International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-5, March 2019

An Efficient Test Case Prioritization using Hierarchical Clustering for Enhancing Regression Testing

Gayam Rupa Sri, Addepalli Leela Supriya, Enukonda Raja Avinash Reddy, Venkata Naresh Mandhala

Abstract*: To enhance the efficiency of regression testing, we propose an experiment order approach dependent on different strategies to arrange test cases into powerful and non-viable gathering. The Hierarchal clustering methodology depends on the inclusion data got for the before arrivals of the program under test. We utilized two regular grouping calculations specifically centroid-based and hierarchal clustering. The experimental investigation results demonstrated the experiment grouping can viably recognize viable experiments with high review proportion and significant exactness rate. The paper additionally explores and looks at the execution of the proposed grouping based methodology with some different components including inclusion criteria, development of highlights, and amount of flaws in the prior discharges.

Index Terms*: Cluster, Hierarchal Clustering, Regression testing, Test cases.

I. INTRODUCTION

Grouping is a standout amongst the most in unsupervised learning procedures (for example utilized for interfacing the causative hole among info and yield perception). Grouping is "the method to divide objects into groups . In a general sense, grouping is to find the inside arrangement of unlabeled data. In grouping, we compose the data in the state of bundles or into groups.

There are different grouping systems, for example, Test case prioritization procedures plan test cases so as to upgrade their viability. Experiment prioritization worries with the recognizable proof of the ideal experiments. The motivation behind this system is to reassemble some execution objectives like rate of defect location[16], increment the viability and so forth pace of defect discovery is utilized to survey how quickly defect are identified inside procedure of testing. This offers input to framework which is under test. The primary reason for prioritization will limit the test suits [8]. Experiment prioritization is utilized to categorize and actualize the experiments so as to save time and cost. Experiment prioritization is increasingly productive and broadly utilized by the analyzers.

Clustering is an information mining strategy of collection set of information objects into different gatherings or groups so questions inside the bunch have high likeness, yet are extremely not at all like items in alternate groups. Dissimilarities and similarities are surveyed dependent on the quality qualities portraying the items. Grouping calculations are utilized to sort out information, classify information, for information pressure and model development, for recognition of exceptions and so on.

Normal methodology for all bunching procedures is to discover group focus that will speak to each bunch. Bunch focus will speak to with information vector can advise which group this vector have a place with by estimating a likeness metric between information vector and all group focus and figuring out which group is closest or most comparative one [3]. Testing programming is an essential and testing action.

About portion of product generation improvement cost is spent on testing. The principle of programming test with grouping approach is to take out the whatever number mistakes as would be prudent to guarantee the programming meets the satisfactory dimension of a value. Relapse testing is a profoundly vital however tedious action [1]. A lot of work has been performed on formulating and assessing systems for choosing, limiting, and organizing relapse test cases [2]. Such strategies are fundamental, however tragically not adequate to help scale regression testing to huge, complex frameworks. To be sure, practically speaking, even with productive prioritization[4] or choice, various relapse test deviations may should be broke down to decide whether they are because of a relapse blame or essentially the impact of a change.. An issue that has been to a great extent overlooked up until now, yet which is profoundly imperative practically speaking, is the manner by which to adapt to the numerous errors (deviations) that can be seen when running relapse test cases on another form of a framework. Relapse testing is done when modifications are made to existing programming; the motivation behind relapse testing is to give certainty that the recently presented changes don't block the practices of the current, unaltered piece of the product. It is an intricate technique that is all the all the more difficult as a result of a portion of the ongoing patterns in programming improvement standards.

Manuscript published on 30 March 2019.
***Correspondence Author(s)
Gayam Rupa Sri, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram (A.P.), India.
Addepalli Leela Supriya, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram (A.P.), India.
Enukonda Raja, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram (A.P.), India.
Avinash Reddy, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram (A.P.), India.
Venkata Naresh Mandhala, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram (A.P.), India.

For instance, the segment based programming improvement technique will in general outcome being used of many discovery segments, frequently embraced from an outsider.

Any adjustment in the outsider segments may meddle with whatever remains of the product framework[14], yet it is difficult to perform relapse testing in light of the fact that the internals of the outsider segments are not known to their clients.

Programming frameworks and their surroundings change constantly. They are upgraded, revised, and ported to new stages. These progressions can influence a framework , hence programming specialists perform relapse testing to guarantee nature of the changed frameworks. Since relapse testing is in charge of a huge level of the expenses for programming upkeep and on the grounds that the supports costs frequently overwhelm add up to lifecycle costs [13], relapse testing is one of the biggest supporters of the general expense of programming.

Necessities Tests connecting goals get the groups of prerequisites, to use the necessity test cases detect ability lattice to gather test cases that are related with every prerequisite group. Figure 1 surveys the procedure. The two ovals on the left side speak to the groups of necessities. For example, group 1 contains prerequisites 1, 3, and 4. The figure speaks to the prerequisite tests recognizable grid. There are two cases (TC1 and TC2) related with prerequisite 1. The prerequisites tests mapping goals process gets the group of experiments (the ovals on the correct side of the figure).

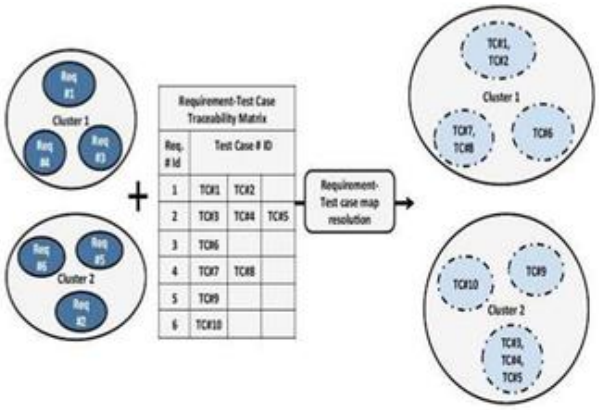


Fig. 1 Requirements Test Mapping Resolution.

For example, bunch 1 on the correct side contains five experiments (1, 2, 6, 7, and 8) that are related with necessities 1, 3, and 4.

II. LITERATURE SURVEY

J. Jones and M. Harrold [10] talked about the product testing is especially costly for designers of high-affirmation programming, for example, programming that is created for business airborne frameworks. One explanation behind this cost is the Federal Aviation Administration's necessity that test suites be altered condition/choice inclusion (MC/DC) satisfactory. In spite of its expense, there is proof that MC/DC is a successful confirmation system, and can reveal wellbeing shortcomings. As the product is adjusted and new experiments are added to the test suite, the test suite develops, and the expense of relapse testing increments.

To address the test-suite measure issue, analysts have explored the utilization of test-suite decrease calculations, which distinguish a diminished test suite that gives a similar inclusion of the product, as indicated by some standard, as the first test suite, and test-suite prioritization calculations, which recognize a requesting of the experiments in the test suite as per a few criteria or objectives.

G. Rothermel, R. Untch, C. Chu, and M. J. Harrold [9] showed the experiment prioritization systems plan test cases for execution in a request that endeavors to build their adequacy at meeting some execution objective. Different objectives are conceivable; one includes rate of blame identification, a proportion of how rapidly blames are distinguished inside the testing procedure.

The creators depicted a few systems for utilizing test execution data to organize test cases for relapse testing, including: 1) procedures that arrange test cases dependent on their aggregate inclusion of code parts; 2) strategies that arrange test cases dependent on their inclusion of code segments not recently secured; and 3) methods that arrange test cases dependent on their evaluated capacity to uncover blames in the code segments that they cover.

Sibson and Robin [5] stated that by using single link algorithm we can reduce the dissimilarities between the clusters and the resultant is in the form of dendogram

Defays [6] provided a algorithm for complete linkage and it is same as SLINK algorithm but it reduces the computational cost.

G. Rothermel and M. J. Harrold[11] proposed a relapse testing is a fundamental yet costly support movement went for demonstrating that code has not been antagonistically influenced by changes. Relapse test choice procedures reuse tests from a current test suite to test an altered program. Numerous relapse test choice strategies have been proposed, be that as it may, it is hard to think about and assess these systems since they have diverse objectives. This paper traces the issues significant to relapse test choice procedures, and utilizations these issues as the reason for a structure inside which to assess the methods.

Marre, M., & Bertolino[12] used spanning tree concept to find out the most covered entities and also used to calculate the total number of test cases in a test suite. To make coverage testing more efficient, we are interested in identifying the maximal objects in this ordering. Here subset represents set of completely covered paths so by using this we can reduce cost and improve product quality

Mirarab[7] proposed a Bayesian network a type of probablisitic model that uses feedback mechanism with every feedback we are adding appropriate test case to it

Xue, X., Pang, Y. & Namin[15] predicting the software components increasingly not mistaken shortcoming utilizing machine learning estimation has been finished.Here by using Feature selection technique we are ranking the GUI events we respect to the suspicions of the component

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

III. METHODOLOGY

A. Agglomerative Clustering

We propose a quite distinctive approach and prompt a method that considers all data points as examples. Our main effort is to improve the quality within less time by prioritizing the test cases. To achieve this, we used innovative Hierarchal based clustering algorithm for test case prioritization.

Steps:-

- Obtaining data set of test cases from test suites.
- Remove outlier from test cases.
- To improve its efficiency by using density information, apply Hierarchal algorithm.
- In this Hierarchal clustering algorithm we are using single link distance.Where each test case has some methods or values in it.
- By using Euclidean distance between the test cases we are forming the similar test cases into a cluster.
- We will be choosing less distance test cases and going to merge them.
- Compare the code coverage of the sublist with the entirely taken test cases.
- And, at last result comparison.

B. Hierarchical clustering-based test case classification (HBTC)

Require: *T*
Ensure: *TE*, *TN*

1: C = {}
2: **for** each *tci* in *T* do
3: Build a cluster *ci* for *tci*
4: C = C ∨ {*ci*}
5: **end for**
6: **while** there are more than two clusters do
7: Find the closest pair of clusters
8: Merge the pair into one cluster
9: Remove the pair of clusters from *C*
10: Add the new cluster into *C*
11: **end while**
12: *ci* = the cluster which contains the previous failing test cases
13: *TE=ci*

IV. RESULTS

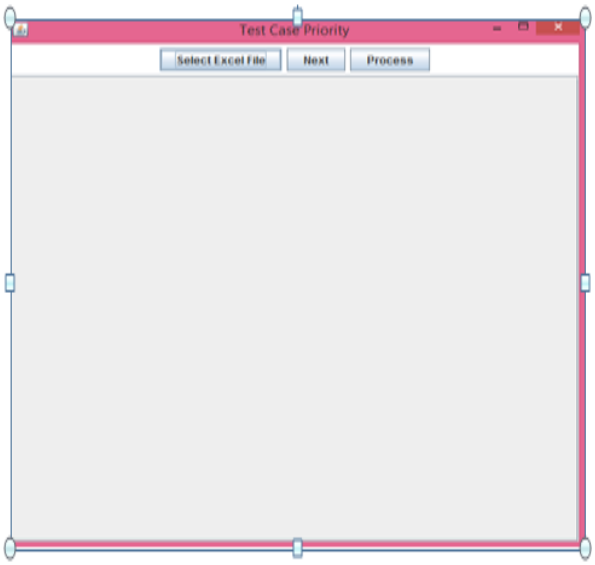


Fig. 2 A GUI is displayed where we can select an excel file.

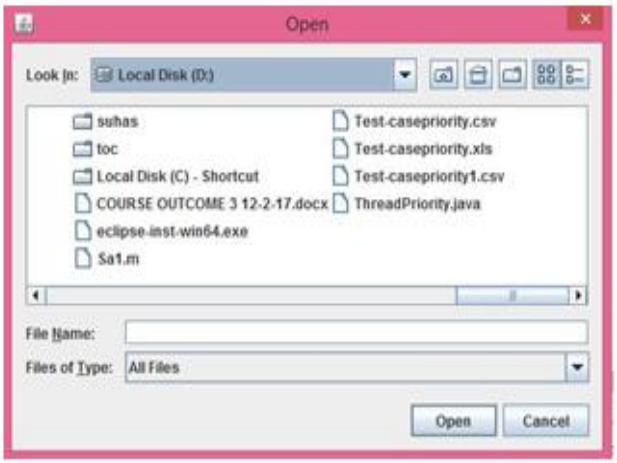
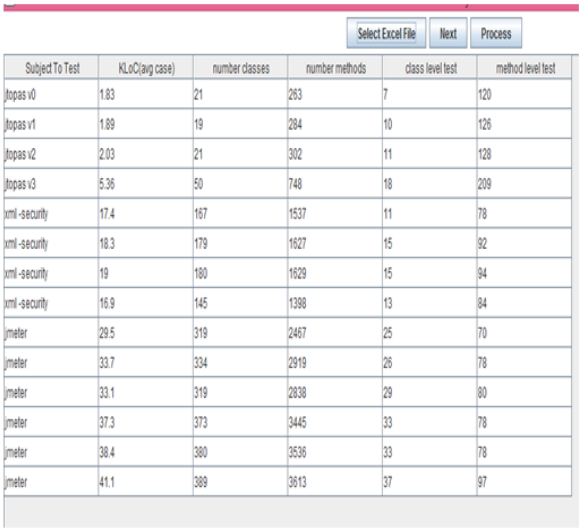


Fig. 3 A GUI is displayed when we click on select Excel file where we can select an Excel file (Xls).



Subject To Test	KLoC(avg case)	number classes	number methods	class level test	method level test
jtopas v0	1.83	21	263	7	120
jtopas v1	1.89	19	284	10	126
jtopas v2	2.03	21	302	11	128
jtopas v3	5.36	50	748	18	209
xml-security	17.4	167	1537	11	78
xml-security	18.3	179	1627	15	92
xml-security	19	180	1629	15	94
xml-security	16.9	145	1398	13	84
jmeter	29.5	319	2467	25	70
jmeter	33.7	334	2919	26	78
jmeter	33.1	319	2838	29	80
jmeter	37.3	373	3445	33	78
jmeter	38.4	380	3536	33	78
jmeter	41.1	389	3613	37	97

Fig. 4 A Table is displayed upon selecting an excel file.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

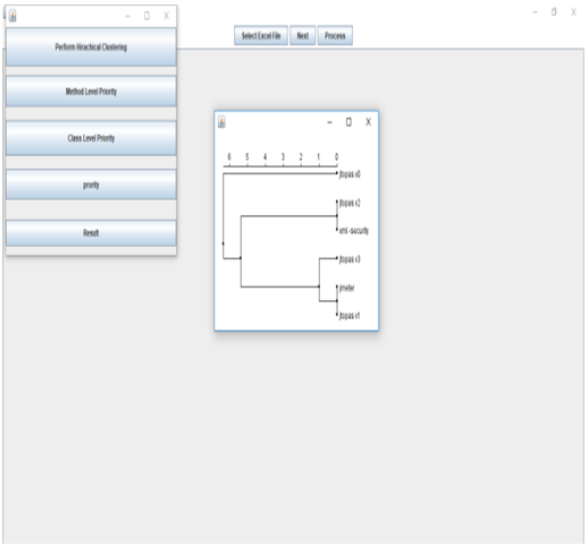


Fig. 5 A Hierarchical Cluster diagram is displayed.

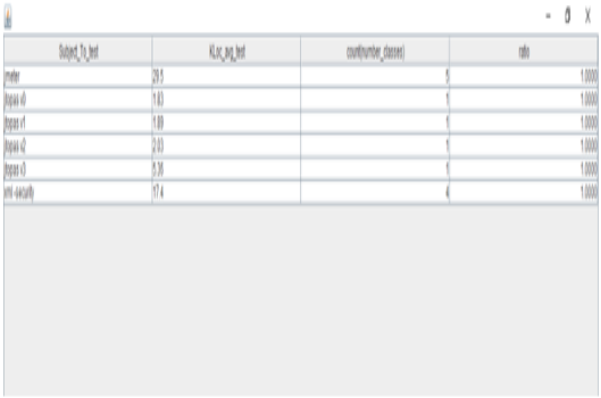


Fig. 6 Display's the iteration of Subjects being tested

V. CONCLUSION

To improve the quality of a product and to minimize the cost required for regression testing we have prioritized the test cases where all the test data is selected and similar data is merged into a cluster based upon the shortest distance between the test data.

REFERENCES

1. Pressman, Roger S. "Software engineering: a practitioner's approach." Aufl., New York (2001).
2. Pang, Yulei, Xiaozhen Xue, and Akbar Siami Namin. "Identifying effective test cases through k-means clustering for enhancing regression testing." In Machine Learning and Applications (ICMLA), 2013 12th International Conference on, vol. 2, pp. 78-83. IEEE, 2013.
3. Anderberg, Michael R. Cluster analysis for applications. No. OAS-TR-73-9. Office of the Assistant for Study Support Kirtland AFB N MEX, 1973.
4. MacQueen, James. "Some methods for classification and analysis of multivariate observations." In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, no. 14, pp. 281-297. 1967.
5. Sibson, Robin. "SLINK: an optimally efficient algorithm for the single-link cluster method." The computer journal 16, no. 1 (1973): 30-34.
6. Defays, Daniel. "An efficient algorithm for a complete link method." The Computer Journal 20, no. 4 (1977): 364-366.
7. Mirarab, Siavash, and Ladan Tahvildari. "An empirical study on bayesian network-based approach for test case prioritization." In 2008 International Conference on Software Testing, Verification, and Validation, pp. 278-287. IEEE, 2008.
8. Elbaum, Sebastian, Alexey G. Malishevsky, and Gregg Rothermel. "Test case prioritization: A family of empirical studies." IEEE transactions on software engineering 28, no. 2 (2002): 159-182.
9. Rothermel, Gregg, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. "Prioritizing test cases for regression testing." IEEE Transactions on software engineering 27, no. 10 (2001): 929-948.
10. Harrold, Mary Jean, David Rosenblum, Gregg Rothermel, and Elaine Weyuker. "Empirical studies of a prediction model for regression test selection." (2001).
11. Rothermel, Gregg, and Mary Jean Harrold. "Empirical studies of a safe regression test selection technique." CSE Journal Articles (1998): 11.
12. Marré, Martina, and Antonia Bertolino. "Using spanning sets for coverage testing." IEEE Transactions on Software Engineering 29, no. 11 (2003): 974-984.
13. Do, Hyunsook, Sebastian Elbaum, and Gregg Rothermel. "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact." Empirical Software Engineering 10, no. 4 (2005): 405-435.
14. Olson, David L., and Dursun Delen. Advanced data mining techniques. Springer Science & Business Media, 2008.
15. Xue, Xiaozhen, Yulei Pang, and Akbar Siami Namin. "Feature selections for effectively localizing faulty events in gui applications." In Machine Learning and Applications (ICMLA), 2014 13th International Conference on, pp. 306-311. IEEE, 2014.
16. Xue, Xiaozhen, Yulei Pang, and Akbar Siami Namin. "Trimming test suites with coincidentally correct test cases for enhancing fault localizations." In Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual, pp. 239-244. IEEE, 2014.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)