explained in section 3. Section 4 described the dataset. Section 5 presented result and discussions. Finally, section 6, explain the conclusions drawn along with the future scope of research.

2. RELATED WORK

The prioritization of test cases during RT process illustrated in research by different researchers. Rothermel et al. (2001) first studied test case prioritization based upon branch coverage. Test cases that cover maximum level of statements executed first. The work by Badwal and Raperia (2013) focused on code coverage and function calls based clustering. Conclusion shows that prioritized cases perform well in detecting faults than non-prioritized. Indumathi and Selvamani (2015) prioritized test case by deriving dependencies existing between the functions automatically. The results demonstrate that fault detection rate enhanced at early stage. Kaur and Ghai (2016) exploit functional dependency technique to enhance the performance of existing hill-climbing method.

Lichade and Thakur (2016) defined a novel density based K-mean clustering technique and test cases based on coverage information are prioritized using prim's algorithm. Mishra et al. (2019) considered mutant coverage to reduce the number of test cases, statement coverage and fault exposing potential to prioritize test cases using genetic algorithm. Rajarathinam and Natarajan (2013) introduced trace event based test case prioritization approach. Trace events used to find out the most relevant test cases in a project. Raju and Uma (2012) described agglomerative hierarchical clustering technique to prioritize test cases using customer and project requirements. Gokilavani and Bharathi (2019) defined an optimized DBSCAN algorithm, where feed forward neural networks were used to optimize test cases for better results. Predicted faults from the test cases prioritized with the help of bubble sort algorithm and it proved that proposed method (NDBC-FFNN) outperformed all other existing methodologies.

Hasan et al. (2017) introduced dissimilarity-based-clustering framework, which integrated historical failure information, coverage information and dissimilarity clustering to rank test cases. The framework evaluated on data set 'Defects4j' with APFD and compared with random, similarity and original ordering prioritization approaches. The results recorded an average 88.5% APFD and proved that the proposed framework outperforms the other prioritization methods. Recently, Yi et al. (2018) discussed a 'concrete-hyper-heuristic framework' to prioritize test cases. Praba and Mala (2011) suggested 'Critical Component Analyser framework' for real-time systems that used dependency of critical-modules for prioritization. In a recent work, Ju and Zhou (2016) proposed a framework suitable for android applications to prioritize test cases using their memory leak(ML) capability based on a prediction-model. Azizi and Do (2018) narrated-'Graphite (graph-based) framework', which concentrated on realizing two goals consecutively during prioritization.

Shrivathsan et al. (2019) discussed two fuzzy based clustering techniques based on similarity coefficient and dominancy test named FSTPM and DTCTP respectively. Real time data from SIR (software artefact infrastructure repository) has obtained and evaluated to measure the strength of the proposed methods and it has proved that test cases grouped because of similarity and dominancy requirements managed effectively in comparison to others. Badanahatti and Murthy (2017) proposed kernel fuzzy C mean clustering and grey wolf optimization algorithm is used for cloud based TCP. Gokilavani and Bharathi (2020) described EARS algorithm for TCP using k-mediods based fuzzy clustering. The results shows an improved ratio of RT in object oriented software's. After performing the extensive literature analysis on test case prioritization, some of the observed challenges in study are:

Prioritization techniques proposed majorly concentrate on code coverage information. Few methods utilized customer requirement, cost based and history-based techniques of test case prioritization too but still in real time application, they remain unexplored.

- Majority of frameworks have used single objective. Therefore, in order to optimize TCP additional objectives should utilized.
- Mostly clustering techniques adopted did not exploit the interdependencies between test cases and faulty function.
- Fuzzy C mean clustering proposed for test case prioritization works on predefined number of clusters, which in turns increases the search criteria to determine the peak count of clusters and increases the convergence time.
- Also, in FCM initial partition matrix is generated randomly, which do not assure the accuracy of the probability of association of an item to all clusters.

Featured issues addressed in the proposed research by introducing the following contributions:

- First issue addressed by considering the different customer requirement factors along with code coverage factors at the beginning, so that various faults that originate from requirements raised by customers can handled at an early stage.
- Single objective issue is resolved by proposing a new multi-objective model; it exploit dependencies and fuzzy clustering logic together to optimize TCP.
- The issue related to interdependencies is resolved by putting forward a novel dependency structure based fuzzy clustering based TCP model.
- Last two challenges are suppressed by introducing a novel density based FCM, in which it first calculate the optimal cluster count and then generate a more efficient initial partition matrix with the help of proposed initial membership calculation function.

3. PROPOSED FRAMEWORK

Practically, to assure software quality in the field of regression testing, TCP always plays a vital role. This research presents a novel dependency structure based density based fuzzy c- mean clustering- test case prioritization model (NDS-FCMPM) to enhance the cost-effectiveness of RT contrary to techniques proposed in literature. The suggested model (Fig. 1) define three integrated work stages: dependency detection, fuzzy clustering based prioritization and metric assessment.

3.1 Pre-Processing

The model takes as input test data generated from customer requirement and code coverage factors for a software. Then, the data from pool1 and pool2 pre-processed to make data concise and noise free. For this, first XML data converted into CSV format, so that we can clearly read our data. After that, each data separated by using the “report id”.

3.2 Dependency Structure Formation

High coupling between the sub modules of a software system results into more complexity. So, this step based on the assumption that by testing highly coupled sub modules first can improve the fault detection rate. Therefore, at first, the dependency structures among the faulty functions exercised and then exact number of dependent faulty functions derived from them.

3.3 Feature Extraction

Once the structure formation based on dependency between test cases done, the next step is feature extraction. It helps in describing the huge set of data with relevant accuracy even with reduced number of resources. This is required here, to convert string features into numeric data, making it suitable for clustering process. For example if we are having string ‘Null’, then it will be replaced by ‘0’. Likewise, for all the unique strings, we assign a separate value in incremental order and that string will replaced by the respective numeric value.

3.4 Proposed Density-Based Fuzzy C Mean (NDB-FCM)

This step results in cluster formation using the newly derived NDB-FCM algorithm. Dunn stated fuzzy c- mean clustering in 1974. Later, Bezdek further developed FCM by introducing the degree of membership (Bezdek et al., 1984) with each weight of data element. FCM works on dividing the given dataset X into n clusters. This algorithm set a random membership degree m to every item i in the data set X, which indicate the level of association of an item to every cluster. The higher the degree of membership, closer is the item to the centroid. Fuzzy C mean algorithm seeks to minimize the objective function, OB (ip, Z_i), defined in Equation (1), which is made up of membership function and distance between the data items.

$$\text{OB}(ip, Z_i) = \sum_{i=1}^{n}\sum_{k=1}^{j}(\boldsymbol{ip_{ij}})^{\boldsymbol{m}} \; ||\boldsymbol{Z_i} - \boldsymbol{F_c}||^2 \qquad (1)$$

The major drawback of fuzzy C mean clustering is its prerequisite i.e randomly selected centroids and defining the number of clusters in advance. Therefore, the proposed NDB-FCM with newly derived initialize membership function algorithm works on the principle of generation of optimal number of clusters at first using a density- based algorithm automatically. This step minimizes the search criteria to figure out prime cluster number and enhance convergence rate too. In addition, a novel method to assign the initial membership value suggested here, which increases the probability of correctness of assigned membership value to each data item in the beginning of the clustering process.

3.4.1 Proposed NDB-FCM

Step1: Generate number of clusters j using density-based algorithm according to Equation (2), and assign fuzziness index m and epsilon (m = 2, epsilon = 0.01).

$$ld_i = \sum_{z=1}^{n} e^{-d_{iz}^2 \ /dc^2} \tag{2}$$

Suggested approach is to sort the corresponding densities in descending order and assign the dc value according to density rate (where density rate Ɛ [0,1]). The maximum number of clusters can be decided based on cut-off density (dc).

Step 2: Calculate initial membership value and initialize the initial partition $(\boldsymbol{ip0_{ij}})$ membership matrix according to Equation (3).

$$\boldsymbol{ip0_{ij}} = \frac{\boldsymbol{ip_{ij}}}{\sum_{\boldsymbol{k=1}}^{\boldsymbol{j}}(\boldsymbol{ip_{ij}})} \tag{3}$$

Step 3: Compute the fuzzy cluster centroid Fc, using Equation (4)

$$\boldsymbol{Fc} = (\textstyle\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{ip_{ij}}\ )^{\boldsymbol{m}}\boldsymbol{Z_i})\ /\ (\textstyle\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{ip_{ij}}\ )^{\boldsymbol{m}}) \tag{4}$$

Step 4: Update the fuzzy membership function matrix according to Equation (5)

$$\boldsymbol{ip1_{ij}} = \frac{\boldsymbol{1}}{\sum_{\boldsymbol{k=1}}^{\boldsymbol{c}}\left(\frac{\boldsymbol{u}}{\boldsymbol{l}}\right)^{\left(\frac{\boldsymbol{2}}{\boldsymbol{m-1}}\right)}} \tag{5}$$

Step 5: Check convergence using Equation (6)

$$(\text{fin}) =(\boldsymbol{ip1_{ij}} - \boldsymbol{ip0_{ij}})^{\boldsymbol{2}} \tag{6}$$

If fin <= epsilon where epsilon is the termination threshold (0.01), then end; else go to step3 and repeat.

3.5 Prioritization Algorithm

This algorithm rank the clusters based on sum of severity level of each data in a cluster according to the Equation (7) and prioritize clusters according to the equation.

$$\sum_{\boldsymbol{\sigma=0}}^{\boldsymbol{N}} \boldsymbol{if S_\sigma} > \boldsymbol{1}\ ,\textbf{set}\ \mathbf{P} = \mathbf{1}, \boldsymbol{if}\ \boldsymbol{S_\sigma} < 1 \text{ then set } \mathrm{P} = 2,\ \boldsymbol{if}\ \boldsymbol{S_\sigma} = 0 \text{ then set } \mathrm{P} = 3 \tag{7}$$

4. DATASET

To evaluate the proposed model, complete framework is implemented on four products: Platform, PDE, JDT and CDT of Eclipse defect tracking dataset fetched from Github repository (Lamkanfi et al., 2013). Table 1 enumerate the preferred products along with absolute number of components and number of reports obtained from each incremental modification carried out in the lifecycle of software system. Number of reports are nothing but the bugs extracted with respect to modifications in the products.

Each product contains ten separate XML files, in which bug attributes are stored. The files selected for testing motive illustrated with description in Table 2. Every file is associated with the priority to fix the bug, severity level of the bug, the software application and version of that application to which bug is related, the sub modules of the system and the operating system for which bug is found, current state of the bug, resolution of the bug and identifier of the bug. Also, the attributes 'report ID', 'opening-time' (time when bug reported) and 'assigned_to' remain unchanged during the complete life cycle of the bug.

Table 1. Eclipse dataset products with corresponding number of components and bugs

Product type	No. of components	No. of bugs
Platform	22	24775
PDE	5	5655
JDT	6	10814
CDT	20	5640

Table 2. Different attributes selected for products in eclipse defect tracking dataset

Sr.No	Attribute	Description
1	Priority	The priority denotes how soon the bug should be fixed. This attribute typically varies between P1 to P5 where P1 denotes the highest priority.
2	Severity	The impact of the bug on the software system. This attribute varies between trivial, minor, normal, major, critical and blocker.
3	Product	The particular software application the bug is related to.
4	Component	The relevant subsystem of the product for the reported bug.
5	Bug_status	The attribute indicates the current state of a bug. The value of this attribute varies between unconfirmed, new, assigned, reopened, ready, resolved, verified.
6	Resolution	This attribute indicates what happened to this bug. The value of this attribute varies between fixed, invalid, won't fix, duplicate, works for me, incomplete.
7	Assigned_to	The identifier of the developer who got assigned the bug.
8	CC	Users who are interested in the progress of this bug.
9	Version	The version of the product the bug was found in.
10	Op_sys	The operating system against which the bug is reported.

Table 3. Extracted dependencies (Sample from large output)

Report ID	Assigned_to	Bug_status	CC	OP_sys	Product	Severity	Priority	Resolution	Version	Component
1136246610	Null	New	Null	Windows XP	JDT	major	Null	Null	3.2	Core
1136246657	Null	RESOLVED	Null	Null	Null	Null	Null	INVALID	Null	Null
1136258575	Null	New	Null	Windows XP	JDT	normal	Null	Null	3.2	UI
1136262170	Null	RESOLVED	Null	Null	Null	Null	Null	FIXED	Null	Null
1136271526	jdt-text-inbox@eclipse.org	New	Null	Windows XP	JDT	normal	Null	Null	3.2	Text
1136273412	Null	RESOLVED	Null	Null	Null	Null	Null	REMIND	Null	Null
1136279650	Null	REOPENED	Null	Null	Null	Null	Null	Null	Null	Null
1136279674	Olivier_Thomann@ca.ibm.com	New	Null	Null	Null	Null	Null	Null	Null	Core
1136296036	Null	RESOLVED	Null	Null	Null	Null	Null	FIXED	Null	Null
1139997327	Null	VERIFIED	Null	Null	Null	Null	Null	Null	Null	Null
1136263498	Platform-UI-Inbox@eclipse.org	New	Null	Windows XP	Platform	normal	Null	Null	3.2	UI
1136283656	Null	RESOLVED	Null	Null	Null	Null	Null	WORKSFORME	Null	Null

5. RESULTS AND DISCUSSION

This section describes the performance evaluation of the proposed model (NDB-FCMPM) and strengths of the NDB-FCM clustering method with techniques discussed in past. During pre-processing stage, all the XML files converted into CSV format based upon 'report ID' and 'when-tag'. This tag constitutes the reporting time of a bug. After that, dependencies generated between the bugs using the most stable information about any bug i.e. 'opening_time' and 'report ID' as both remains unchanged throughout the whole life cycle of a bug. Table 3, represents the output generated after applying dependency structure formation algorithm. For understanding, data for few report IDs are presented below for product CDT.

The next step in proposed methodology is feature extraction, which aims to convert unique string values form Table 3 with respect to each attribute into numeric value in an incremental manner starting from '0'. For example, for attribute 'assigned_to' and 'report ID- 1136246610', string 'NULL' is replaced with '0'. Next three IDs also contains 'NULL' string for same attribute, so their values are also replaced by '0'. Next 'report ID- 1136271526' contains string 'jdt-text-inbox@eclipse.org', we will assign '1' to this string and so on. Similarly, string entries of all columns/attributes of Table 3 replaced with numeric values. Table 4 summarizes the results of the feature extraction step.

Table 4. Encoded data generated after feature extraction

Report ID	Assigned_to	Bug_status	CC	OP_sys	Product	Severity	Priority	Resolution	Version	Component
1136246610	0	0	0	0	0	0	0	0	1	0
1136246657	0	1	0	1	1	1	0	1	0	1
1136258575	0	0	0	0	0	2	0	0	1	2
1136262170	0	1	0	1	1	1	0	1	0	1
1136271526	1	0	0	0	0	2	0	0	1	3
1136273412	0	1	0	1	1	1	0	1	0	1
1136279650	0	2	0	1	1	1	0	1	0	1
1136279674	2	0	0	1	1	1	0	1	0	0
1136296036	0	1	0	1	1	1	0	1	0	1
1139997327	0	3	0	1	1	1	0	1	0	1
1136263498	3	0	0	0	2	2	0	0	1	2
1136283656	0	1	0	1	1	1	0	1	0	1

Table 5. Time taken to cluster

		Time taken to cluster (in sec)		
S. No.	Reports count	K-means	FCM	Proposed
1	250	0.02	0.03	0.017
2	500	0.02	0.09	0.019
3	750	0.02	0.2	0.02
4	1000	0.03	0.3	0.028
5	1250	0.01	0.44	0.011

5.1 Performance Analysis

This section discusses the performance of proposed fuzzy clustering algorithm.

5.1.1 Metrics Evaluation

To scale the performance of the model proposed, significant metrics needed. To evaluate the algorithm under consideration for performing clustering and prioritization - Fuzzy Rand Index, run time, APFD measures used respectively.

Run time performance measure: The proposed fuzzy clustering algorithm evaluated with respect to time required for clustering and compared with k-means and Fuzzy C mean for analysis. Table 5 shows the attributes of data i.e. number of classes (5), report count and number of dimensions (2). Results proved that proposed fuzzy clustering algorithm outperform the clustering algorithms K-means and FCM stated in literature.

Fig. 2, demonstrate the pivot chart showing the run time accuracy of the proposed density based fuzzy C mean (NDB-FCM) in comparison to K-means and FCM algorithm. The results ascertained that finding maximum number of clusters prior automatically instead of defining them manually and then reaching to optimal number of cluster at later stage in clustering, supports quick convergence.

Classification accuracy measure

To compute the closeness between two clusters "Fuzzy Rand Index" is used (Campello, 2007) for the fuzzy clustering algorithms and "Rand Index" is used for K-means. Its value lies between 0 and 1, where '0' indicate utter dissimilarity and '1' indicates absolute similarity. Performance of the clustering algorithm considered high if the value of Rand index converges towards one. Table 6 represents the classification accuracy of the proposed algorithm for the product CDT in comparison to K-mean and FCM algorithms.

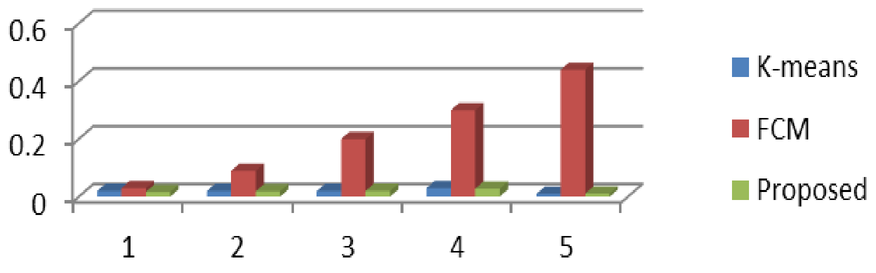

Fig. 2. Comparison of time taken to cluster by K-mean, FCM and NDB-FCM

Table 6. Classification accuracy measurement

S. No.	Reports count	Classification accuracy		
		K-means	FCM	Proposed
1	250	0.6	0.70	0.74
2	500	0.67	0.91	0.95
3	750	0.72	0.95	0.98
4	1000	0.69	0.94	0.96
5	1250	0.67	0.91	0.94

Table 7. Metrics estimation for each product with non-prioritized and prioritized test case

	APFD	
Products	Non-prioritized	Prioritized
CDT	0.39	0.55
JDT	0.13	0.22
PDE	0.54	0.67
Platform	0.53	0.71

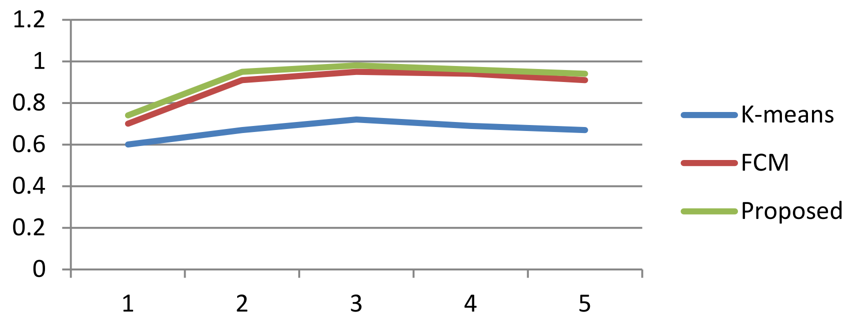

Fig. 3. Classification accuracy with respect to K-mean, FCM and proposed algorithm

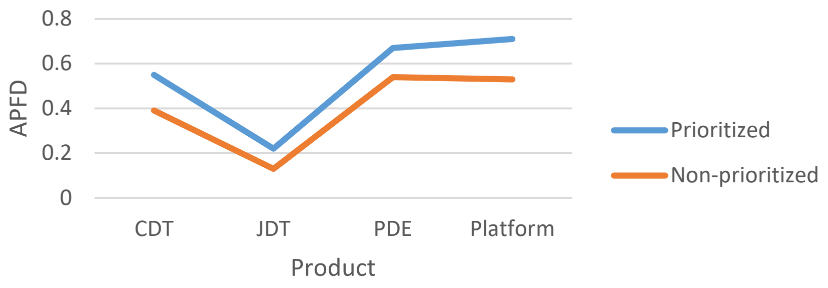

Fig. 4. APFD for prioritized and non-prioritized test cases using NDB-FCMPM

The analysis from Fig. 3, depicts that the classification strength of the existing fuzzy C mean clustering can be improved by adding the newly derived initialize membership function. This initialize membership function provides much accurate probability of membership values of an item with all clusters in the initial partition matrix.

APFD

To quantify the aim of improving the rate of early fault detection, here we used a metric known as average percentage of fault detected proposed by Elbaum et al. (2002). It is a measure that manifest how rapidly flaws can identified for a particular test suite in a system. The formula for calculating APFD is in Equation (8).

$$\text{APFD} = \mathbf{1} - \left(\frac{\mathbf{FDT_1+FDT_2+\ldots\ldots+FDT_m}}{\mathbf{mn}}\right) + \left(\frac{\mathbf{1}}{\mathbf{2m}}\right) \quad (8)$$

Where 'FDT' represents the fault detected at very first time among the test cases, 'm' indicates the whole test cases and 'n' refers an entire number of faults. The higher the value of APFD, faster is the rate of fault detection. With this motive, APFD metrics estimated for all products (summarized in Table 7) and examined for both cases (test cases with prioritization and without prioritization).

The results ascertained that prioritized test cases always results in improved rate of early fault detection in contrast to non-prioritized approach. The line chart in Fig. 4, obtained using data in Table 5 indicates the performance analysis of prioritized and un-prioritized test cases.

Table 8. Total Time taken to prioritize by eclipse products

Prioritization time (Ms)				
S.No	Product	Component	No. of bugs	Time
1	CDT	20	5640	390
2	JDT	6	10814	756
3	PDE	5	5655	446
4	Platform	22	24775	2078

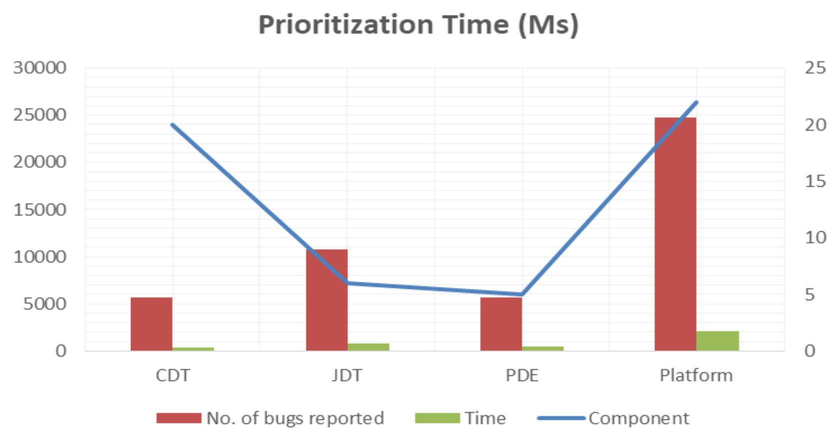

Fig. 5. Trend of prioritization time against number of bugs and number of components for all products

Also, it is concluded that the total prioritization time taken to prioritize test cases for different products is proportional to the number of bugs reported and not impacted by the number of components in a system. Illustration of prioritization time along with number of bugs reported and number of components is in Table 8.

Fig. 5, generated using data in Table 8 illustrate that there is no relation between the prioritization time and the size of the system, i.e. if the component count increases while modifying a software, the quality of the system (in terms of total prioritization time) is not compromised. However, the total bugs encountered affect prioritization time. Fewer the bugs reported, shorter is prioritization time and vice-versa.

6. CONCLUSION

Not all bugs are of equal importance as per defined customer's requirements. They can classified based on the severity impact viz. minor, major, severe and critical. Therefore, for maintaining the quality of the software systems, bugs need to be detect and debug in early phase of development cycle. The proposed test case prioritization (NDS-FCMPM) model is an effort in this direction that examined the dependency structure of the software system at preliminary stage using improved fuzzy C mean clustering algorithm. The research work discusses the three different algorithm to achieve the efficient test case prioritization: a 'novel dependency structure formation algorithm', NDB-FCM clustering with newly derived initialize membership function and a severity based prioritization algorithm. The experimental results for prioritization model validated with APFD, and the improvement of an average of 42%, observed in detecting fault at early stage when test cases prioritized. Also, suggested DB-FCM approach provides better classification accuracy and rum time over fuzzy C mean and reduced the number of iterations to find the optimal number of clusters. This enhance the rate of convergence in spite of the various fuzzy calculation involved in comparison to the other fuzzy and non-fuzzy clustering algorithm discussed in literature for test case prioritization. The present research focused on specific 'feature vector extraction' method for transformation, in future principal component analysis method can adopted to enhance the efficacy of feature extraction. In addition, grid based fuzzy clustering approaches can be applied utilizing the products stated in the work and analogy can be drawn to find out the effectiveness.

REFERENCES

Azizi, M., Do, H. 2018. Graphite: A greedy graph-based technique for regression test case prioritization. Proceedings - 29th IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW, 245–51.

Badanahatti, S., Murthy, Y.S.S.R. 2017. Optimal test case prioritization in cloud based regression testing with aid of KFCM. International Journal of Intelligent Engineering and Systems, 10, 96–106.

Badwal, J., Raperia, H. 2013. Test case prioritization using requirements-based clustering. International Journal of Current Engineering and Technology, 3, 488–92.

Bezdek, J.C., Ehrlich, R., Full, W. 1984. FCM: The fuzzy c-means clustering algorithm. Remote Sensing. Proc. IGARSS ’88 Symposium, Edinburgh, 1988. 3, 191–203.

Campello, R.J.G.B. 2007. A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. Pattern Recognition Letters, 28, 833–41.

Carlson, R., Do, H., Denton, A. 2011. A clustering approach to improving test case prioritization: An industrial case study. IEEE International Conference on Software Maintenance, ICSM, 382–91.

Chaudhary, S. 2018. Findings and iplications of test case prioritization techniques for regression testing. International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES), 4, 1259–66.

Chaudhary, S., Jatain, A. 2020. A systematic review: Software test case prioritization techniques. International Journal of Advanced Science and Technology, 29, 12588–99.

Elbaum, S., Malishevsky, A.G., Rothermel, G. 2002. Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering, 28, 159–82.

Garg, D., Datta, A., French, T. 2013. A novel bipartite graph approach for selection and prioritisation of test cases. ACM SIGSOFT Software Engineering Notes, 38, 1–6.

Gokilavani, N., Bharathi, B. 2019. Towards the prioritization of test cases by using NDBC-FFNN.

Gokilavani, N., Bharathi, B. 2020. Based test case prioritization using K-mediods based fuzzy clustering. Proceedings of the Fourth International Conference on Trends in Electronics and Informatics (ICOEI 2020), Icoei: 567–72.

Hasan, A., Rahman, A., Siddik, S. 2017. Test case prioritization based on dissimilarity clustering using historical data analysis. Springer Nature, 269–281.

Hüllermeier, E., Rifqi, M. 2009. A fuzzy variant of the rand index for comparing clustering structures. 2009 International Fuzzy Systems Association World Congress and 2009 European Society for Fuzzy Logic and Technology Conference, IFSA-EUSFLAT 2009 - Proceedings, 1294–98.

Indumathi, C.P., Selvamani, K. 2015. Test cases prioritization using open dependency structure algorithm. Procedia - Procedia Computer Science 48 (Iccc), 250–55.

Ju, Q., Zhou, D. 2016. Prioritizing test cases for memory leaks in android applications. Journal of Computer Science and Technology, 31, 869–82.

Kaur, S., Ghai, S. 2016. Performance enhancement in Hill-climbing approach for test case prioritization using functional dependency technique. International Journal of Software Engineering and Its Applications, 10, 25–38.

Lamkanfi, A., Pérez, J., Demeyer, S. 2013. The eclipse and mozilla defect tracking dataset: A genuine dataset for mining bug information. IEEE International Working Conference on Mining Software Repositories, 203–6.

Lichade, S.S., Thakur, P. 2016. A hybrid tree based approach to regression testing using clustering algorithm. International Journal of Engineering Trends and Technology, 37, 408–12.

Mani, P., Prasanna, M. 2017. Test case generation for real-time system software using specification diagram. Journal of Engineering Science and Technology, 12, 860–74.

Miranda, B., Cruciani, E., Verdecchia, R., Bertolino, A. 2018. FAST approaches to scalable similarity-based test case prioritization. Proceedings - International Conference on Software Engineering 2018-Janua.

Mishra, D.B., Panda, N., Mishra, R., Acharya, A.A. 2019. Total fault exposing potential based test case prioritization using genetic algorithm. International Journal of Information Technology, 11, 633–37.

Pang, Y., Xue, X., Akbar, A. 2017. A clustering-based test case classification technique for enhancing regression testing. Journal of Software, 12, 153–64.

Praba, M.R., Mala, D.J. 2011. Critical component analyzer - A Novel Test Prioritization framework for component based real time systems. 2011 5th Malaysian Conference in Software Engineering, MySEC 2011, 281–86.

Rajarathinam, K., Natarajan, S. 2013. Test suite prioritisation using trace events technique. IET Software, 7, 85–92.

Raju, S., Uma, G.V. 2012. An efficient method to achieve effective test case prioritization in regression testing using prioritization factors. Asian Journal of Information Technology, 11, 169–80.

Rothermel, G., Untcn, R.H., Chu, C., Harrold, M.J. 2001. Prioritizing test cases for regression testing. IEEE Transactions on Software Engineering, 27, 929–48.

Shrivathsan, A.D., Ravichandran, K.S., Krishankumar, R., Sangeetha, V., Kar, S., Ziemba, P., Jankowski, J. 2019. Novel fuzzy clustering methods for test case prioritization in software projects. Symmetry, 11, 1–22.

Yi, B., Li, Z., Guo, J., Zhao, R. 2018. Concrete Hyperheuristic Framework for Test Case Prioritization. Journal of Software: Evolution and Process, 30, 1–24.

Yoo, S., Harman, M. 2010. Regression testing minimization, Selection and prioritization: Asurvey S. Software Testing Verification and Reliability, 67–120.

Yoo, S., Harman, M. Tonella, P., Susi, A. 2009. Clustering test cases to achieve effective & scalable prioritisation incorporating expert knowledge. Proceedings of the 18th International Symposium on Software Testing and Analysis, ISSTA 2009.