A Novel Density based K-means Clustering for Test Case Prioritization in Regression Testing

[1]Isha Luthra [2] Harsimranjeet Kaur
Chandigarh Engineering College, India

Abstract **- In this paper, we work on improving the test case prioritization on the basis of clustering approach. A novel density based k-means clustering approach is used to make clusters of different test cases on the basis of statement coverage. Then, prim's algorithm is used to find out the minimum path between different test cases according to their coverage information. Test cases are select from every cluster; which have maximum coverage information. According to Prim's algorithm, we will find the tree of test cases; this technique reduces the test cases numbers. Only those test cases are selected which have maximum coverage information. It will reduce the effort, cost and time also.**

Keywords **- Test Case Prioritization, Density based Kmeans, Regression Testing**

I. INTRODUCTION

Software industry has come into existence into 4 eras, 50's –60's, mid 60's –mid 80's, and mid 80's-present. In 1822, Charles Babbage starts working on a prototype of his first difference engine, which is made to compute values of polynomial functions. Software testing began in 1950s. Until 1956 it was the debugging oriented period, where testing was often associated to debugging. Testing and debugging both were not clearly different from each other. In 1979, Glenford J. Myer was introduced debugging and testing was distinguished now, which was demonstration oriented period. From 1988, it was prevention oriented period in which software satisfies its specification, to detect and prevent faults. Software test techniques and black box testing was also introduced in 1980's by Boris Beizer. Black box may include functional testing, which is divided into different parts. Regression testing is one of the most populated testing types of functional testing. There is a lot of work done in this testing. In regression testing, reset all, test case selection, test suit minimization and test case prioritization are different techniques of regression testing, which play important role during software maintenance phase of SDLC. Research in regression testing is a wide range of topics. In today's era, it has become an interesting topic in the field of research.

Software Testing

Software testing may also define as "A process of executing a program with the goal of finding errors". Testing means that one inspects behaviour of a program on a finite set of test cases for which valued inputs always exist. Software testing is the process of verifying and validating that computer program or application/product meet the requirements that guided its design and development, works as expected and also satisfies the needs of stakeholder. Testing mainly consists of two V's that are verification and validation. Verification is also known as static testing. In static testing, software are examine manually and missing requirements, design defects, syntax errors etc. are different types of defects find in static testing by review, inspection and walk through. Static testing can start early in the life cycle (e.g. by verifying User Requirement) and find bugs before compilation. Now, the term validation is also known as dynamic testing.

II. LITERATURE REVIEW

Rothermel et al. [1] has presented new techniques for test Case Prioritization. This paper deals with the effectiveness of these techniques for increasing rate of fault detection- measuring how quickly faults are detecting during testing process. In this paper, authors provide professional persons with useful, cost-effective techniques for improving regression testing processes through prioritization of test cases.

Rothermel et al. [2] has presented several techniques to prioritize test cases for regression testing using test execution information. This paper based upon three techniques, these are: first, techniques that order test cases based on their total coverage of code components, techniques that order test cases based on their coverage of code components not previously covered, and last, techniques that order test cases based on their estimated ability to reveal faults in the code components that they cover. This paper shows the results with respect to code-coverage-based techniques, if code coverage techniques are used in testing, they can be leveraged for additional gains through prioritization.

S. Elbaum et al. [3] has explained several questions raised by previous work to improve the rate of fault detection: can prioritization techniques be effective when targeted at specific modified versions; what trade-offs exist between fine granularity and coarse granularity prioritization techniques; can the incorporation of measures of fault proneness into prioritization techniques improve their effectiveness? Based upon several new controlled experiments and case studies fine-granularity techniques typically outperformed coarse-granularity techniques to improve rate of fault detection, but only by a relatively small margin overall.

Arafeen et al. [5] has explained the requirements-based clustering approach that incorporates traditional code analysis information can improve the effectiveness of test case prioritization techniques. The author performed an empirical study using two Java

programs with multiple versions and requirements documents. This paper concluded that by grouping test cases associated with a similar or related set of requirements, we can manage regression testing processes more effectively. The software which is independently developed and its operation will provide better testing and improved security.
Kayes et al. [6] has proposed a new metric for assessing rate of fault dependency detection and an algorithm to prioritize test cases. Analysis is done for prioritized and non-prioritized test cases with the help of proposed metric. This metric consider fault severity and test case execution time to be uniformed. This paper implemented a new regression test suite prioritization algorithm that prioritizes the test cases with the goal of maximizing the number of faults dependency detection that are likely to be found during the execution of the prioritized test suite. Prioritized test cases are more effective in detecting dependency among faults.

III. PROBLEM FORMULATION

Regression testing is one of the most important parts of software life cycle. This type of testing is essential for large organization. The main purpose of this testing is to finding side effects in new version of code or to determine whether a change in one part of the software affects other parts of the software. This testing will be conducted after any bug fixed or any functionality changed.
In this thesis work, to reduce this cost by prioritizing test cases and running the tests for the selective test cases as per the available time and manpower. There are a number of test cases available which can consume a lot of time and effort. A selective number of test cases needs to be selected which would be otherwise used for the same purpose. The priorities of the test cases need to be decided on the basis of several parameters. The parameters for the test case prioritization need to be chosen and a model needs to be developed which would set priority among the test cases. First of all a data set needs to be generated which would be utilized for our proposed algorithm testing. Then the dataset needs to be pre-processed for outlier removal and redundancy removal. Then a technique for clustering of the test cases needs to be developed which would be utilized for the above mentioned problem.

IV. METHODOLOGY

Steps of Density Based K- Means Algorithm:
Let X = {x_1, x_2, x_3... x_n} be the set of data points in Dataset D, Euclidean "ε" (eps).
K is the number of clusters to be found, minPts is a minimum number of neighbors required in ε neighborhood to form a cluster and N is a set of points in ε neighborhood.
Step 1: Start with arbitrary starting point (not visited).
Step 2: Find all the neighbor points of starting points using Euclidean distance "eps".
Step 3: If number of neighbors 'N' is greater than or equal to minPts, then starting point and its neighbors are added to cluster. Starting point is marked as visited; otherwise this point is marked as noise.
Step4: A new unvisited point is recall and processed for further make a part of cluster or noise.
Step 5: Repeat Step 2, until all points are marked as visited.
Step 6: We have 'm' cluster for each detected clusters; then find cluster centers, 'Cm' by taking the mean find the total number of points in each clusters.
Step 7: If 'm' cluster is greater than 'K' clusters, then join two or more cluster based on density and no. of points to find the new cluster center. Repeat Step 7, until achieving K clusters with 'Ck' centers.
Step 8: Otherwise if 'm' is greater than or equal to the number of clusters initially found by density based clustering algorithm 'l'; select a cluster based on density and number of points split it using k-means clustering algorithm. Repeat Step 8, until achieving k clusters with 'Ck' centers.
Step 9: Apply iteration of k-mean clustering with k and new 'Ck' centers as the initial parameters and label all the clusters with k labels.
Step 10: End

Flow chart of Density Based K-means Clustering Algorithm:

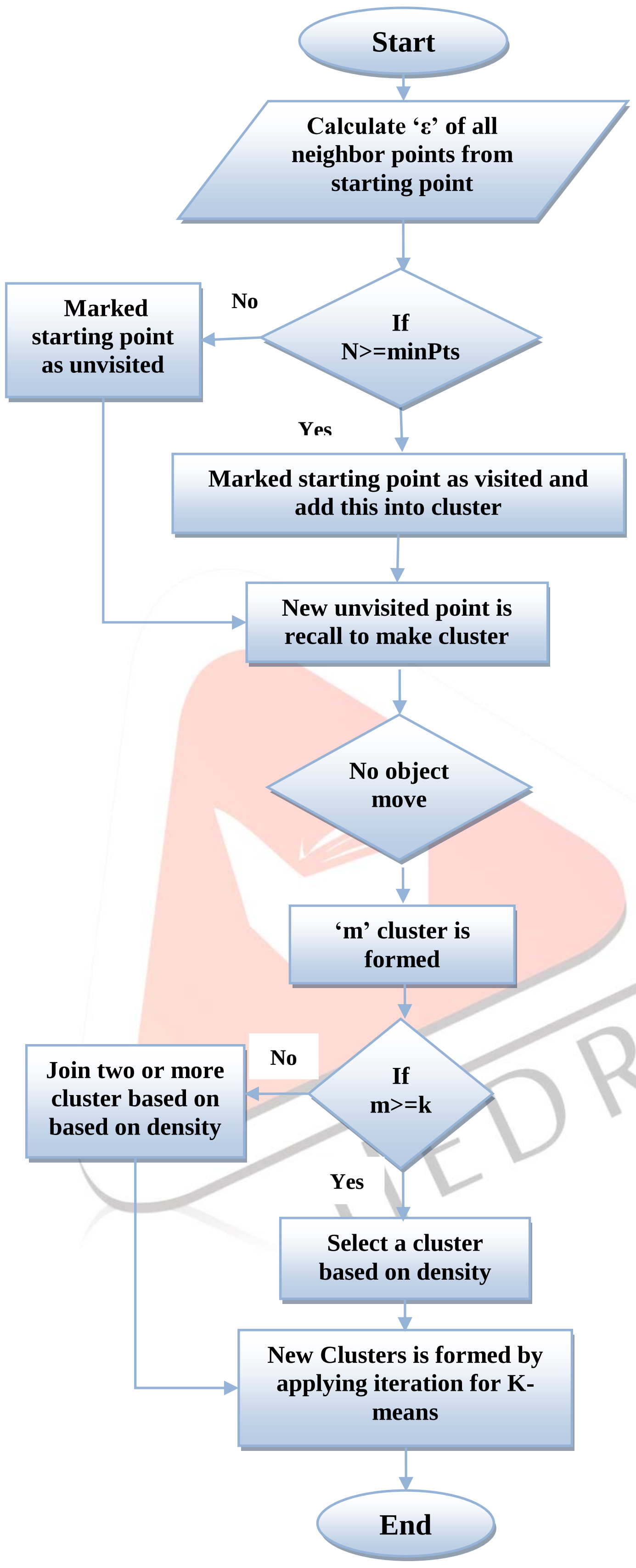
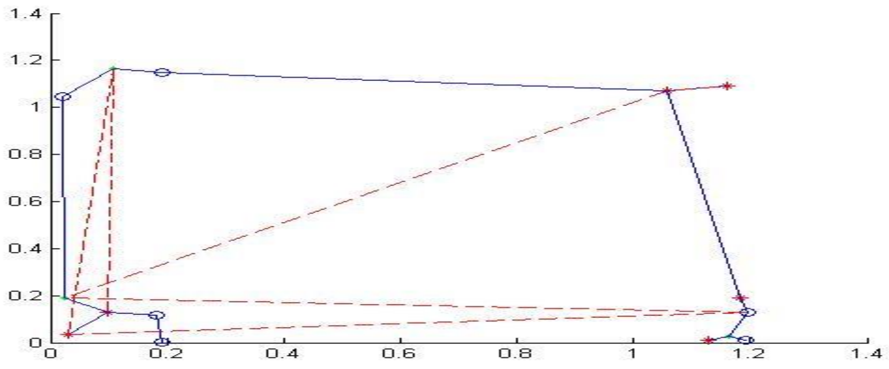

Start
Calculate 'ε' of all neighbor points from starting point
If N>=minPts
No
Marked starting point as unvisited
Yes
Marked starting point as visited and add this into cluster
New unvisited point is recall to make cluster
No object move
'm' cluster is formed
If m>=k
No
Join two or more cluster based on based on density
Yes
Select a cluster based on density
New Clusters is formed by applying iteration for K-means
End

Figure 4.1 Flow chart of Density Based K-means Algorithm

VI. RESULTS



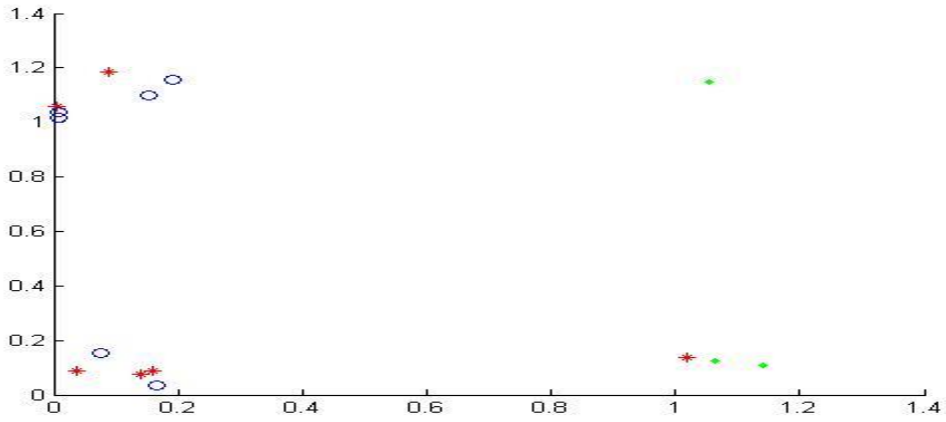
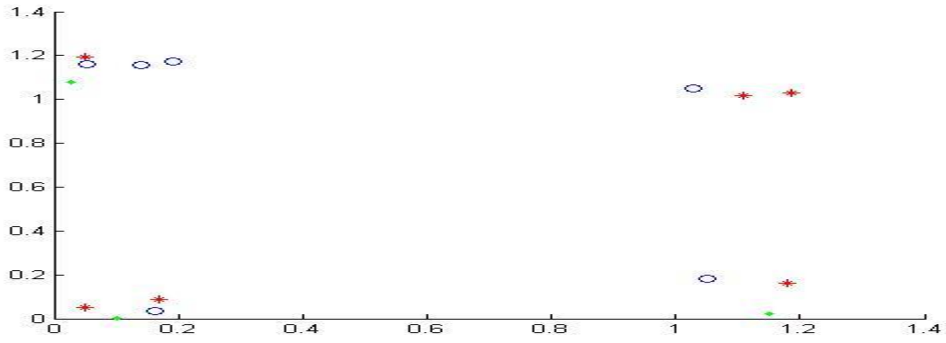
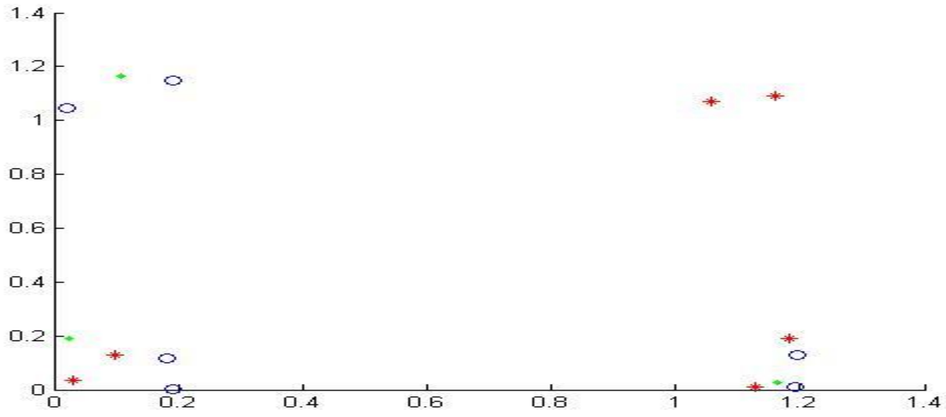
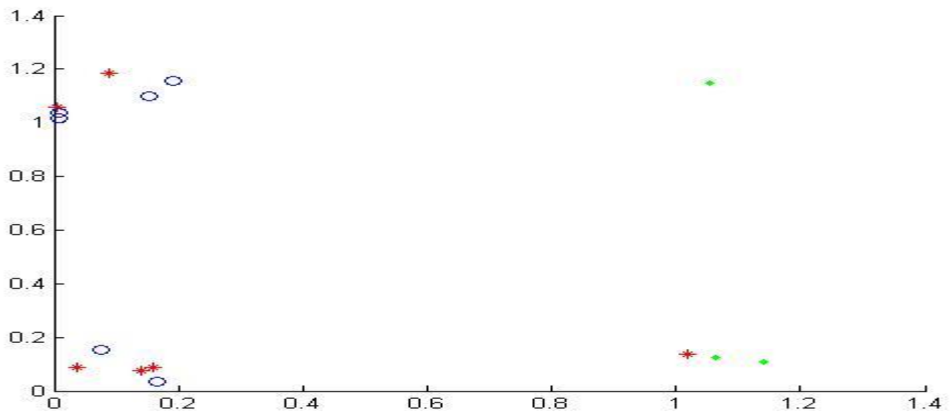
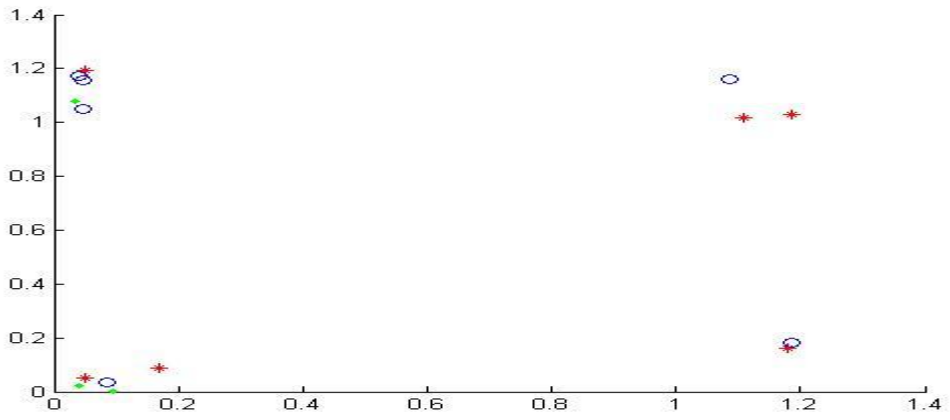


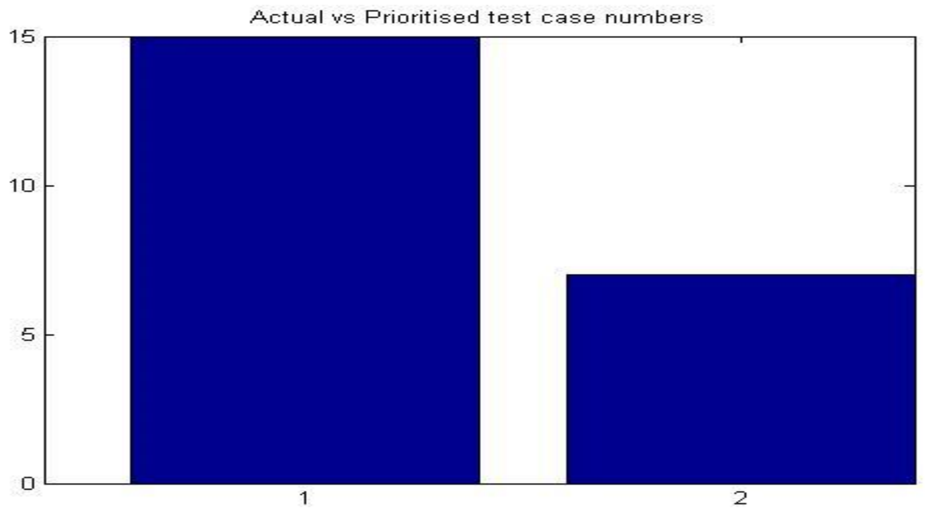
Fig 5.1

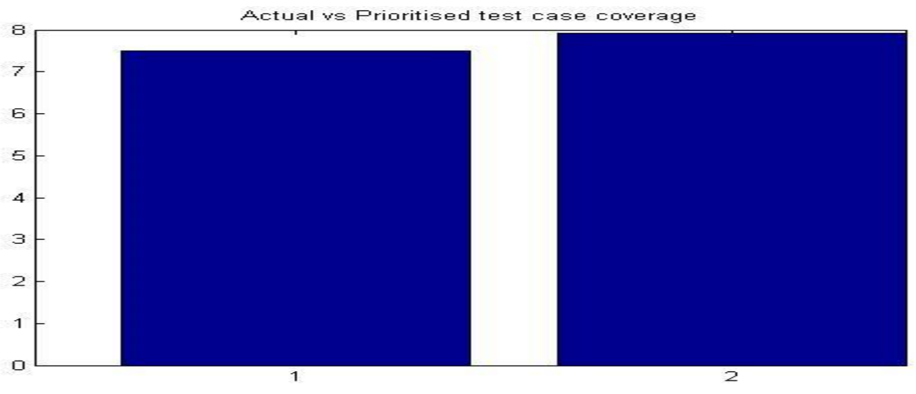









Actual vs Prioritised test case numbers


Actual vs Prioritised test case coverage

IV. CONCLUSION AND FUTURE SCOPE

A novel Density based Kmeans clustering has been applied for software testing. Test Case prioiritisation has been done using a hybrid algorithm of minimum spanning tree and density based kameans clustering. The results has been found to be quite better than their traditional counterparts. In future other algorithms can be implemented for the same and compared for performance.

REFERENCES

[1] Rothermel, Gregg, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. "Test case prioritization: An empirical study." In Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on, pp. 179-188. IEEE, 1999.

[2] Rothermel, Gregg, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. "Prioritizing test cases for regression testing." Software Engineering, IEEE Transactions on 27, no. 10 (2001): 929-948.

[3] Elbaum, Sebastian, Alexey G. Malishevsky, and Gregg Rothermel. "Test case prioritization: A family of empirical studies." Software Engineering, IEEE Transactions on 28, no. 2 (2002): 159-182.

[4] Carlson, Ryan, Hyunsook Do, and Anne Denton. "A clustering approach to improving test case prioritization: An industrial case study." In Software Maintenance (ICSM), 2011 27th IEEE International Conference on, pp. 382-391. IEEE, 2011.

[5] Arafeen, Md Junaid, and Hyunsook Do. "Test case prioritization using requirements-based clustering." In Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on, pp. 312-321. IEEE, 2013.

[6] Kayes, Md Imrul. "Test case prioritization for regression testing based on fault dependency." In Electronics Computer Technology (ICECT), 2011 3rd International Conference on, vol. 5, pp. 48-52. IEEE, 2011.

[7] Wu, Huayao, Changhai Nie, and Fei-Ching Kuo. "Test suite prioritization by switching cost." In Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on, pp. 133-142. IEEE, 2014.

[8] Kumar, Mukesh. "An optimized farthest first clustering algorithm." InEngineering (NUiCONE), 2013 Nirma University International Conference on, pp. 1-5. IEEE, 2013.

[9] Mei, Hong, et al. "A static approach to prioritizing junit test cases." *Software Engineering, IEEE Transactions on* 38.6 (2012): 1258-1275.

[10] Marijan, Dusica, Arnaud Gotlieb, and Sagar Sen. "Test case prioritization for continuous regression testing: An industrial case study." In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pp. 540-543. IEEE, 2013.

[11] Bryce, Renée C., Sreedevi Sampath, Jan B. Pedersen, and Schuyler Manchester. "Test suite prioritization by cost-based combinatorial interaction coverage." *International Journal of System Assurance Engineering and Management* 2, no. 2 (2011): 126-134.