2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI) | 979-8-3503-1590-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ACCAI58221.2023.10199547

Fuzzy and ANN based model for Test case prioritization for Regression testing

Dr.B.Nithya and Dr.B.G.Prasanthi

Department of Computer Science St Joseph's University Bangalore, India

E-mail : nithyamadhusuden@gmail.com,nithyamadhusuden@sjc.edu.in nitai2009@gmail.com, prasanthi.b.g@sju.edu.in

Abstract- This research article performs the prioritization of the test case to test the software system after the occurrence of changes for Regression testing. The test expert here will categorize the sets as Optimistic test cases and Pessimistic test cases as formatted data for preprocessing by the Fuzzy rules. The optimistic test cases ensure that they are considered for regression testing by the tester. They are allowed to go into the next phase for deciding the prioritization. The test case is expected to have the details of case_id, case_name, case_details, predicted_result, obtained_result, seconds_time, and status. The ANN model deployed, gives the ranking to only Optimistic test cases by ensuring its capability to a dynamic environment. The efficiency of the regression testing on the proposed ANN model is evaluated by representing the faults, statements, and paths using the average percentage. The results provide a superior value above 95% when compared to the other methods taken in literature survey. The future scope of this ANN-based model can be used for prioritizing, selecting, and categorizing every cycle using reinforcement learning methods.

Keywords: Software testing, Test case prioritization, Regression testing, Fuzzy, ANN, APF, APS, APP.

I.INTRODUCTION

Computer engineering is not simply software and scripting. It itself is a systemic application of engineering processes in the production of some software [1]. Software testing requires a longer period of implementation within a software development phase and can be the costliest step [2]. Software testing itself is typically carried out repeatedly, even though time constraints and predetermined sources are present. Because of financial and time-required relations, software engineering groups are regularly pressured to cancel their research operations, which can trigger certain obstacles, such as software consistency concerns and customer agreement [3]. Test Case Prioritization (TCP) implementation tends, however for increase viability test in the activity of the testing of software [4].

Regression tests must be done after the alteration of the program has been made. Regression testing is a form of software testing that attempts to discover new software glitches, or regressions, after modifications, such as updates, fixes or interface changes, have been made to them in current functional and non-functional areas of a code [5]. The aim of regression testing is to ensure that new flaws have not been added by modifications such as those described above. The assessment of whether an improvement in one aspect of the software affects other parts of the software is one of the key explanations for regression testing. Testers cannot find adequate tools and enough time to effectively run the regression test. Random testing, test case collection, test case minimization and test case prioritization are well-known approaches adopted by the participants during the regression test. The work on this paper is on the Test case prioritization and preprocessing technique.

Prioritization of the test case applies the prioritizing of test cases in the test suite because of several criteria. Code's coverage, risk or critical unit, feature, characteristic, etc. may be variables. Prioritization is provided to test cases that would be beneficial for subsequent updated model models. It does not consist of any details about advances made to the commodity. It is also conceivable to rank test cases in such a way that they are useful for specific model models. Awareness of improvements that have been made in the product comprises this form of ordering.Prioritization of the test case schedules the test cases in a sequence that increases consistency in meeting those success targets. The rate of fault detection is one of the most critical success objectives [5]. Test cases should be performed in a sequence that increases the probability of fault detection and therefore detects

the most critical faults at the earliest in the life cycle of the test. Prioritization strategies for test cases have been found to be useful for optimizing regression testing practices. Although code coverage based prioritization approaches are found to be studied by most researchers, so far, test case prioritization based on criteria has not been used for studies in a cost-effective manner [6].

II. LITERATURE REVIEW:

A modern static black-box TCP technique was introduced that describes test cases using a previously unused data source in the test suite: the test case linguistic data, i.e. their identifier names, statements, and string literals. The outcome of the analytical analysis [7] shows that the static black-box TCP approach proposed outperforms current static black-box TCP techniques and has equal or stronger efficiency than two existing TCP techniques based on implementation.Seven-research suites coverage review and the findings verified that cluster-based prioritization more easily identifies faults. Using clustering methods can minimize the number of comparisons in test cases and the results are checked using a metric. The clustering-based prioritization methodology increases the rate of fault detection and is efficient. It also minimizes the slippage of defects that could be overlooked during testing [8]. Requirement factors were used for the test cases. Through clustering, the suggested solution minimizes the number of pair-wise comparisons. An approach to prioritization of fault dependencies and claimed that prioritization can be achieved in an efficient way by leveraging the dependence between faults. This paper introduced a new metric. The number of test cases also increases with every update of a software update [9].

In [10] empirically examined the compilation and prioritization of test case-based code coverage for the WebKit open source web browser engine project and discovered the technological challenges and the expected advantages of this approach by integrating it into the actual construction process. Also explained how this technique can be used in the official building context.Often some parts of the script remain unchanged due to updates and some often undergo changes, resulting in redundant test cases with respect to the untouched part. Nevertheless, any test case has to be performed to ensure consistency, resulting in an increased execution time. Therefore, to map new and old test cases, the author discussed a bipartite graph approach, removing the undesired test cases.

The percentage of observed faults is obviously the most used measure favored in primary studies by scholars [11]. In early TCP studies, this measure was initially adopted and later massively used by other researchers. This metric is used to measure how easily flaws can be found by an ordered and structured test suite.Performed surveys on emerging regression test suite optimization methods, using different tools and statistical models for regression test optimization. The author argued that a multi-objective regression test optimization technique has not been properly studied and the best candidate for regression test optimization is the fuzzy logic technique.In [12] analyzed the various commonly used approaches in the prioritization of test cases, identified the various facets of the prioritization of test cases studied and presented a framework for enhancing study work on test case prioritization and assessed the latest developments in this research field. Nine recommendations for improving the prioritization of test cases were also recommended[13].

The cumulative test cases may be sectionalized into multiple components by the number of test cases under the current multiple components-based methodology. Often, by using the test case prioritization method, any single module is prioritized [14]. Prioritized test cases with primary and secondary components are concatenated after prioritization of each module and are further performed using the test case prioritization method. As per previous study analysis, the test case prioritization approach is best for the early identification of faults. Since test case prioritization strategies do not discard test cases themselves, they are able to prevent the negative aspects that can occur when the collection of regression tests along with minimization of the test suite reduces test cases [15]. Alternatively, in situations where the elimination of test cases might be appropriate, prioritization of the test case may be addressed in combination with the collection of regression tests or also minimization techniques of the test suite to prioritize the test cases inside the preferred or decreased test suite.

Proposed model for Test case prioritization:

The objective of the test case prioritization is to

achieve the collective performance goal. The main classes used for classification in the model into subclasses of priorities are the faults, statements and paths. Each class holds the criteria of case_id, case_name, case_details, predicted_result, obtained_result, seconds_time and status.

1.1. Fuzzy based preprocessing system:

The initial work starts with the primary categorization of the test cases evolved as the Optimistic and Pessimistic test cases. The optimistic test cases are the ones that require the testing and the pessimistic test cases are the ones that can be ignored. This preprocessing is done by the Fuzzy based preprocessing system. The sets are decided by the Testing Subject Matter Expert (TSME) in a semi supervised way. The need of such preprocessing system is that to have only the needed test cases for the Regression testing. By the referred works, there are lot of effort that is put into the test suite to process unwanted test cases. Therefore, the proposed fuzzy based processing system makes the decision regarding the test case as the optimistic, pessimistic and undecided. The optimistic test cases pass through the regression testing. The pessimistic test cases are ignored from processing. Finally, the undecided test cases is sent to the tester for fitting to either optimistic or pessimistic category. The decided test case will further be fitted to the decided category in forthcoming new testing process. This is why this model is referred as the semi-supervised model due to the validation requirement.

Algorithm of Fuzzy based preprocessing system:

Initialize test cases by the numbers 1,2,..,n.
Test cases for fault =$TC - F_{1,2,\ldots,n}$
Test cases for statement =$TC - S_{1,2,\ldots,n}$
Test cases for path =$TC - P_{1,2,\ldots,n}$
Fuzzy Set {
Membership Functions {Optimistic Test Cases = TC^{+}, Pessimistic Test Cases =TC^{-}, Undecided = TC^{∞}}
If {
{$TC - F_{1,2,\ldots,n}$, $TC - S_{1,2,\ldots,n}$ and$TC - P_{1,2,\ldots,n}$ = TC^{+}}
Do
ANN_model ( )
}
If {
{$TC - F_{1,2,\ldots,n}$, $TC - S_{1,2,\ldots,n}$ and$TC - P_{1,2,\ldots,n}$ = TC^{-}}
Do
Ignore ( )
}
Else{
{Define [$TC - F_{1,2,\ldots,n}$, $TC - S_{1,2,\ldots,n}$ and$TC - P_{1,2,\ldots,n} = TC^{\infty}$]
Do
Learn_decision ( ) and TC^{∞} = null
}
} End

- ANN_model ( ) – The test cases which is successfully defined in $TC^{+}$are taken for consideration to the ANN model for class based prioritization scheme.
- Ignore ( ) – The test cases which are into TC^{-} is removed from the processing immediately to the bin. This makes sure that only the valid test cases are made into the processing of ANN model ( ).
- Learn_decision( ) –These contain the test cases which is not yet fuzzified to fall into either TC^{+} or TC^{-}. Therefore, these require manual validation for getting the test cases into correct fuzzy set. This will get into the definite category for future fuzzification of test cases by the test suite.

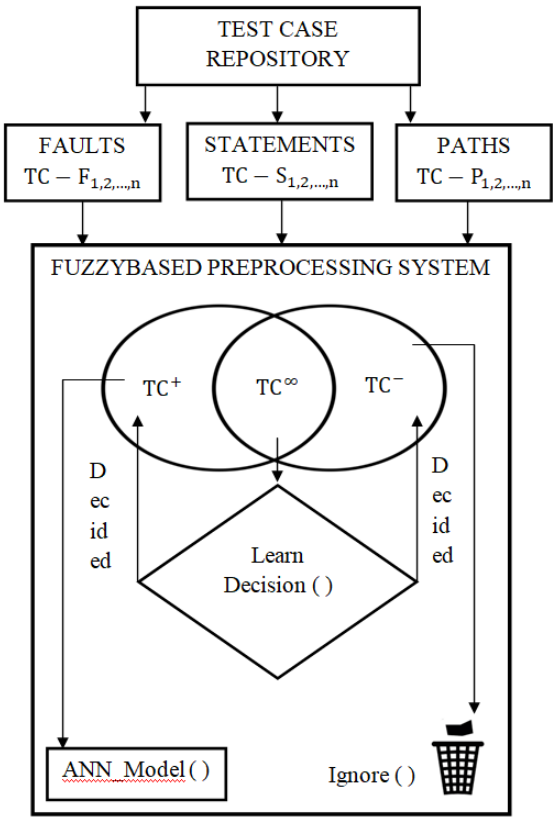

Figure 1. Flow chart representation of preprocessing of Test cases identified for Regression testing

The Figure 1, represents the flow chart representation of test case preprocessing. The step starts with the segregation of initial class as Faults, statements and paths from the test case repository. Then comes the Fuzzy based processing system containing the membership functions of TC^{+}, TC^{-} and TC^{∞}. The test cases into the class of TC^{+} directly goes for ANN_Model ( ) processing for getting the test case prioritization. The test cases into the class of TC^{-} are sent to the bin by the Ignore ( ) function. Some test cases exist in undecided TC^{∞} class. This requires the one time validation by TSME to categorize either into TC^{+}or TC^{-}. The decision by the TSME is learnt by the Fuzzy system to classify the similar test cases to either TC^{+}or TC^{-} in future. The effort that is put by the TSME is only for the single instance for a test case. This ensures that the TSME presence is not required by the Fuzzy system once it's matured or it is feeded with the test cases which were already decided.The usage of this Fuzzy based filtering scheme of test cases helps to identify the "degree's of truth" to the test cases which is initially identified. The advantage of this system is that its flexible and capable of modification of the rules or decisions by the TSME whenever its required. Even a kind of error, such asTC^{∞} is capable to be processed by the system with a skip level decision approach. The design of the system is very simple and easily understandable. This makes the initial level testers to use the decision system with a standard operating procedure without any complexity.

1.2 ANN based classifier for priority estimation:

The inputs for the values of the criteria that is to be made as the training methods for the Hidden layer of the ANN is obtained from the team of TSMEs. They were asked to provide the feedback about the seven criteria's of the Table 1 by the Google form. The values for the criteria's is in the category of Low, Medium and High impact values. The values which occupy the maximum polling for Low, Medium and High is to be decided as the dedicate factor of the corresponding category.

1.2.1 Criteria's for the Hidden layer training

1.2.1.1 Criteria 1 – case_id

The case_id is the unique id used for the test cases. The naming convention of this depends upon the test case requestor. This will have the starting sequence with numbers 1 to 9. The numbers from 1 to 9 is deployed according to the role based mapping of the deciding authority. In general cases, the responsibility pyramid will be having 1 at the apex and 9 at the base. This shows a clear idea that the case_id which starts with 1 is having high impact whereas with 9 is low impact irrespective of Faults, statements and paths.

1.2.1.2 Criteria 2 – case_name

The case_name is the name that is given for the case_id to have a naming mechanism. The name might have the brief description of the case details. This makes the tester to identify the impact of the test case or the need of the test case for testing. This criterion has the limitation of the characters due to the character number limitation. As per the survey by the google form from Table 1, the impact factor of this is high for Faults and Paths. On the other hand, the impact value of statements is low.

1.2.1.3 Criteria 3 – case_details

Case_details criteria is the detailed description of the case_name. This provides the end to end details of the test case. The effect and impact is mentioned in the criteria to have a detailed understanding of the test case. So, this being a generic free text criteria to decide the High impact value of the statements, medium impact value of the faults and low impact value of the paths. This is decided as per Table 1 as per the google form survey.

1.2.1.4 Criteria 4 – predicted_result

This criterion is the result of the test case that is at the training level obtained by probability prediction. The test suite does the evaluation as usual and the values are obtained to perform the comparison metrics. This is required for the functioning of the test case as this provides the theoretical knowledge on the test case evaluation. The knowledge outcome theoretically defines this criteria. This has high impact value for all three classes of faults, statements and paths.

1.2.1.5 Criteria 5 – obtained_result

This obtained_resultcriterion is the actual result of the test case, which is an alternate to the predicted_result. This is the only criterion along with predicted_result criterion to have all three classed with high impact value. Therefore, the skill of the testers is mandate to obtain the test results in an effective way to do the metrics of this criterion. This criterion is used eventually for both Black box and White box testing schemes.

1.2.1.6 Criteria 6 – seconds_time

This is the criterion, which measures the time for evaluating the test case. This helps to identify the loop challenges for the testing sequence by the sorting the dependency issues. This makes the effort evaluation with respect to time for a specific test case to be evaluated. The TSMEs has provided the impact of this as high for the faults, medium for paths and low for statements. This has the dual advantage for test prioritization and resource allotment for the testing scheme.

1.2.1.7 Criteria 7 – status

This status is the criterion to evaluate the success or fail or error for any test cases. This has high impact on the faults as per the TSMEs input in the survey. This has low and medium impacts for the paths and statements. This has the need to decide the test case as the needful one. Most of the test cases which passes through the Fuzzy based preprocessing process are all the needed ones. This is due to the reason of non-needed ones is filtered by the deciding mechanisms of the Fuzzy member function. The status is a dependent criterion of all the remaining six criteria included in Table 1.

Table 1. Impact Value (IV) for each criteria for modelling ANN based decision classifier

S.No	Criteria	Faults	Statements	Paths	IV - High	IV - Medium	IV - Low
1	case_id	High	High	High	3		
2	case_name	High	Low	High	2		1
3	case_details	Medium	High	Low	1	1	1
4	predicted_result	High	High	High	3		
5	obtained_result	High	High	High	3		
6	seconds_time	High	Low	Medium	1	1	1
7	status	High	Medium	Low	1	1	1

1.2.2 TF-IDF involvement for the Criteria evaluation

The term frequency - inverse document frequency (TF-IDF) is a measure that is used to do a measurement of criteria relevancy in a set of multiple test cases. This is evaluated by doing a multiplication of two metrics like the number of times the criteria happens in the test case X_a and the number of inverse document of criteria to the test cases X_b as represented in the equation 1.In alternate, the representation can be like as in equation 2 also. The $t(f)_{a,b}$ is the number of occurrences in test cases a and b. $d(f)_a$ is the number of criteria containing in test case a. X is the total number of test cases

$$TF - IDF_{a,b} = [N_a \times N_b] \quad (1)$$

$$TF - IDF_{a,b} = t(f)_{a,b} \log[\frac{X}{t(f)_a}] \quad (2)$$

The X_a is very simple by the evaluation of the raw volume of the criteria which exist in the test case. Further the frequency of it is adjusted according to the complexity of the test case. The X_b is the frequency of the criteria in a set of test cases. This is like a representation of being close to 0 the more common criteria. The metric by equation 2, is done by the total number of criteria divided by the number of criteria the test case have and by logarithmic calculation. The result would be like, to be 0 if the criteria exist in various test cases or it will be the value of 1.

1.2.3Algorithm for ANN based classifier

Initialize test cases by the numbers 1,2,..,n.
Test cases for fault = $\mathrm{VTC} - \mathrm{F}_{1,2,\ldots,n}$
Test cases for statement =$\mathrm{VTC} - \mathrm{S}_{1,2,\ldots,n}$
Test cases for path =$\mathrm{VTC} - \mathrm{P}_{1,2,\ldots,n}$
For all $\mathrm{VTC} - \mathrm{F}_{1,2,\ldots,n}$, $\mathrm{VTC} - \mathrm{S}_{1,2,\ldots,n}$, $\mathrm{VTC} - \mathrm{P}_{1,2,\ldots,n}$
Do {
$TF - IDF_{a,b}$ by equation 2
Assign the weight for all criteria and all test cases
$W_{a,b(1,2\ldots,n)} \rightarrow W_f$
ANN_model ( )
Green = W_f (high)
Red = W_f (low)
}
End
ANN_model ( )
Do {
$W_f \rightarrow$ ANN
For (I=1 to N)
{

If (ANN = = Criteria)
{Do priority assignment}
Else (Backpropagation error)
{Reinitialization of W_f}
}
}
End

The test cases are here classified as Green and Red. So, as per the requirement matrix only the validated test cases which fall in the category of green will be proceeded with the regression testing. The test cases in red are the ones, which does not need to be considered for the evaluation process of testing. These red ones have only the less or even null impacted test cases for the regression testing. The estimation of weights plays a crucial role for the evaluation of the validated test cases into red or green category for the decision of prioritization of test cases.

II. Results and Discussion

The verification of the proposed Fuzzy and ANN algorithms is done over the College Management System (CMS) developed by college, which is developed in Java platform. This ensures that the proposed method is validated in the real time practical test cases. This CMS has 5,912 lines of code and 593 test cases for evaluation. This system has some important features like staff attendance, Hostel reservation mechanism, online fee payment, searching courses, billing, lab stock maintenance, etc.This CMS has already undergone upgrades for 8 times. The current upgrade is performed with the proposed preprocessing and prioritization system for the purpose of evaluation.

2.1 Evaluation Metrics

The evaluation of the proposed system for the faults statements and paths is to be evaluated by the equations represented by 3, 4, 5, 6, 7 and 8. The evaluation of average percentages is by two different metrics with g as the general metrics for unprocessed test cases by Fuzzy and v as the validated metrics by processing test cases by Fuzzy processing. The need of the evaluation of average percentage is to estimate the success rate for the evaluation of the algorithm. The same evaluation is to be done with the comparative methods to derive the outcome for every method and to decide the superior method

2.1.1 Average percentage of Faults

The average percentage of faults is the calculation of the faults n with the criteria j by using the Fuzzy based preprocessing system APF_v as in equation 4. If Fuzzy based preprocessing system is ignored then average percentage of faults APF_g is evaluated by equation 3.

$$\mathrm{APF}_g = 1 - \frac{\{[\mathrm{TC}-\mathrm{F}_1]+[\mathrm{TC}-\mathrm{F}_2]+\cdots+[\mathrm{TC}-\mathrm{F}_n]\}}{n\,j} + \frac{1}{2j} \quad (3)$$

$$\mathrm{APF}_v = 1 - \frac{\{[\mathrm{VTC}-\mathrm{F}_1]+[\mathrm{VTC}-\mathrm{F}_2]+\cdots+[\mathrm{VTC}-\mathrm{F}_n]\}}{n\,j} + \frac{1}{2j} \quad (4)$$

2.1.2 Average percentage of Statements

The average percentage of statements is the calculation of the statements n with the criteria j by using the Fuzzy based preprocessing system APS_v as in equation 6. If Fuzzy based preprocessing system is ignored then average percentage of statements APS_g is evaluated by equation 5.

$$\mathrm{APS}_g = 1 - \frac{\{[\mathrm{TC}-\mathrm{S}_1]+[\mathrm{TC}-\mathrm{S}_2]+\cdots+[\mathrm{TC}-\mathrm{S}_n]\}}{n\,j} + \frac{1}{2j} \quad (5)$$

$$\mathrm{APS}_v = 1 - \frac{\{[\mathrm{VTC}-\mathrm{F}_1]+[\mathrm{VTC}-\mathrm{F}_2]+\cdots+[\mathrm{VTC}-\mathrm{F}_n]\}}{n\,j} + \frac{1}{2j} \quad (6)$$

2.1.3 Average percentage of Paths

The average percentage of paths is the calculation of the statements n with the criteria j by using the Fuzzy based preprocessing system APP_v as in equation 8. If Fuzzy based preprocessing system is ignored then average percentage of paths APP_g is evaluated by equation 7.

$$\mathrm{APP}_g = 1 - \frac{\{[\mathrm{TC}-\mathrm{P}_1]+[\mathrm{TC}-\mathrm{P}_2]+\cdots+[\mathrm{TC}-\mathrm{P}_n]\}}{n\,j} + \frac{1}{2j} \quad (7)$$

$$\mathrm{APP}_v = 1 - \frac{\{[\mathrm{VTC}-\mathrm{F}_1]+[\mathrm{VTC}-\mathrm{F}_2]+\cdots+[\mathrm{VTC}-\mathrm{F}_n]\}}{n\,j} + \frac{1}{2j} \quad (8)$$

Table 2. Average percentage of Faults, Statements and Paths of the ANN based proposed method with the comparative methods

Method / Algorithm Used	Faults		Statements		Paths	
	APF_g	APF_v	APS_g	APS_v	APP_g	APP_v
No Prioritization	60.07%	63.85%	57.36%	60.10%	61.80%	65.77%
Additional Greedy Algorithm	67.93%	72.66%	66.19%	70.18%	69.48%	74.97%
Multiple Criterion Based Algorithm	86.05%	88.70%	84.39%	88.12%	83.59%	89.19%
ANN Based Proposed Algorithm	90.15%	95.90%	92.12%	96.11%	90.83%	96.52%

The table 2 provides the evaluation values for APF_g, APF_v, APS_g, APS_v, APP_g, and APP_v for the No Prioritization, Additional Greedy Algorithm, Multiple Criterion Based Algorithm and ANN Based Proposed Algorithm. The initial evaluation of No prioritization provides the value in average 61.49% for all averages for the faults, statements and paths. For the Additional Greedy Algorithm, the average value of performance is 70.24%. The Multiple Criterion Based Algorithm gives the performance percentage of 86.67%. Finally, the ANN based proposed algorithm gives the supreme value of performance as 93.61%. During the overall evaluation process, the test cases which were processed without priority was not performingwell. The expectation level is not compromised by the No prioritization method. Secondly, by using the additional greedy algorithm, the values were good than the no prioritization method. Still, they were not promising. The value of average percentage of statements was better than average percentage of faults and paths. Thirdly, the multiple criterion-based algorithm gives better results than No prioritization and additional greedy algorithm. The values obtained by it was decent than the previous algorithms which were used for processing. Still, the aim is to get the superior values of at least above 90%. So, the evaluation continued to the final method, which is the proposed method, based on ANN. The values were so promising showing the best accuracy ever seen in the literature survey. The comparison of the values as in Table 2 indicates the superiority of the proposed ANN based classification method. The percentage values of APF_g is 90.15%, APF_v is 95.90%, APS_g is 92.12%, APS_v is 96.11%, APP_g is 90.83%, and APP_v 96.52%. All the values of percentage are above 90% range which indicates the lowest error rate. Of the acquired values, the ANN model is superior for the paths as it has the maximal percentage of 96.52%. Then the second is by the statements with 96.11%. Finally, it is with the faults with 96.90%. So, if the fuzzy logic is induced into the system before for the preprocessing. The efficiency goes up from 90% scale to 95% scale. This gives the significance of the need of Fuzzy based system for the processing of test cases. The Figure 2 provides the clarification on the need of Fuzzy based preprocessing system with the ANN based prioritization classification scheme. The same type of upscaling of test cases gives good performance even for the comparative methods. Therefore, it is evident that preprocessing scheme is very much needed for validating the test cases for all time.

As mentioned at the start of the section, in Figure 2. The representation of g indicates the general ANN processing without the fuzzy preprocessing. The representation of v indicates the processing by ANN along with Fuzzy preprocessing. By the color representation, the red color represents the g representation in chart without the Fuzzy preprocessing. The green color represents the representation of v in the chart with Fuzzy preprocessing. For the average percentage of faults APF_g gets the value of 90.15% and APF_v gets the value of 95.90%. On the other hand for the average percentage of statements APS_g and APS_v the percentage values obtained are 92.12% an 96.11%. The third, for the paths in the average percentage representation APP_g and APP_v gives the evaluated value of 90.83% and 96.52%. So, combining the evaluation of all the values. The representation of the system is to be totally with the best representation as the green color and the less performing representation in red color. The way of fuzzy based preprocessing makes the quality of the software testing to be improved. The issues of the test cases, which is used without fuzzy, may be crucial like the crashes of the software, leakage of the memory and the exceptions that are unhandled. If there are any test cases failing, again fails to be

noticed by the testers due to the time limitation and resource limitation might cause serious problems to the software.

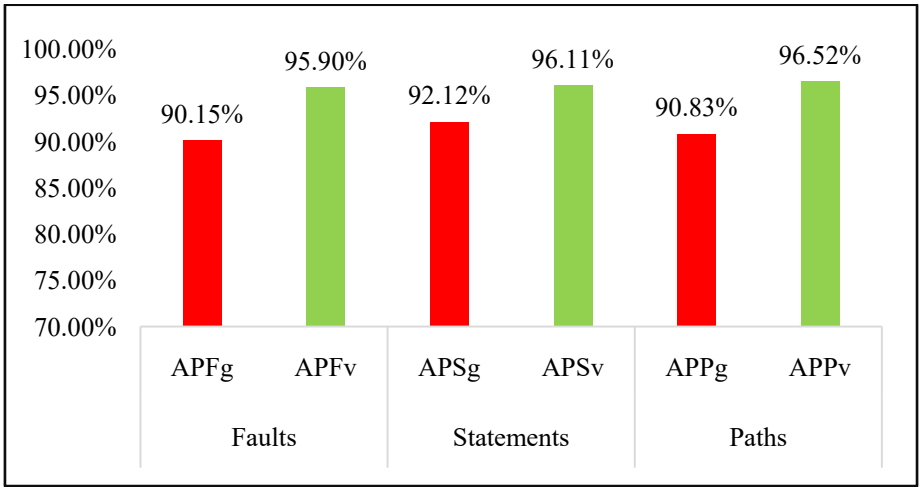

Figure 2. Performance evaluation of the ANN based proposed method with Fuzzy preprocessing and without Fuzzy preprocessing

The fuzzy based preprocessing is used to shorten the test cases. This makes the test case evaluation to be efficient enough to get the good results. The usage of this gives the effective result for Black box testing, White box testing, Debugging and Beta testing methods. The usage of Fuzzy makes the cost effective way for the evaluation of the test cases. If not, the test cases will go as bulk and makes the system to face dead lock issues for the processing.The fuzzy logic as per the evaluation metrics has to evolve with the Six steps which starts with the identification of the targeting system, Input identification, Fuzzy data generation, Executing of the fuzzy data tests, monitoring the behavior of the system, and finally the logging of the valid test cases.

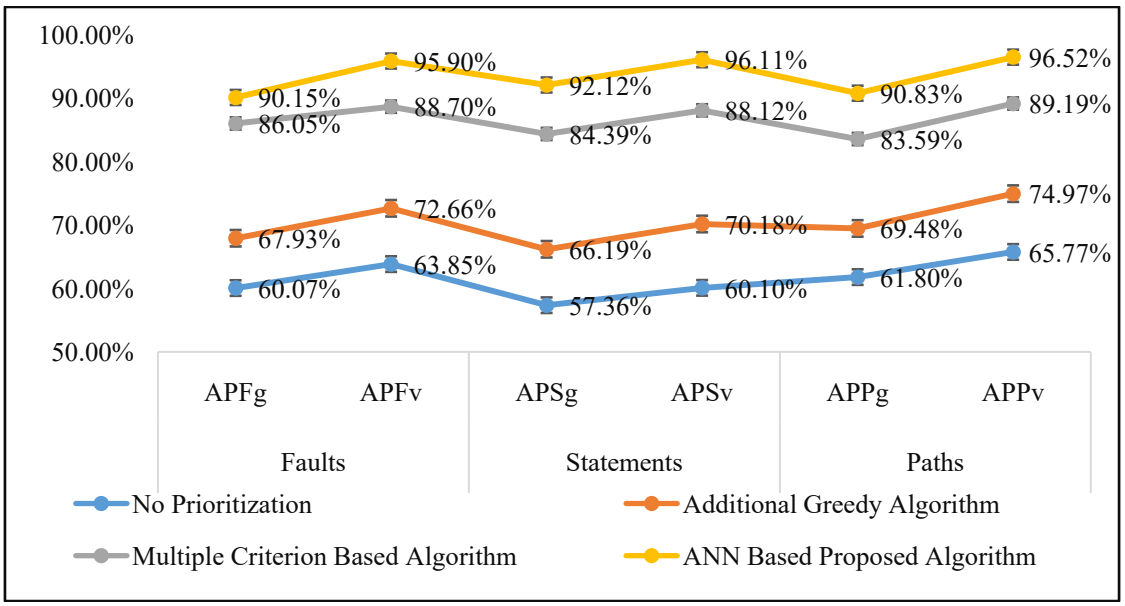

Figure 3. Comparison of performance for all methods with and without Fuzzy based preprocessing

The Figure 3 by the evaluation shows the details of all method, which is used, for comparison with or without Fuzzy based preprocessing. The outcome from all the methods show the output giving the values as much higher than the other methods without the usage of preprocessing. This indicates that every system should have a preprocessing system to do the initial filtering of the test cases to

obtain the dedicated output for the performance evaluation. The suitability of the Fuzzy based filtering method is fine for the methods used for comparison in this paper. In case for other methods the fitting to be evaluated separately.

III. CONCLUSION:

This paper discussed about the various test case prioritization methods used by the researchers until now. The dependency issues with the existing works is also discussed by the survey from the test experts. In think of all such feedbacks, the need of preprocessing for the test cases to have an effective test case prioritization is identified. This made a conclusion to develop Fuzzy based preprocessing system with three member functions. The next stage of ANN based test case prioritization is capable of providing the test case prioritization results for average percentage of faults, statements and paths. In overall, usage of only ANN based test case prioritization gives the performance average of 90% and above than the comparative methods. On the other hand, the usage of preprocessing system before ANN based prioritization gives 95% above performance average. This makes to conclude that ANN method's results is superior than the other comparative methods. Additionally, preprocessing by Fuzzy makes the ANN method to function super superior due to effective utilization of testers and resources.

REFERENCES:

[1] A D Shrivathsan (2019): Novel fuzzy clustering methods for test case prioritization in Software Projects, Symmetry (Basel), Vol. No. 11, pp.1 – 22.

[2] Ali N B, Engstrom E, and Taromirad M (2019): On the search for industry-relevant regression testing research, Empirical Software Engineering, Vol. No. 24, pp. 2020 –2055.

[3] Anwar Z, and Ashan A (2014): Exploration and analysis of regression test suite optimization, ACM SIGSOFT Software Engineering Notes, Vol. No. 39(1), pp. 1 - 5.

[4] B Miranda, and A Bertolino (2016) Scope-aided test prioritization, selection and minimization for software reuse, Journal of System Software, Vol. No 1, pp. 1 – 22.

[5] Beszedes A, Gergely T, Schrettner L, Jasz J, Lango L and Gyimothy T (2012): Code coverage-based regression test selection and prioritization in WebKit, Conference Proceedings, 28th IEEE International Conference on Software Maintenance, pp. 46 – 55.

[6] CagatayCatal, and Deepti Mishra (2012): Test case prioritization: a systematic mapping study, Software Quality Journal, Vol. Np. 21(3), pp. 445 - 478.

[7] D Hao, L Zhang, L Zang, Y Wang, X Wu, and T Xie (2016): To be optimal or not in test-case prioritization, IEEE Transactions on Software Engineering, Vol. No. 42(5), pp. 490 – 504.

[8] Engstrom E, Petersen K, Ali N B, and Bjarnason E (2017): SERP Test: a taxonomy for supporting industry academia communication, Software Quallity Journal, Vol. No. 2(4), pp. 1269 – 1305

[9] Franch X, Fernandez D M, Oriol M, Vogelsang A, Heldal R, Knauss E, Travassos G H, Carver J C, Dieste O, and Zimmermann T (2017): How do practitioners perceive the relevance of requirements engineering research - an ongoing study, Conference Proceedings, IEEE 25th International requirements engineering conference, pp. 382 – 387.

[10] G J Myers, T M Thomas, and C Sandler (2004): The Art of Software Testing, Vol. No. 1, John Wiley & Sons, Print.

[11] G Rothermel, R H Untch, C C Chu, and M J Harrold (1999): Test case prioritization: an empirical study in: Software Maintenance, Conference Proceedings ICSM 1999, pp. 179 – 188.

[12] Haghighatkhah A , Mantyla M, Oivo M, and Kuvaja P (2018): Test Prioritization in Continuous Integration Environments, Journal of Systems and Software, Vol. No. 146, pp. 80 - 98.

[13] Kazmi R, Jawawi D N A, Mohamad R, and Ghani I (2017): Effective regression test case selection: a systematic literature review, ACM Computer Survey, Vol. No 50(2), pp. 1 – 29.

[14] L Mei (2015): A subsumption hierarchy of test case prioritization for composite services, IEEE Transactions on Service Computing, Vol. No. 8(5), pp. 658 – 673.

[15] Minhas N M, Petersen K, Ali N B, and Wnuk K (2017): Regression testing goals view of practitioners and researchers, Conference Proceedings, 24th Asia-Pacific software engineering conference workshops, pp. 25 – 31.