A CLUSTERING-DEFECT PREDICTION TECHNIQUE FOR TEST CASE SELECTION

Rongshang CHEN[1], Lei XIAO[2], Weiwei ZHUANG[3]

Regression testing is an important and expensive process in software maintenance cycle life. Test case reduction, test case prioritization and test case selection are all effective techniques for improving the effectiveness of regression testing. In this paper, we investigate whether a clustering-defect prediction technique can improve the effectiveness of test case selection technique. We cluster certain test cases with coverage methods based on similarity, and prioritize them according to the results of defect prediction. Finally, we perform two test case selection techniques, namely PICK-ONE and PICK-MORE, and compare their execution results through an empirical study. The experiment results indicate that PICK-ONE outperforms PICK-MORE when executing 25 percent test cases rather than 75 percent. They are better than the random test case selection technique.

Keywords: Clustering-defect Prediction; Test Case Selection; Missed Faults

1. Introduction

Regression testing is performed with modified software by reusing the existing test cases to verify the correctness of software and that the modifications have not produced new faults. It is reported that regression testing has become a bottleneck in the process of software development [1], but it is a very important part in software development and maintenance. In the past twenty years, many researchers in the field of software engineering have paid attention to looking for the most cost-effective execution approach for regression testing. The main strategies include test case minimization [3], regression test selection [2], and test case prioritization (TCP) [4]. The purpose of regression test selection is to choose an effective subset from the original test case set [6]. Test case prioritization is to reorder test cases based on a given criteria, to excite test cases earlier and achieve a higher probability to find fault. These techniques are effective strategies proven by many researchers previously. Test case prioritization and test case selection all concern about the way to effectively use the existing test cases. According to certain rules, partial test cases could be selected and executed by using the prior techniques [7][8]. The previous empirical studies have demonstrated that the test

[1] College of Computer and Information Engineering, Xiamen University of Technology, Xiamen, China, e-mail: rschen@xmut.edu.cn

2 College of Computer and Information Engineering, Xiamen University of Technology, Xiamen and School of Computer Engineering and Science, Shanghai University, Shanghai, China;

3 College of Computer and Information Engineering, Xiamen University of Technology, Xiamen, China

cases with common properties often have the same fault detection ability. This motivates many researchers to use cluster techniques for test case selection [9][10][11][13][14]. Cluster techniques firstly divide test cases into groups according to certain rules, and then select certain test cases from each group. There are two test case selection strategies, namely one-per-cluster sampling and failure-pursuit sampling [12]. The former selects exactly one test case from each cluster, and the latter chooses k nearest neighbors of any failure.

Inspired by the failure-pursuit sampling mentioned in this paper, we introduce a test case selection based on clustering algorithm in combination with defect prediction. For clustering, we collect the function call profile of each test case as clustering feature. The results of defect prediction guide the test case selection. Our experiment results indicate that this technique can catch more faults and reduce the number of slip faults when test cases must be omitted because of time and budget constraints.

The rest of this paper is organized as follows. Section 2 discusses the methodology. Section 3 presents the process and results analysis of the empirical study. Section 4 describes the related works about test case selection and prioritization. Section 5 addresses the threats to validity. Section 6 draws a conclusion and discusses the future work.

2. Methodology

This section mainly describes the process of clustering test cases, test cases selection and test case prioritization. Our approach consists of 5 main steps, namely features extraction, clustering of test cases, defect prediction, test case prioritization and test case selection, as shown in Fig. 1.

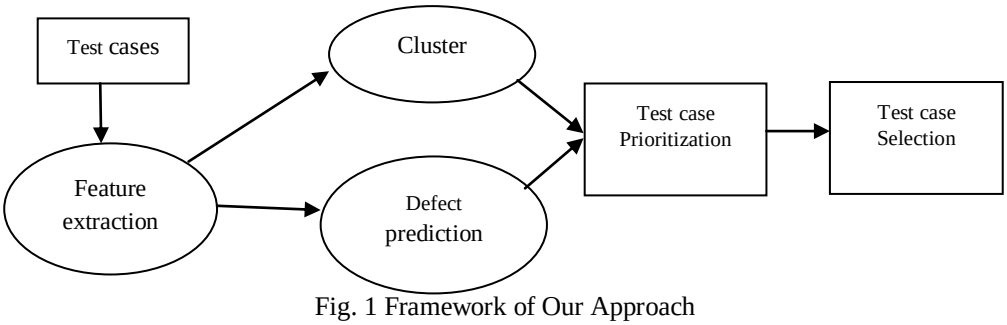

Fig. 1 Framework of Our Approach

2.1. Feature Extraction and Distance Measure

Generally, we can easily acquire the function call profile in the process of executing test cases. In this paper, we select function call profile as clustering feature. Function call profile is a binary vector. Each bit value represents whether

the corresponding function is called or not. If the function is called, the bit value is 1; otherwise, it is 0. In regression testing, we can capture the binary vector corresponding to each test case. To obtain the pair-wise distances between test cases, we use Euclidean distance as dissimilarity function. The formula is as follows.

$$D(t,t') = \sqrt{\sum_{i=1}^{n} (f_i - f_i')^2} \quad (1)$$

Wherein, if the corresponding function call profile binary vector of test case t and t' are $<f_1, f_2 \ldots f_n>$ and $<f_1', f_2', \ldots, f_n'>$, n is the number of function. If the ith function is called by test case t, fi =1; otherwise, it is 0. The situations of t' and f_1' are similar.

2.2. Clustering of Test Cases

Most of the exiting techniques that apply clustering algorithm to test case prioritization such as those introduced in literatures [9], [11] and [14] choose agglomerative hierarchical clustering method. Fig. 2 is an example of a hierarchical tree. Hierarchical clustering algorithm proceeds by using a new similarity table. At later stages, an element (a test case in our paper) may be merged with another element or a cluster, or two clusters may be merged, based on the average distance between test cases that define the clusters. As a distance function, the result of hierarchical clustering algorithm is a tree of test cases that return one clustering with k clusters for every k from 1 to the total number of test cases. However, we can find that the time complexity and space complexity of the later are better than those of the former through a comparison between hierarchical clustering algorithm and the k-means algorithm. In order to improve execution efficiency, we choose k-means to cluster test cases in this paper.

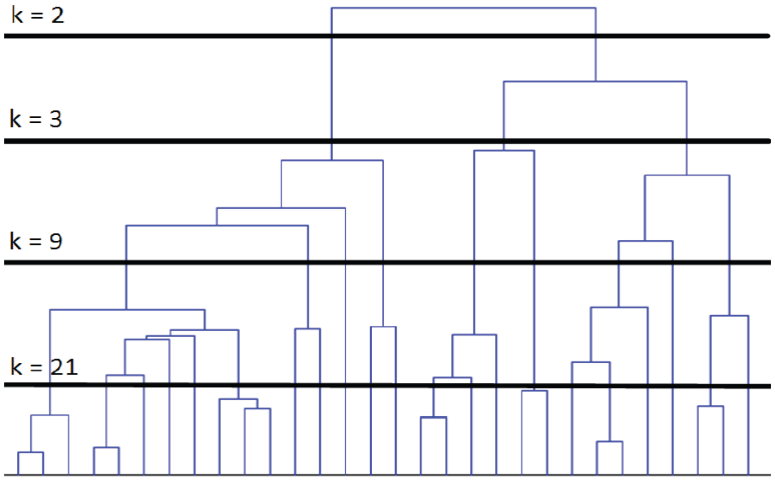

Fig. 2 an example of hierarchical tree

The k-means algorithm is a simple and common method in clustering analysis. It automatically partitions n data (test cases in this paper) into K clusters. Each datum belongs to the same cluster with the nearest distance [15]. The formula is as follows:

$$V = \sum_{j=1}^{k} \sum_{x_i \in c_j} (x_i - u_j)^2 \quad (2)$$

Wherein, uj is the mean variable. xi denotes an instance in a cluster cj , and k signifies the number of clusters. According to the size of our experiment object, the best k value is 3. As reported by Songyu Chen et al. [16], k-means is a suitable clustering algorithm compared with the others, such as hierarchical cluster and DBScan. Consequently, we choose k-means clustering algorithm to classify the test cases in our work.

2.3. Defect Prediction

Software defect prediction has been introduced by many researchers in the previous literatures [17, 18]. Last year, we introduced an approach assembling cluster algorithm and fault prediction for test case prioritization [17]. Our experiment results encourage us to do this work. In this work, the process of defect prediction is the same as that described in literature [17]. We adopt PREST tool to capture the code. Software faults prediction is based on the code features corresponding to test cases. A demo of defect prediction results is shown in Table 1. The first row lists the items, and the third row presents the corresponding parameters of Test 2. Its probability of fault is 92.33%. The second row gives the corresponding parameters of Test 1. Its probability of non-fault is 85.24%.

Table 1

Demo of Defect Prediction Results

Test Case ID	Probability	Result	Label
Test 1	0.8524	0	non-fault
Test 2	0.9233	1	fault

2.4. Test Case Prioritization

Test case selection is defined as follows: Given the program P, the modified version of P,P', and a test suite, T finds a subset of T,T', with which we test P'[19]. The aim of test case selection is to find the subset T'. Many existing test case selection techniques mainly solve the problem of how to find the best subset T', which can reduce the cost of regression testing and maintain similar defect detection capability with the original test suite T. In this paper, we apply test case prioritization technique before test case selection to solve this problem. The goal of test case prioritization is to maximize early defect detection. Test case prioritization technique involves inter-cluster and intra-cluster prioritization. Intra-cluster prioritization considers about two circumstances. One is defect prediction probability, and the other is the distance from the center point of each test case. In consequence, the process of intra-cluster prioritization is divided into two steps. The first stop is to sort the test cases which have the "fault" label according to the defect predication probability. The bigger the probability is, the higher the priority is. The second step is to sort the remaining test cases which have the "non-fault" label according to the distance from the center point. The closer the center point is, the higher the priority is. Inter-cluster prioritization mainly applies the lines of code coverage through every test case.

2.5. Test Case Selection

As stated previously, the purpose of test case selection is to find the ideal subset T' instead of re-running the complete test suite. In our work, we select three subsets 25%, 50%, and 75%, which means to shorten the test cases by 75%, 50%, and 25% in regression testing. After executing the test case prioritization, we choose two test case selection techniques. One is to pick the first test case of all the clusters. Then, all the selected test cases are sorted according to code coverage information and put into an empty prioritized list of test case in turns. Subsequently, we pick the second test case from all the clusters, and repeat the same process until all the test cases are picked up (PICK-ONE). The other technique is to firstly pick all the test cases with "fault" lablel (The probability is above 80% in this paper.) from every cluster, and then pick the remainder according to the first technique (PICK-MORE). For example, we suppose that we have three clusters and 14 test cases (T1, T2… T14), and the details of each test

case are given in Table 2. With the prioritization technique that we have described, we reorder the test cases of each cluster, as shown in Fig. 3.

Table 2

Details of Test Case

Test Case NO	Cluster No	Defect prediction value	Defect prediction probability	Distance from the center point	LOC
T01	2	1	0.8711	1.06703	31
T02	2	1	0.8715	2.932415	45
T03	1	1	0.8186	3.009338	10
T04	1	0	0.8959	1.086261	3
T05	3	0	0.896	1.086261	3
T06	1	1	0.9136	2.951646	20
T07	2	1	0.8712	2.8975	33
T08	3	1	0.879	1.9975	15
T09	2	1	0.9131	1.9975	22
T10	3	1	0.8729	3.970876	18
T11	1	0	0.8958	2.0478	10
T12	1	1	0.7108	1.9975	18
T13	1	1	0.8786	2.990107	12
T14	2	1	0.8821	2.951646	19

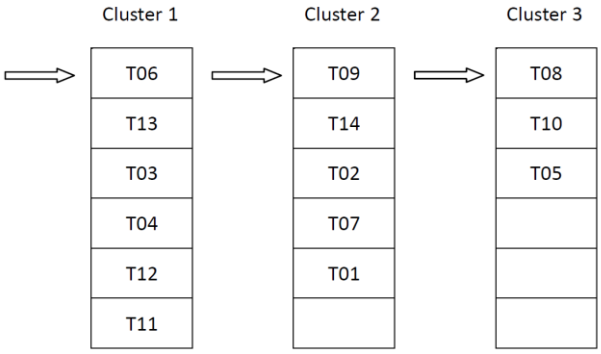

Fig. 3 an example of prioritized test cases in clusters

We choose the test cases using the two selection techniques. The results are displayed in Fig. 4 and Fig. 5.

Fig. 4 A prioritized lists using the first technique ((PICK-ONE))

Fig. 5 a prioritized list using the second technique ((PICK-MORE))

3. Case Study

The goal of our case study is to reduce the number of faults that slip through testing when selecting a subset to replace the original test suit.

3.1 Study Process

This study object has been used in [20], and data collection system is developed by one of the authors. Details are shown in Table 3.

Table 3

Study Object and Associated Data

Object	Classes	methods	Size (KLOCS)	Test Cases	Faults
DCS (V1.0)	54	81	6.75	146	38
DCS (V2.0)	70	97	7.63	172	46

All the activities of data collection, test case feature capture, defect prediction and test case clustering are similar to those described in literature [20]. In this study, we consider three test case subsets, namely TCL-25, TCL-50 and TCL-75, which represent shortening the test cases by 25%, 50%, and 75% respectively. We ignore the difference in all the faults, and assume that they have equivalent priority. In this case study, we just investigate how many test cases would slip when executing the three subsets mentioned above.

1) Independent variables: Our experiment involves two independent variables. One is the prioritization technique, and the other is the execution subset. In this work, we think about two prioritization techniques which are stated in Section 2.3 and threes subsets, namely TCL-25, TCL-50, and TCL-75.

2) Dependent variables: The goal of test case selection is to find a subset that can replace the original test suite. In other words, the aim is to execute whether the subset can find faults as the original test suite does. As a result, we count the number of missed faults of each subset as the dependent variable.

3.2 Data and Analysis

Our experiments focus on the number of missed fault only in the exaction of a subset. In this work, the number of the best cluster is 3, according to the study object and literature [20]. We observe the data change based on every subset, and find the ideal subset of test suite. Our research considers three techniques, including Random, PICK-ONE, and PICK-MORE. The results of each version are shown in Table 4. The first column lists the three execution subsets. The second and third columns list the selection techniques and corresponding number of missed faults based on each subsets of Version 1.0. The fourth and fifth columns list the information of Version 2.0.

To show the experiment results visually, we present them in line plots in Fig. 5. The figure on the left displays the results of Version 1.0, and the figure on

the right the results of Version 2.0. The X-axis presents the selected subsets of test suite, and the Y-axis the miss faults. Each line shows the experiment results of its corresponding technique. From the Table 4 and Fig. 5, we can observe that PICK-ONE and PICK-MORE are better than RANDOM technique. the left 25 percent.

Table 4

Experiment Results of Three Subsets

The Subsets of Test Cases	Version 1.0		Version 2.0	
	Techniques	No.of missed faults	Techniques	No.of missed faults
TCL-75	Random	33	Random	41
	PICK-ONE	18	PICK-ONE	28
	PICK-MORE	21	PICK-MORE	32
TCL-50	Random	30	Random	37
	PICK-ONE	11	PICK-ONE	22
	PICK-MORE	10	PICK-MORE	19
TCL-25	Random	13	Random	20
	PICK-ONE	8	PICK-ONE	9
	PICK-MORE	4	PICK-MORE	5

In other words, the clustering-defect prediction techniques outperform the random technique. By comparing two heuristics techniques, we find that PICK-ONE misses the least faults at TCL-75 of both versions, and PICK-MORE misses the least faults at TCL-25. They miss nearly the same number of faults at TCL-50. In Version1.0, the number of missed faults of the PICK-ONE is 18, and that of PICK-MORE is 21 at TCL-75. The reason is that they have different selection strategies. PICK-ONE just picks the first test case of the cluster and moves to the next cluster. Then, it repeats the same process until all test cases are picked up. PICK-MORE picks all the test cases which have “fault” label from the first cluster and moves to the next cluster. When we execute 25 percent test cases (TCL-75), PICK-ONE has the best results. It misses the least faults. According to the selection principle of PICK-ONE and PICK-MORE, the experiment result is reasonable. When we execute 25 percent test cases, PICK-MORE could find faults from most of the selection test cases. However, the executed test cases belong to the same clusters ; several test cases are found to have the same faults. The probability of finding the saman faults by PICK-ONE from several test cases is low. PICK-MORE outperforms PICK-ONE at TCL-25 (execution of 75 percent test cases). In Version 1.0, the former misses 4 faults, and the latter 8. In Version

2.0, the former misses 5 faults, and the latter 9. Based on test case prioritization and selection technique, the ability of 75 percent test cases to find faults is better than

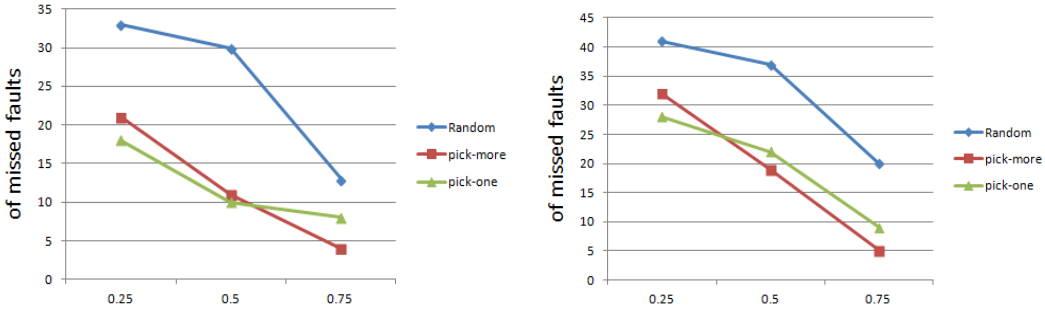

Fig. 6 Number of missed faults for Version1.0 and Version2.0

4. Background and Related Works

Test case minimisation, selection and prioritization in regression testing have been discussed by many researchers in recent decades. Yoo et al. [21] defined test suite minimisation problem, test suite selection problem and test case prioritization problem. They considered that test suite minimization is given a test suite T, and found a representative set T' of the test cases from T that satisfies all test requirements. Test case selection was made. P' indicates the modified version of program P, and T represents a test suite of satisfaction P. A subset of T,T' was found and utilized to test P'. In fact, test case prioritization is to reorder test cases, so that the most beneficial ones are executed first by just changing the execution order for test cases. No matter what kind of technique, researchers mainly focus on finding an ideal test suit or execution order to improve the effectiveness of regression testing.

Rothermel et al. [2] designed the algorithms which construct control flow graphs for the systems under testing, and its modified version. They used these graphs to select tests that execute the changed codes from the original test suite .Cao Xi et al. [22] applied greedy algorithm to test case selection based on control flow graph. Hemmati el al. [19] presented a new similarity-based selection technique for state machine-based test case selection.

The studies more relevant to our research are as follows. One technique is hunt for the test cases which may be found to have bugs. Mirarab et al. [5][23] proposed probability theory and utilized Bayesian network to incorporate source code change. They conjectured software fault-proneness for source code change, so the test cases covering the modified source codes would have a higher probability of finding bugs. In our work, we look for the test cases with a higher

probability of finding bugs through defect prediction. The other technique is to utilize the clustering algorithm to divide the test cases. There are several prioritization techniques using clustering algorithm. Yoo et al. [14] proposed the clustering of test cases to reduce the number of pair-wise comparisons. Arafeen et al. [11] put forward the requirement-based clustering approach. They firstly used requirements textual similarity to cluster requirements, then obtained the clusters of test cases of the requirement-tests traceability matrix, and finally incorporated traditional code analysis information to prioritize the test cases of each cluster. Carlson et al. [9] also proposed to test case prioritization technique based on the clustering algorithm using code coverage information. They just discussed the prioritization of test cases that belong to the same clusters.

5. Internal Validity and External Validity

Threats to internal validity:

Our empirical study executes defect prediction and k-means clustering algorithm, which needs additional time cost. Nevertheless, most existing works utility clustering algorithm test case prioritizations apply hierarchical clustering algorithm of much higher time cost. The choice of clustering features is based on the method coverage similarity between two test cases. This choice may be affected by the clustering effectiveness. However, some literatures such as [10] and [16] have selected the same clustering feature.

Threats to external validity:

The objects that we have studied are just developed by one of the authors. It is not a well-known project, and the size of KLOCs Version 2.0 is 7.63, so the number of test cases is not enough. To solve the above problem, we should select and study some famous industrial software or standard datasets.

6. Conclusion

In this paper, we do not use the traditional technique based on fault history information to prioritize test cases and user defect prediction, because new version usually has mixed test cases (the existing and new test cases) in regression testing. Under this circumstance, the fault history information is not available. Through defect prediction, we can get the probability of new test cases to find faults. In addition, software is a unity, and its components have underlying relationships, so we can conjecture the test cases with common properties and maybe similar fault detection ability. Our research considers defect prediction, and incorporates the clustering algorithms for test prioritization and selection. We compare two test case selection approaches based on three different subsets of test suite, and choose the missed faults as evaluation index.

On the basis of study results, the test case selection approach that combines defect prediction and the clustering algorithm has a better performance. Under the same circumstance, our technique misses fewer faults than the random technique. Nonetheless, our research has some limitations. Firstly, the study object is designed by one of the authors. It is a small object with non-representative activeness. Secondly, the case study just compares our techniques with the random technique. We cannot think about other test case prioritization and selection techniques such as those based on fault history information clustering algorithm or requirement clustering. Thirdly, we only choose an evaluation index, i.e. the number of missed faults, without considering the disparity among faults. All the limitations suggest several research directions of our future work.

Acknowledgments

This research work is supported by the universities and colleges in Fujian province under grants (No. JK2015033), the research fund from Xiamen University of Technology under grants (No. XYK201431), the research project of Xiamen science and technology program under grants (No.3502Z20133043) and the research project of Xiamen overseas students (NO. XRS201631401).

R E F E R E N C E S

[1]. *Jiang B, Chan W.* On the integration of test adequacy, test case prioritization, and statistical fault localization. In: Proc. of the 10th Int'l Conf. on Quality Software (QSIC). IEEE, 2010. 377−384. [doi: 10.1109/QSIC.2010.64]

[2]. *Rothermel G. A safe,* efficient regression test selection technique. Acm Transactions on Software Engineering & Methodology, 2000, 6(2):173-210.

[3] *Harrold M J, Gupta R, Soffa M L.* A methodology for controlling the size of a test suite[C]// Software Maintenance, 1990. Proceedings. Conference on. IEEE Xplore, 1990:302-310..

[4]. *CHEN Xiang,CHEN Ji-Hong,JU Xiao-Lin,Gu Qing.* Survey of Test Case Prioritization Techniques for Regression Testing. Journal of Software,2013,24(8):1695-1712

[5]. *Siavash Mirarab and Ladan Tahvildari*, "An Empirical Study on Bayesian Network-based Approach for Test Case Prioritization" 2008 International Conference on Software Testing, Verification, and Validation, pp.278-287.

[6]. *Zhang, C, Chen, Z. Y., Zhao, Z. H., Yan, S. L., Zhang, J. Y.and Xu, B. W.* An improved regression test selection technique by clustering execution profiles. In Proceedings of the 10th International Conference on Software Quality (QSIC'10), 2010, pp.171-179.

[7]. *Wong WE, Horgan J R, London S, et a1.*A study of effective regression testing in practiceLC//Proc of the 8th lEEE ln-ternational Symposium on Software Reliability Engineering Albuquerque, 1997: 264-274

[8]. *Kim Jung. Min, Porter Adam.* A history-based test prioritization technique for regression testing in resource constrained environments [c]//Proc of the 24th International Conference on Software Engineering. New York. 2002: l19-129

[9]. *R. Carlson, H. Do, and A. Denton,* "A clustering approach to improving test case prioritization: An industrial case study,"in ICSM, Sep. 2011, pp. 382–391.

[10]. *Yan S, Chen Z, Zhao Z, et al.* A Dynamic Test Cluster Sampling Strategy by Leveraging Execution Spectra Information[C]// Third International Conference on Software Testing, Verification and Validation. IEEE, 2010:147-154.
[11]. *Md. Junaid Arafeen and Hyunsook Do,*" Test Case Prioritization Using Requirement-Based Clustering,"2013 IEEE Sixth International Conference on Software Testing, Verification and Validation,Dec.2013,pp.312-321.
[12]. *Leon, Masri and Podgurski.*An Empirical Evaluation of Test Case Filtering Techniques Based On Exercising Complex Information Flows. ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA.
[13]. *D. Leon and A. Podgurski,* "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases,"in Proceedings of the International Symposium on Software Reliability Engineering, Nov. 2003, pp. 442–453.
[14]. *S. Yoo, M. Harman, P. Tonella, and A. Susi,* "Clustering test cases to achieve effective and scalable prioritization incorporating expert knowledge," in Proceedings of the International Conference on Software Testing and Analysis, Jul. 2009, pp. 201–212.
[15]. *J. MacQueen et al.,* "Some methods for classification and analysis of multivariate observations," in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, no. 281- 297. California, USA, 1967, p. 14.
[16]. *Songyu Chen, Zhenyu Chen, Zhihong Zhao, Baowen Xu, Yang Feng.* Using Semi-Supervised Clustering to Improve Regression Test Selection Techniques. 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation
[17]. *Lei Xiao, Huaikou Miao,Weiwei Zhuang,Shaojun Chen.*" Applying Assemble Clustering Algorithm and Fault Prediction to Test Case Prioritization". In SATE,NOV.2016,pp.109-116.
[18]. *J Hui-Yan,Z Mao,L Xing-Ying.* Research of Software Defect Prediction Model Based on ACO-SVM. Chinese Journal of Computers .2011, 34(6):1148-1154
[19]. *Hemmati H, Briand L, Arcuri A, et al.* An enhanced test case selection approach for model-based testing: an industrial case study[C]// Eighteenth ACM Sigsoft International Symposium on Foundations of Software Engineering. ACM, 2010:267-276.
[20]. *Lei Xiao,Huaikou Miao,Weiwei Zhuang,Shaojun Chen.*" An Empirical Study on Clustering Approach Combining Fault Prediction for Test Case Prioritization". In ICIS,May.2017.
[21]. *Yoo S, Harman M.* Regression testing minimization, selection and prioritization: a survey. Software Testing Verification & Reliability, 2012, 22(2):67-120.
[22]. *Cao X, Xu L.* Method for test case selection and execution of web application regression testing[J]. Journal of Southeast University(English Edition), 2008, 24(3):325-329.
[23]. *Mirarab S, Tahvildari L.* A Prioritization Approach for Software Test Cases Based on Bayesian Networks.[C]// Fundamental Approaches To Software Engineering, International Conference, Fase 2007, Held As. DBLP, 2007:276-290.