A NOVEL APPROACH TO TEST SUITE REDUCTION USING DATA MINING

KARTHEEK MUTHYALA

Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, Rajasthan, India,
kartheek0274@gmail.com

RAJSHEKHAR NAIDU P

Computer Science and Information Systems, Birla Institute of Technology and Sciences Pilani, Rajasthan, India,
rajsekhar28@gmail.com

Abstract

Software testing is the most important and time consuming part of software development lifecycle. The time spent in testing is mainly concerned with generating the test cases and testing them. Our goal is to reduce the time spent in testing by reducing the number of test cases. For this we have incorporated data mining techniques to reduce the number of test cases. Data mining finds similar patterns in test cases which helped us in finding out redundancy incorporated by automatic generated test cases. We proposed a methodology based on clustering by which we can significantly reduce the test suite. The final test suite is tested for coverage which yielded good results.

***Keywords*:** Testing; data mining; test case; data item; test suite; weka; cluster; automated; redundancy.

1. INTRODUCTION

Software engineering[5] is a field of study related to designing, implementing and modifying software to built it affordable and maintainable. In this process Software testing[5] is one of the main processes of software engineering. Software testing[5] is one of the indispensable area of the software development life cycle. It is generally conducted on large scale to find the errors in our software and also to test the software for correct results. It aims at providing assurance to the client that our software works fine at any circumstances. This requires building test cases that exploits each and every possible route of the software. It is very tedious to explore each and every possible test case manually. Automated test case generation[2] has already been started where test cases are built automatically. This generates thousands of test cases through a simple program at a faster rate. But the problem with the above approach is if the software is built on thousands of lines of code, for execution of each test case it takes a lot of time to know the output and if the test suite is more the execution of the test suite[4,6,17] may take even days to complete. Test suite contains test cases that are machine generated. It contains redundant test cases too.

Our approach particularly deals with the above issue. Testing these redundant test cases even increases the time taken by testing phase. With increase in number of test cases this amount of time is significant. We have used data mining[10] approach to deal with the above issue. Data mining[10] is a novice research area where it is intended to extract patterns out of data that are not visible. In Data mining we have use clustering technique[10,8] which clusters similar data. The application of data mining techniques to our test suite significantly reduces the test suite. The coverage[12] either path or conditional by the reduced test suite yielded good results. The future directions of the given work is provided at the last section. This works involves dependency among test cases.

This paper is organized as follows. In section 2 we discuss the background work done in this area. Then we give introduction to concepts related to software testing covering what are the problems faced by it .Then we give an overview of data mining mainly focusing on data clustering by showing how K – Means algorithm[15] is used. In section 3 we show our proposed pickupcluster algorithm. Then we discuss our proposed methodology using above 2 algorithms in section 4.Then we implement our algorithm for a sample test suite in section 5.In section 6 we discuss the results. In section 7 we discuss the work to be done in future. Finally we conclude in section 8.

2. Background

There has been lot of work done in the area of test suit reduction but they are mostly in the area of heuristic algorithms,0-1 integral programming methods and so on[4,6,9,16.17].The idea of this paper emerged from the knowledge mining of test case system proposed by Lilly Ramesh[10] in which he talked about test case behavior, test suite redundancy among test cases and knowledge mining of the test suite. This sufficiently helps us identify the patterns in the test suite, which may be clearly extracted in many ways. In this paper we propose

one such method which extracts the test suite behavior by clustering. Before going to the methodology we proposed, in the next few sub sections we shall learn some basic understanding of concepts related to software testing and data mining.

2.1. *Software testing*

Testing is indispensible phase of software development life cycle. Debugging[5]is the search for cause of defects. Testing leads to uncovering problems which enhances further debugging. Software deployed without testing leads to unreliability. Hence testing the software to the full extend is a necessary task while building a software. Testing the software implies executing possible test cases. The extend of testing can be evaluated using several techniques like path coverage, conditional coverage, code coverage etc.

This phase of Software Development Life Cycle[5] is the most expensive phase. It requires lot of time and effort. Hence optimization of test cases is a must. But first we need to see how a test case look like.

A test case[4,6] is a collection of different attributes of the software. Attributes are the inputs to the software. So a test case can be compared to a touple in a database table. It has ID, attribute1,attribute 2, ……..attribute n. A good test case is one that is able to find faults with the software. Hence the output of test case is a pass/fail.

A test suite[4,6] is a collection of automated generated test cases for a particular software. But due to the process of automation redundancy can be initiated in the process of test data generation. Redundancy is the repetition of data, between one test case and the other. So optimization of test suite is important to achieve by which lot of time can be saved from executing redundant or unnecessary test cases. The behavioral patterns exhibited by the test suite helps us in this process of automation.

Due to the development in software processes, a lot of automation work is carried out in all of its activities. Like so automation is carried out in generating test cases also which enhanced the functionalities in testing phase . An automated test data generator is a program built with the feature of generating inputs to the software by considering business rules and input domain[2].

Inspite of a lot of care taken in generating test cases significant amount of data is replicated in different test cases. This replicated data isn't visible enough to capture unless and until we use sophisticated techniques like data mining.

2.2. *Data mining*

Data mining [10,11] is a semi automated process of finding patterns in the data. It is basically knowledge discovery in data. This knowledge discovered can be represented by a set of rules, equations relating different variables and other mechanisms of predicting outcomes.

The manual component of data mining[10] is the preprocessing phase where data is prepared acceptable by the algorithms and post processing phase involving discovering patterns to find out new ones that are useful. There are three main techniques in data mining classification, association rules and clustering[10]. Classification is a technique that classifies data into different classes by building models like decision trees. By using these models it predicts the behavior of future data. Association rules are the techniques used to find relationships or associations between different entities of an instance. With these associations we can predict the nature of one when the other changes.

Clustering[8] is a technique used in finding clusters of points in the given data. In other words clustering is grouping together similar points into a single cluster. This behavior of grouping can be found out by different metrics like distance, density and grid based approaches. Within a cluster all set of points in that cluster are found to have similar behavior. In order to ease this process of data clustering, in the next section we introduce a tool called weka[14] which helps us in filling the gap between the process of software testing and knowledge mining.

2.3. *Weka*

Weka[14] is an online freeware tool for data mining. It has implementations for different approaches to data mining. There are various clustering algorithms available like K-means, DBSCAN[10],etc. But for this paper we considered K-Means[15] which is a better way to restrict the number of clusters required depending on the value of K.

2.4. *k-Means algorithm*

One of the most widely used clustering algorithm is K-Means clustering[15]. This minimizes the mean squared Euclidean distance from each data point to its nearest centre.

Here we have a good control upon the number of clusters produced. So while reducing the test suit depending upon the number of test cases we wanted we can fix the value of k. Given a database

$$\mathbf{D = \{t_1, t_2, \ldots, t_n\}} \quad (1)$$

tuples and an integer value k , the k-Means algorithm defines a mapping

$$\mathbf{F : D \rightarrow \{1, 2 \ldots k\}} \quad (2)$$

[1].A cluster k_i contains all the tuples that are mapped to it.

Step1: Randomly pick k points as centroids of k clusters.
Step2:

- For each point assign the point to the nearest cluster.
- Re compute the cluster centroids.
- Repeat **Step2**(until there is no change in clusters between consecutive iterations).

With this idea of what k-Means do now we are going to discuss certain facts with respect to cluster behavior.

- The idea of clustering is to group data items having high similarity and to separate from dissimilar data items.
- The quality of a cluster is defined as high intra cluster similarity and low inter cluster similarity.

Having clustered our data, we now need some mechanism to choose test cases from each cluster. In the next section we used an algorithm called Pickupcluster that does the selection work.

3. Pickupcluster algorithm

k-Means algorithms clusters data items(test cases). That is the test cases in the same cluster have the same behavior. It would be redundant if we test different test cases from the same cluster because they would exhibit the same results. So in order to reduce this redundancy we need a selective approach of choosing test cases. This algorithm proposes a method to choose tuple randomly from a cluster.

3.1. *Algorithm:*

Input: Clustered data points from k-MEANS.
Output: Single data point from each Cluster.

Step:

- For each cluster (1,….k), where each C_i contains all the tuples ($t_{i1}, t_{i2}, \ldots t_{in}$) that are mapped to it from k-Means.
- Pick one tuple randomly from ($t_{i1}, t_{i2}, \ldots t_{in}$).
- Add this tuple with the label C_i to file output.

At the end of execution of this algorithm we have one tuple from each cluster. In the next section we are going to see a detailed overview of our methodology which is based on the background provided on this section.

4. Proposed methodology

With this idea of software testing and data mining, the methodology we are going to propose in this section deals with the efficient reduction of test suite.

4.1. *Algorithm proposed*

(1) Generate test cases for the software using automation. Save them to a text file.
(2) Convert the file into attribute related file format(arff) according to the specifications in Weka.
(3) Load the converted arff file into Weka.
(4) Apply k-means algorithm to the above loaded data (k signifies how many clusters or in other words how many test cases we are looking for).
(5) Save the cluster assignments in a arff file.
(6) Take the clustered arff file and convert it into a simple text file by following the opposite process followed in step2
(7) Load the above text file into pickupCluster function .Execute the algorithm and save the output to a text file.
(8) Use this text file to test for coverage of the software.
(9) Repeat from step4 until acceptable coverage is achieved.

The complexity of K-Means algorithm is O(KNM) where K is the number of clusters,N is the number of test cases and M is the number of iterations. The complexity of Pickupcluster algorithm is O(K) where K is same as above. Therefore the overall complexity is O(KNM+K).The above methodology is deployed in different unit cases and tested for coverage and they showed good results. The next section deals with one of such a case.

5. Sample test case

Suppose we have a software which takes only two inputs x and y . It expects values of x and y between 0 and 100 say. And different conditions are imposed in the software program for the values of x and y. For example

if(x>63) && if(y<73) (3)

An automated test case generator[2] generates randomly different values for x and y. Suppose it generates 1000 test cases, and on testing these test cases on the software exhibited 30 different behaviors(i.e the behavior may be due to the conditions, there exists 30 different conditions). We can think of having 30 different paths. All the test cases followed one of these 30 different paths.

Optimally speaking a limited number of test cases each having different behavior may be sufficient enough to test for these 30 different paths. So applying the methodology proposed in the previous section , clustering these 1000 test cases. Let us choose k value for the algorithm depending as 30 as 30 different paths exists. Clustering these 1000 test cases into 30 clusters of different behavior. Now picking one test case from each of these 30 clusters have different behavior. So the output of the above algorithm in this case gives 30 different test cases. Test the above algorithm for coverage[12,13]. If all the paths are not covered then repeat the process by taking some higher value of k and until good coverage is achieved.

By this approach we reduced the time wasted in testing unnecessary test cases. This case is limited to only 2 variables, what if our program has more than 10 inputs. In such cases this approach reduces our effort further. The step wise process of the above sample case is depicted in the following flow diagram.

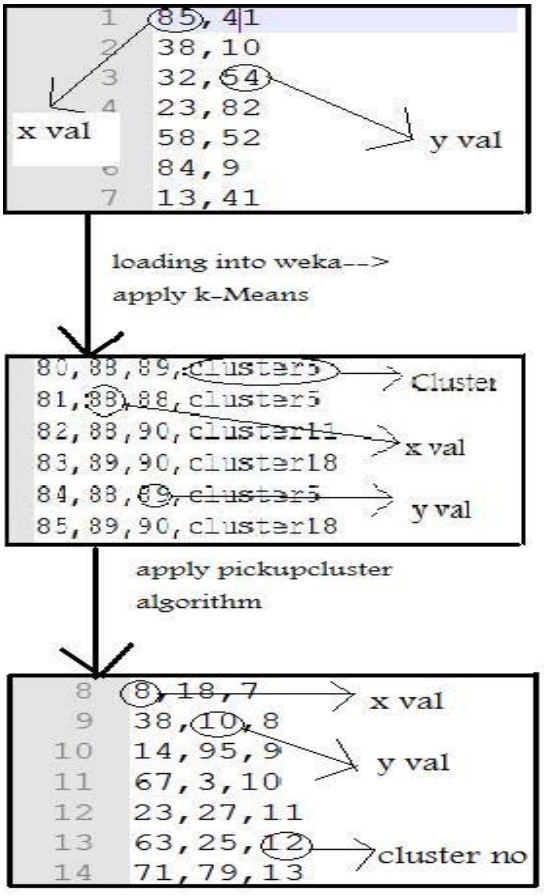

Fig 1. Showing the 3 stages of the proposed methodology
Stage1: Generation of random test cases
Stage2: Clustering of test cases
Stage3: Final set of test cases

6. Results

The methodology was further tested on test suite that contains many variable. The program containing several conditional checks and coverage of the conditions is tested for different values of k. As the k value increases the number of test cases to be tested increases and also there is improvement in coverage. Coverage[12,13] is tested by no of paths explored by my test cases with predicted output divided by total number of paths in the program. The following graph shows k vs coverage for the above test case.

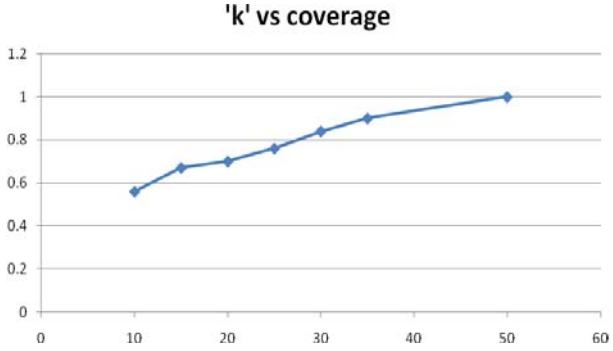

Fig 2.Graph showing k(X-axis) vs Coverage(Y-axis).

The average running time of the testing now is generation of the test cases + cluster generation+ pickupcluster+testing of the software with the new test suite. The running time was found to be far less than the actual execution of all the generated test cases. This optimization outperforms normal testing when the program to be tested is very huge.

7. Future work

The future work of the above approach needs to include dependency among test cases. If we some business rules that test case C should be executed only before test case B and test case A should be executed only before test case B and so on.

The order of execution is A→B→C

The above approach couldn't handle the situation directly, as my pickupCluster algorithm may chose only one among A,B,C if they exist in one cluster or it may or may two of them or it may not pick any. Any one of the above scenario's may occur. So in such cases we may need our algorithm to efficiently tackle the ordering of test cases execution. A small change in the pickupCluster can incorporate these changes. But it is restricted to a particular business rule. So our future work deals with generalization of these dependencies with the above approach which works fine with any test suite.

8. Conclusions

We discussed different problems related to software testing and the test suite size. We discussed about the automated test case generators and the redundancies they incorporate. We took help of data mining and proposed a new technique in software testing which reduced the size of the test suite significantly. We tested our reduced test suite for coverage and shown results yielded by the sample test case. We proposed future works of these lines of research. So deployment of testing techniques with this methodology significantly reduced time and effort spent in executing thousands of test cases.

Acknowledgments

We would like to acknowledge Mr. Praveen Ranjan Srivatsava (Lecturer Computer Science and Information Systems,BITS Pilani) for his constant help during our project.

References

[1] Abraham Silberschatz, Henry F.Korth and S.Sudarshan, "Database system concepts", Internation Edition ,2006,pp 739-741.

[2] Ajitha Ranjan."Automated Requirements-Based test case Generation". *Communications of ACM*,2006

[3] Antonia Bertolino."Software testing Research: Achievements, challenges and dreams *"Future of Software Engineering*,2007.

[4] T. Y. Chen and M. F. Lau. A new heuristic for test suite reduction. *Information and Software Technology*,40(5):347-354, 1998.

[5] David Alex Lamb, "Software Engineering, planning for change," Prentice Hall, Englewood Cliffs, NJ 07632, pp. 109–112, 1988.

[6] M. J. Harrold, R. Gupta, and M. L. Soffa. A methodology for controlling the size of a test suit. *ACM Trans. on Soft. Eng. and Meth*., 2(3):270-285,1993.

[7] A. E. Hassan, A. Mockus, R. C. Holt, and P. M. Johnson.Guest editor's introduction: Special issue on mining software repositories. IEEE Trans. Softw. Eng., 31(6):426–428, 2005.

[8] A. K. Jain, M. N. Murty, and P. J. Flynn.A Data clustering: review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[9] J. G. Lee and C. G. Chung. "An optimal representative set selection method". *Information and Software Technology*, 42(1):17-25, 2000.

[10] Lilly Ramesh, "Knowledge Mining of Test Case System," *International Journal on Computer Science and Engineering* Vol.2(1), 2009, 69-73.

[11] Mark Last and Menahem Friedman."The Data Mining approach to automated software testing.".*Communications of ACM*,2003.

[12] Martina marre and Antonia Bertolino, "using spanning sets for coverage testing". *IEEE transactions on software Engineering*, vol.29.

[13] Myra B Cohen and Matthew B Dwyer."Coverage and adequacy in software product line testing",*Communications of ACM*,2006.

[14] Remco R. Bouckaert, "Weka Manual 3-6-1", Software manual, June 4,2009, pp-11-14.

[15] Tapas Kanugo and David M Mount."A local search approximation algorithm for K-means clustering". *Communications of ACM*,2002.

[16] Yanping Chen and Robert L Probert."Regression test suite reduction using extended dependence analysis" *Communications of ACM*,2007.

[17] Zhenyu Chen and Baowen Xu."A novel approach for test suite reduction based on requirement relation contraction". *Communications of ACM*,2006.