

AN EFFICIENT APPROACH FOR TEST SUITE REDUCTION USING K-MEANS CLUSTERING

¹MOHAMMED AKOUR, ²IMAN AL JARRAH, ³AHMAD A. SAIFAN

^{1,2,3}The Faculty of Information Technology and Computer Sciences, Yarmouk University, Irbid, Jordan

ABSTRACT

Software testing is the primary approach that is used to test and evaluate software under development. The main goal of testing is to find defects before customers find them out. It is very costly. Therefore, reducing the cost of the test is a big challenge. This paper aims at reducing the cost of the test by eliminating the redundant test cases. Our methodology begins with generating the test cases randomly. The Procedural Language/Structured Query Language (PL/SQL) tool is used to generate test cases from the payroll system database functions. The SPSS software package is used to apply the K-means Clustering algorithm to reduce the test cases. The results reveal that the proposed approach significantly reduces the number of test cases from 776 to 240 while keeping the same coverage.

Keywords: *Software Vulnerabilities, Software Complexity, Fault prediction, relation, code complexity*

1. INTRODUCTION

Software Engineering [1] is a field of study that is related to designing, implementing and modifying software to make it affordable and maintainable. Thus, the software testing is an important step to improve quality at each phase in the system development life cycle. The software testing is the process of finding errors and meeting the business goals with a good quality by minimizing the number of test cases. In fact, it is very hard to generate every possible test case manually. This issue has motivated researchers to invent different automated generating tools of test cases [1]. This automation allows a faster rate to generate thousands of test cases from a very simple given program. Unfortunately, the problem of the cost for running each test case, and knowing the output still exists, especially when the software is large. Also, if the test suite is very large, the execution of redundant test cases may take several days to be completed [1, 2, 3, 4].

In this paper, we used K-means Clustering technique [1,2] in order to reduce the number of test cases. Consequently, the time and cost of executing them will be minimized. The main purpose of using data mining in reducing the number of test cases is its ability to extract patterns of test cases that are invisible [5].

Figure 1 illustrates the framework of reducing the size of test suite.

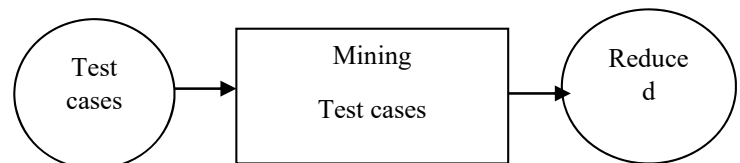


Figure 1: Framework of Test Suite Reduction [5]

We start our approach by building our data set that consists of the test cases of a given system. First, we automatically generate the test cases. Then, we apply the K-means Clustering algorithm. After that, the test cases that are far from the center of its cluster will be labeled as redundant test cases.

2. BACKGROUND

2.1 Software testing

Software testing is the process of detecting and finding errors. Detecting errors requires some

serials of inputs to test each function separately and other serials of inputs to make sure that those functions behave well with its all integrated functions. The software testing is a time consuming phase in the system development life cycle. The time which is spent on testing is mainly spent on the generation process of the test cases and testing them [1]. Anbarasu [4] defines software testing as “a process of ratifying the functionality of software”, which is one of the crucial processes that consumes a lot of time and then has a high cost. The time spent on testing is significantly related to the large number of test cases that are generated to be run. Unfortunately, those test cases are sometimes unreliable [4].

The main goal of testing is to detect errors as much as possible which makes the customer more confident in the quality of the software. So we need effective test cases that are able to detect the errors within the budget and scheduled limitation [5]. Testing activity is performed to make sure that changes and modifications during maintenance phase do not affect the existing behavior of the software badly [6]. It represents a way to deliver high quality software [6].

A test case is a representative value formulated by inputs to put the program under test. The actual result is compared with the expected result to verify reliability and consistency [5]. If they don't match, then there is a bug so some corrections have to be applied to the system. In this case, we say that the test case failed and a new test should be conducted. Notably, the skill of writing effective and efficient test cases can be achieved by some experience and deep study of the system requirements. In fact, writing those test cases manually consumes too much time and efforts so the automatic test case generator has been emerged.

A test suite contains thousands of test cases that can be automatically generated by tools for a simple program in few seconds, but the problem is the amount of time that is required to execute all of them. This test cases auto generation leads to the existence of redundant test cases [1]. Redundancy is the repeated data among different test cases. Those redundant or unnecessary test cases also cost too much. Obviously, the

problem is how to choose the effective test cases and exclude all the redundant test cases that waste lots of time and effort [5].

The test suite reduction aims at identifying and removing all the redundant test cases; therefore, we minimize the number of tests from the test suite [10]. The problem was introduced in 1993 by Harrold et al. [16] as follows:

Given:

- a. T is a test suite that consists of m test cases t_1, t_2, \dots, t_m .
- b. R is a set of n test requirements r_1, r_2, \dots, r_n , that must be satisfied to provide the desired coverage of the program,
- c. T_1, T_2, \dots, T_n are sub sets of T , that are associated with one of the r_i 's such that each test case t_j belonging to T_i can fulfill r_i .

Problem:

-Find the most effective set of test cases from the test suite, T by removing the redundant ones and at the same time cover almost all of the requirements [10].

Generally, trying to find the minimal subset of the original test suite, which satisfies all requirements of T and covers the same set of the test requirements, is an NP-complete hard problem (one of the most fundamental complexity classes). In order to solve this issue, several approaches are used to test suite reduction. In this work, clustering is used for this purpose.

2.2 Data mining

Data mining is a computational process that has the ability to extract patterns from large data sets that are indivisible [6].

There are several approaches in data mining that can be used to extract patterns from large data set, such as classification, association, and clustering [4].

Classification is a learning process that classifies data into various classes according to a classification model, such as decision trees,

Support Vector Machine, and Artificial Neural Networks. It finds the common features among a group of objects [1] [6].

Association rules are the techniques that are used for discovering interesting relations or associations among different entities of an instance. With these associations, we are able to extract the association patterns from large data set [1] [6].

Clustering is one of the data mining techniques that is used to divide a dataset into meaningful groups. The main task of clustering analysis is to group a set of objects as “close” as possible in the same group and as “far” as possible to those in other groups based on Similarity [1] [8][9]. Clustering can be achieved by various algorithms like distance, density and grid based approaches. Here, Clustering is one of the techniques which helps us for mining the test cases [1] [4]. A good clustering will lead to high quality clusters with minimum Intra-cluster distances and maximum Inter-cluster distances. Clustering is the most effective method of all the test case mining techniques to [8][10]. In order to facilitate this process of data clustering, we used a software package called SPSS.

3. LITERATURE REVIEW

3.1 Mining Techniques for Test case Reduction

Lilly Ramesh and G.V. Uma [9] proposed an approach that was able to automatically generate set of test cases from the requirement specification of the software. Firstly, they used weka tool in order to classify the requirements into functional and non-functional by using specific classification rules. Secondly, from non-functional requirements, state diagram was derived to specify the behavior of the system and to generate test cases. Finally, clustering algorithms were applied on the generated test cases in order to mine test cases and minimize the test suite size.

Another approach that was presented by Sarita Sharma and Anamika Sharma [12]. Firstly, they generated the test cases from the software requirement specification. Secondly, they reduced the number of test cases using data mining approach. In their approach, they

formally transformed the software requirement specification into UML state model. Then, they automatically generated set of test cases from the model. Finally, they used the clustering algorithm to minimize the number of test cases.

Akour et al. [22] Provide a systematic literature review that compare existing techniques for test case reduction. Their review summarized the cons and pros of the studied techniques and provided sort of guide lines for software testers.

Uma and Lilly[13] developed a mining approach that is used to get a good knowledge about the test cases and remove all the redundant ones. They developed a knowledge mining system that mines the test cases using attribute selection and k-mean clustering algorithm.

Chauhan et al. [8] proposed a methodology that is used to remove the redundant test cases by using density clustering technique. The density based approach grouped similar data objects based on connectivity and density functions. Firstly, the test cases were generated from Selenium tool, then Weka tool was used to apply DBSCAN clustering algorithm on them in order to remove redundant test cases with the help of suitable filter.

Saifan et al. [17] proposed new methodology to remove redundant test cases, they reduced the number of test cases based on two criteria average Cyclomatic Complexity and Code coverage, they ran one Java source code using Eclips SDK, the program was small of 885 LOC generated 504 test cases which also small number of test cases, then computing average Cyclomatic complexity metrics using CodePro Analytix tool, computed code coverage which consist of 4 types of coverage (instruction, method, branch, and line) using EclEmma tool, the values of previous 5 criteria were very high which means the program was complex, used SPSS to carry out clustering using K-mean algorithm, they chose only one single value for number of clusters K which was three based on the number of test cases, they called test cases which belong to the same cluster and far the same distance from cluster center " Redundant" test cases then they eliminated these redundant test cases and kept only one test case in the test

suite, the finding showed that 81.5% of test cases were redundant, obtained 93 unique test cases out of totally 504, also it gave good code coverage; the code coverage decreased by an acceptable different percentage for each type of coverage.

3.2 K-Means Clustering Technique

Subashini & JeyaMala [5] proposed new approach in clustering generate test cases using white box testing of 4 small Java source code and transferred it to control flow graph to obtain paths of the graph, the coverage criterion was Path coverage which represented number of paths covered by a test data, they took only direct paths to compute path coverage criteria and ignored the branching, they applied K-mean clustering algorithm and chose random value to K, the result showed that testing performed in efficient way and reduced number of test cases so program can be checked with any clustered of test cases not overall set of test cases of independent path, their methodology didn't consider dependences between sub systems. Muthyala & Naidu [1] used the same criteria to reduce the number of test cases among using two algorithms K-mean clustering algorithm and Pickupcluster algorithm to cluster test cases, they used Pickupcluster algorithm to generate random test cases then cluster these test cases using K-mean algorithm, they verified their methodology on simple program of 2 variables, the result yield good coverage and showed that as number of clusters increased then number of test cases will increased and Path coverage ratio (represented number of paths within a test case divided by overall program paths) will increased too, but in the experiment they didn't include dependencies among test cases also Pickupcluster algorithm didn't resolve the problem of test cases ordering, their methodology was limited to specific business rule.

Chai et al. [2] suggested a new approach that is used to reduce the number of test cases using k-mean clustering algorithm. They performed both white box testing and black box testing approaches, trying to minimize the size of test cases. In their methodology, they used pix software to automatically generate test cases for both the black and white boxing testing. For

black boxing testing, they were able to reduce the number of test cases from 648 to 486 and then to 324 while clustering maintain the same coverage.

Prasad et al. [18] proposed Hybrid optimization algorithms for pairwise test suit generation, used clustering technique among 2 algorithms K-mean and K-medoid algorithms to remove redundant test cases, they designed and implemented a system of 3 parts the number of parameters in the system, each parameter values, and the number of values for each parameter in a matrix as initial input of the input model, in the internal module they combined initial input as pairs and apply K-mean algorithm based on Euclidian distance measure and K-medoid algorithms to cluster test suite, and output module was matrix of an optimal test suite generated in optimized pair wise. To verify the efficiency of their approach they applied it on seven different systems, researchers didn't mention any properties of the systems or test cases used in the experiment, then compared the result with five algorithms namely All pairs, AETG, PICT, SPC, and PTSG_K, the finding showed that number of test cases generated by this technique were quite minimal, K-mean and K-medoid algorithms were competitive and enhanced the efficiency of testing. Their methodology wasn't automated, based on their methodology all parameters will be combined as pairs and tested at least once so it will not be suitable to large systems also it will increase time complexity.

K-mean clustering and GA used as hybrid technique to obtain reliable test cases that satisfies 100% of the path coverage. Khan & Amjad, [19] proposed algorithm to generated randomly test cases then cluster test cases using K-mean algorithm and applied the clustered test cases to control flow graph (CFG) and check path coverage for each cluster, selected clusters which its coverage less than 100%, dropped cluster which was the minimum path coverage percent and cluster that it paths exist in other clusters(redundant test cases) then applied GA operation to generate randomly new test cases and stopped until test cases achieved 100% of path coverage, they applied their algorithm on Java program of XY, they designed DFD and

obtain four paths then generated 47 randomly input values, chose number of clusters to be 4, researcher didn't mention reason to choose this value, each cluster contained a set of input then applied it on CFG of XY to obtain path coverage percent for each cluster they founded cluster1 coverage was 67% covered path 1 and 3, cluster2 was the same, cluster3 coverage was 33% covered only path 3, and cluster4 coverage was 67% covered path 2 and 3, they dropped cluster2 and cluster3 then applied GA on cluster1 and cluster4, because path coverage was less than 75% crossover operation was applied to generate new input data, GA stopped after 51 generation and achieved minimal set of test cases that satisfies 100% path coverage and represent optimal solution. Researchers achieved optimal solution, but it needed 51 generation, which was high number, improvement should be made to reduce the number of GA generation.

3.3 Hybrid Technique

Hybrid technique is a combination of two techniques used to enhance the finding, solve a problem of test cases reduction, and optimize test suite size. Fuzzy Clustering and Genetic Clustering are examples on Hybrid techniques used to optimize test case reduction.

Clustering and Fuzzy logic

Fuzzy Clustering also called Soft Clustering, in contrast of the ordinary clustering were each test case is belong to only one cluster and test cases which are more similar to each other will belong to the same cluster, in soft clustering a test case can belongs to more than one cluster, each test case is associated with group of different membership degrees which determine the strength of test case with a specific cluster, membership degrees range from zero to one, a test case at a cluster center is greater degree than boundary of cluster.

Fuzzy C-mean clustering algorithm

Fuzzy C-mean clustering algorithm (FCM) is one of the most important fuzzy clustering algorithm, C represent number of clusters, it divided a set of test cases to a set of c-fuzzy

clusters, each test case is associated to more than one fuzzy cluster based on membership degree, cluster center is a mean of all points, degree is inversely associated with the distance between cluster center and test cases.

Kumar & Bhatia [20] proposed Fuzzy Clustering to remove redundant test cases, they generated test cases manually and automatedly, initial guess to number of clusters C needed, reliable initial C was essential gab to this approach although the final number of C will be specified among the experiment, applied c-fuzzy algorithm then "Select Cluster algorithm" which applied c-mean clustering algorithm and selected a tuple from each cluster to minimize the number of test cases, saved the finding to a file and used it to examine two criteria Path and Condition coverage until acceptable value of them was obtained then reassign a value to C little greater than previous one and reapplied the algorithm, computed Standard deviation of all clusters centers, the point at which Standard deviation stopped minimizing and started rising considered as optimal number of C. researchers evaluated their methodology on program that determine natural root of quadratic formula its inputs 3 Int (a, b, and c) values range from zero to 100 and output one of these "not quadratic formula, real roots, imaginary roots, equal roots", they generated 500 test cases, chose Cyclomatic complexity as initial value to C which represented 7 different condition statements covered fully condition coverage, applied both algorithm and repeated it until all paths / conditions covered, they increased C from 7 to 8,9,10,and 11 respectively, the finding showed that Standard deviation for clusters centers were (0.957688, 0.221492, 0.03492, 1.354922, 1.606047) respectively, they chose optimal number C equal 9 which was minimal standard deviation that represented best code coverage and optimal solution to solve test case redundancy, researchers achieved single objective for test case reduction based only on the coverage and didn't consider reduction ratio in to account.

Geetha & JeyaMala [21] applied the same approach on another domain an Object-Oriented system, used new criteria a Reflection technique to remove redundant test to solve the problem of

objects ordering, it classified objects in to two types the first was dependent object assigned highest priority based on its ability to call large numbers of data members, and independent object assigned less priority. This classification happened while Runtime, so it needs expert developer in high level of object oriented language and problems happening with compiler. Their methodology consisted of 3 phases to classify objects if they were dependent or independent in Runtime, Clustering technique grouped data items of both dependent and independent objects based on its data types in to clusters, were data items of the same data type associated to the same cluster, the data types were (Int, Short, Long, Double, and String), thus selecting a test case from each cluster reduce number of test cases, then they applied If Then Rule based on Fuzzy logic, they chose only one data member from each cluster, applied If Then Rule to test only one data member from each cluster, numeric data types generated test case of negative and positive numbers, zero, and boundaries of negative and positive numbers, String data type generated three types of test cases a valid string, invalid string, and null. the values were pointed, and the overall values compared with the expected points if they were equal that meant all data types of all data members of all objects were tested with minimal number of test cases. They evaluated their system by student developed object oriented system. Dealing with object oriented system is very hard, complex, and time consuming so automated tool must be developed to save time and cost of testing process. Java Application implemented in Netbeans IDE, the rest of the experiment wasn't automated. Researchers minimized the number of test cases of objects without compromising software quality.

4. RESEARCH METHODOLOGY

The proposed methodology is depicted in Figure 2. It has the following steps:

1. Generates test cases using the PL/SQL random procedure.
2. Applies K-Means Clustering
3. Removes the test cases that are far from the centers clusters.

The following section presents a detailed discussion of the proposed steps.

4.1 Research Tools

Statistical Package for the Social Sciences (SPSS):

SPSS is a comprehensive system for analyzing data. It has the ability to create tables and graphs from several types of files. It is able to deal with large amounts of data and it can be implemented in all analysis covered by the text and more. It is commonly used in the Social Sciences, governmental offices, health researches, education researches, survey companies, marketing organizations, data miners, and others. There are different clustering algorithms that are implemented in it. In this paper, we only apply K-Means clustering.

Procedural Language/Structured Query Language (PL/SQL) Developer:

PL/SQL Developer 10.0 [15] is an "Integrated Development Environment that is specifically targeted at the development of stored program units for Oracle Databases. Over time we have seen more and more business logic and application logic move into the Oracle Server, so that PL/SQL programming has become a significant part of the total development process. PL/SQL Developer focuses on ease of use, code quality and productivity, key advantages during Oracle application development"[15]. In this work, PL/SQL is used to generate test cases.

Table 1. Data Set Sample

INPUT1	INPUT2	REQ	RESULT	COMPLEXITY
1852.574	2300	calc_pct	Pass	med
8471.429	1450	calc_rem	Pass	low
3887.464	6600	calc_pct	Pass	med
-472.705	600	calc_rem	Pass	low
201.93	300	calc_pct	Pass	med

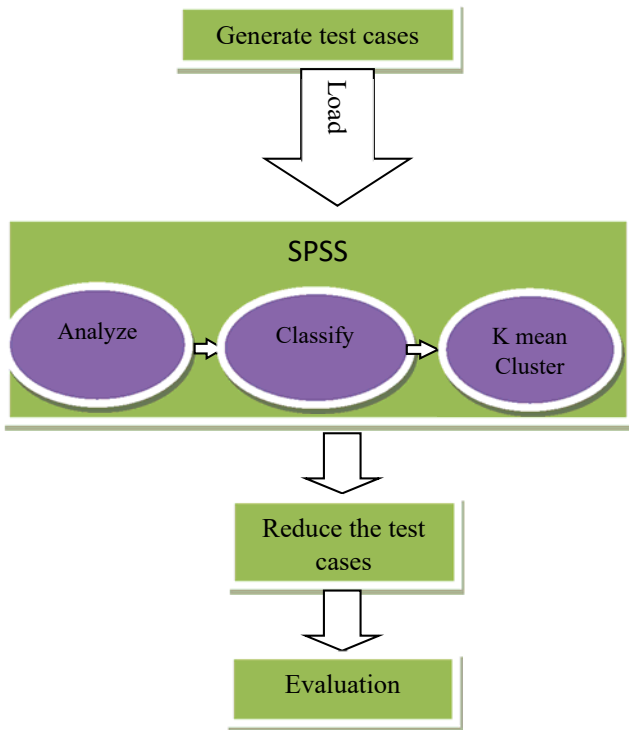


Figure 2: Steps of the proposed methodology

4.2 EXPERIMENTAL RESULTS

Our data set is a random input for payroll functions generated from a procedure that randomly tests those functions and then stores (1) the function name, (2) the function result (pass or fail), and (3) the level of complexity of the tested function and the input parameters for each function into the data base. The PL/SQL tool that is used to write the above procedure allows us to export the results into an excel file to store the redundant test cases and load it to the SPSS tool to apply the K-means Clustering algorithm. Table 1 is a sample of our data set which includes the names of attributes {INPUT1, INPUT2, REQUIREMENT, RESULT AND COMPLEXITY} and the provided test data (776 test cases).

The second step is applying K-means Clustering on our data set using the SPSS tool. Since the k-means algorithm clusters the data into different clusters based on distances, we have to decode the string results into digits, so calc_pct function is replaced with 123 and calc_rem function is

replaced with 345, pass and fail are replaced with 1 and 0, and finally low and med are replaced with 100 and 1000. Table 2 shows the previous sample dataset after replacements.

Table 2. Data Set of Table 1 After Replacements

INPUT1	INPUT2	REQ	RESULT	COMPLEX
1852.574	2300	123	1	1000
8471.429	1450	345	1	100
3887.464	6600	123	1	1000
-472.705	600	345	1	100
201.93	300	123	1	1000

In this step, we determine the number of cluster to be 4 clusters and select iterate option to enable the cluster center to change after each record of the dataset allocated. Firstly, the cluster center is allocated for each variable in the data set. Figure 3 illustrates that the initial center of cluster 1 for INPUT1 is 4800000.

	Cluster			
	1	2	3	4
INPUT1	4800000	14085257	0	8443714
INPUT2	7000000	23000000	0	14500000
REQUIREMENT	123	345	345	345
RESULT	1	1	0	1
COMPLEXITY	1000	100	100	100

Figure 3. Initial Cluster Centers

After finishing the initial cluster centers, every case is assigned to the closest cluster based on its distance from the cluster centers. In each iteration, every case allocated in clusters forces the clusters center to be recalculated. The iterations converge when the cluster centers remains unchanged. In our dataset, four iterations are sufficient as shown in Figure 4.

Iteration	Change in Cluster Centers			
	1	2	3	4
1	1806334.286	463.488	179661.954	463.4
2	642988.909	.000	8905.826	.0
3	482049.178	.000	8095.879	.0
4	.000	.000	.000	.0

a. Convergence achieved due to no or small change in cluster centers. The maximum absolute coordinate change for any center is .000. The current iteration is 4. The minimum distance between initial centers is 8338264.550.

Figure 4. Iteration History

Finally, Cluster Centers for each variable are calculated as illustrated in Figure 5. At the end of the iterations, all cases have been allocated into clusters depending on the last group of cluster centers. Notably, in Figure 5, the cluster center for REQUIREMENT, RESULT AND COMPLEXITY is the same for each cluster; therefore, it will not play a major role in classifying the tuple. The dominant variable is INPUT1 and INPUT2 where the cluster centers are obviously varied. Cluster 3 center for INPUT1 and INPUT2 has the smallest value, while cluster 2 has the largest value.

	Cluster			
	1	2	3	4
INPUT1	2844086	14085257	82495	8443714
INPUT2	4887208	23000000	140204	14500000
REQUIREMENT	234	234	234	234
RESULT	1	1	1	1
COMPLEXITY	550	550	550	550

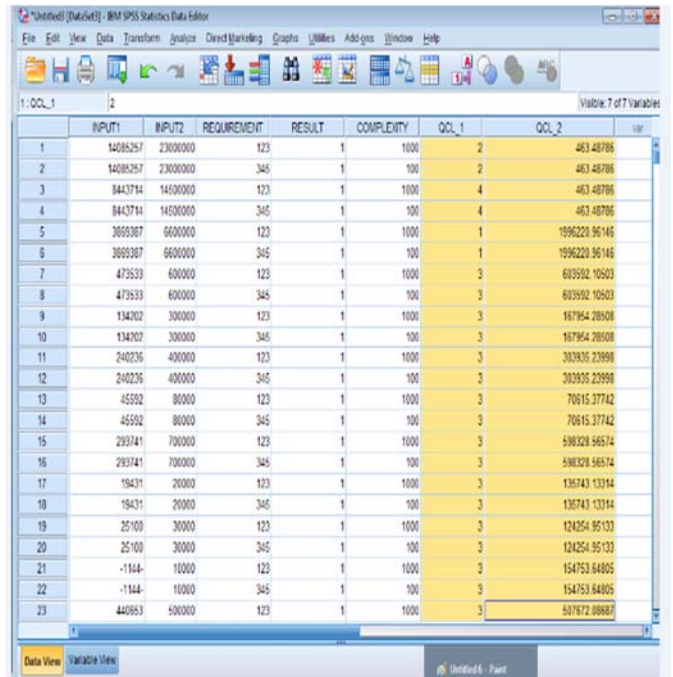
Figure 5. Final Cluster Centers

Figure 6 shows that the cases are almost lie in cluster 3. This means that the cases are almost near to cluster 3 center and far from the three other clusters center. It also shows that 12 cases lie in cluster 1, and 2 cases lie in both cluster2 and cluster4.

Cluster	1	12.000
	2	2.000
	3	760.000
	4	2.000
Valid		776.000
Missing		.000

Figure 6. Number of Cases in Each Cluster

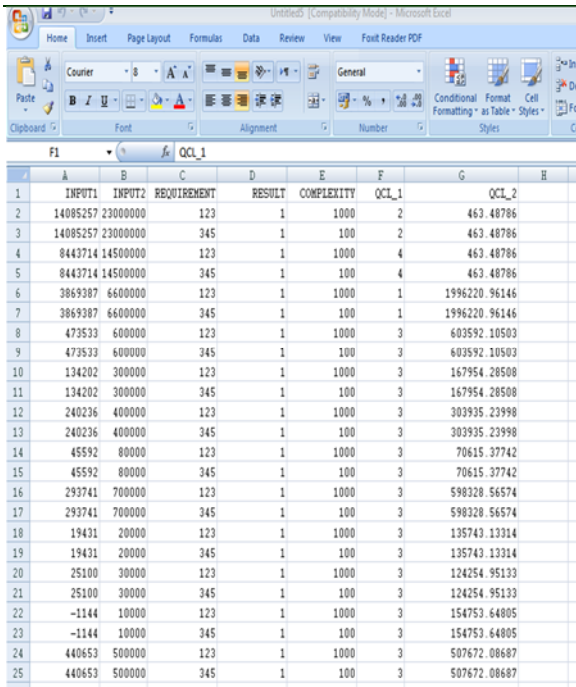
As illustrated in Figure 7, two new columns in the test case data sheet are shown. The first column indicates which cluster the record of the test case lies, while the second one indicates the distance between this record and its cluster center.



	INPUT1	INPUT2	REQUIREMENT	RESULT	COMPLEXITY	OCL_1	OCL_2	var
1	14085257	23000000	123	1	1000	2	463.48786	
2	14085257	23000000	345	1	100	2	463.48786	
3	8443714	14500000	123	1	1000	4	463.48786	
4	8443714	14500000	345	1	100	4	463.48786	
5	3893387	66000000	123	1	1000	1	1596220.96146	
6	3893387	66000000	345	1	100	1	1596220.96146	
7	473533	6000000	123	1	1000	3	603592.10503	
8	473533	6000000	345	1	100	3	603592.10503	
9	134202	3000000	123	1	1000	3	167954.28508	
10	134202	3000000	345	1	100	3	167954.28508	
11	240236	40000000	123	1	1000	3	303836.23998	
12	240236	40000000	345	1	100	3	303836.23998	
13	45592	800000	123	1	1000	3	70615.37742	
14	45592	800000	345	1	100	3	70615.37742	
15	293741	70000000	123	1	1000	3	598328.56574	
16	293741	70000000	345	1	100	3	598328.56574	
17	19431	20000000	123	1	1000	3	136743.13314	
18	19431	20000000	345	1	100	3	136743.13314	
19	25100	30000000	123	1	1000	3	124254.95133	
20	25100	30000000	345	1	100	3	124254.95133	
21	-1144	100000000	123	1	1000	3	154753.64805	
22	-1144	100000000	345	1	100	3	154753.64805	
23	440953	50000000	123	1	1000	3	507872.88887	

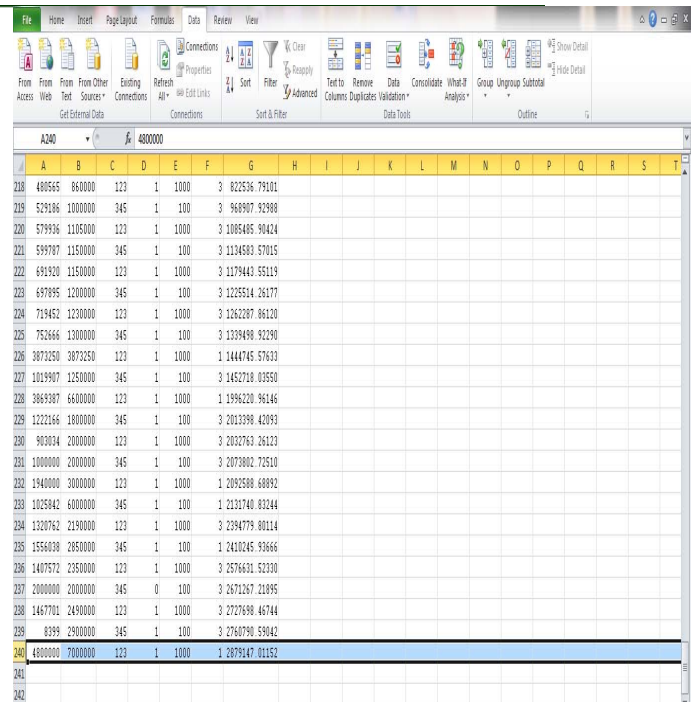
Figure 7: Cluster Number of Case & Distance from its Cluster Centers

The third step is to export this result to an excel file which will enable us to remove the redundant test cases based on the distance from the center. The exported excel sheet is shown in Figure 8.



	A	B	C	D	E	F	G	H
1	INPUT1	INPUT2	REQUIREMENT	RESULT	COMPLEXITY	OCL_1	OCL_2	
2	14085257	23000000	123	1	1000	2	463.48786	
3	14085257	23000000	345	1	100	2	463.48786	
4	8443714	14500000	123	1	1000	4	463.48786	
5	8443714	14500000	345	1	100	4	463.48786	
6	3869387	6600000	123	1	1000	1	1996220.96146	
7	3869387	6600000	345	1	100	1	1996220.96146	
8	473533	600000	123	1	1000	3	603592.10503	
9	473533	600000	345	1	100	3	603592.10503	
10	134202	300000	123	1	1000	3	167954.28508	
11	134202	300000	345	1	100	3	167954.28508	
12	240236	400000	123	1	1000	3	303935.23998	
13	240236	400000	345	1	100	3	303935.23998	
14	45592	80000	123	1	1000	3	70615.37742	
15	45592	80000	345	1	100	3	70615.37742	
16	293741	700000	123	1	1000	3	598328.56574	
17	293741	700000	345	1	100	3	598328.56574	
18	19431	20000	123	1	1000	3	135743.13314	
19	19431	20000	345	1	100	3	135743.13314	
20	25100	30000	123	1	1000	3	124254.95133	
21	25100	30000	345	1	100	3	124254.95133	
22	-1144	10000	123	1	1000	3	154753.64805	
23	-1144	10000	345	1	100	3	154753.64805	
24	440653	500000	123	1	1000	3	507672.08687	
25	440653	500000	345	1	100	3	507672.08687	

Figure 8. Final Result on Excel File (Before Reducing Test Cases)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
218	480565	860000	123	1	1000	3	822536.79181													
219	529106	1000000	345	1	100	3	968907.92988													
220	579936	1185000	123	1	1000	3	1085485.98424													
221	599787	1150000	345	1	100	3	1134583.57815													
222	691920	1150000	123	1	1000	3	1179443.55119													
223	697895	1200000	345	1	100	3	1225514.26177													
224	719462	1230000	123	1	1000	3	1262287.86120													
225	752666	1300000	345	1	100	3	1339498.92290													
226	3872250	3872250	123	1	1000	1	1444745.57633													
227	1015907	1250000	345	1	100	3	1452718.03550													
228	3869387	6600000	123	1	1000	1	1996220.96146													
229	1222156	1800000	345	1	100	3	2023398.42093													
230	983004	2000000	123	1	1000	3	2032763.26123													
231	1000000	2000000	345	1	100	3	2073802.71510													
232	1540000	3000000	123	1	1000	1	2092588.68892													
233	1025842	4000000	345	1	100	3	2131748.88244													
234	1230762	2190000	123	1	1000	3	2394779.88114													
235	1558038	2850000	345	1	100	1	2421045.93866													
236	1407572	2350000	123	1	1000	3	2576636.52330													
237	2000000	2000000	345	0	100	3	2671267.21895													
238	1467701	2490000	123	1	1000	3	2727698.46744													
239	8399	2900000	345	1	100	3	2768798.59842													
240	4800000	7000000	123	1	1000	1	2879147.01152													

Figure 9. Final Result on Excel File (After reducing Test Cases)

Finally, after removing the test cases which are far from the clusters centers, the number of test cases is reduced to 240 as shown in Figure 9.

To strengthen our work, it was evaluated through testing the coverage. We loaded the reduced test cases into oracle data base table and tested the two functions (calc_rem, calc_pct) by using INPUT1 and INPUT2 stored in the reduced file. As a result, we found that the reduced results almost cover all Pass and Fail paths of the both functions.

5. CONCLUSION

In this paper, we used K-Means Clustering to reduce the size of test cases significantly. We started by generating the test cases from the PL/SQL tool. Then, we applied the K-Means Clustering algorithm. After that, we removed the redundant test cases based on the distance from the test case cluster centers. The results showed that K-Means Clustering algorithm is a good test suite reduction technique, since it significantly helps us to locate the redundant test cases and reduce them from 776 to 240 test cases with the same coverage.

REFERENCES

[1] Muthyala, Kartheek, and R. Naidu. "A novel approach to test suite reduction using data mining." Indian Journal of Computer Science and Engineering 2.3 (2011): 500-505.

[2] Chantrapornchai, Chantana, Kanitsara Kinputtan, and Apaporn Santibowanwing. "Test Case Reduction Case Study for White Box Testing and Black Box Testing using Data Mining." International Journal of

- Software Engineering and Its Applications 8.6 (2014): 319-338.
- [3] Vasudevan, Shobha, et al. "Goldmine: Automatic assertion generation using data mining and static analysis." Proceedings of the conference on Design, automation and test in Europe. European Design and Automation Association, 2010.
- [4] Anbarasu, I., and S. Anitha Elavarasi. "A Survey on Test Case Generation and Extraction of Reliable Test Cases." International Journal of Computer Science & Applications 1.10 (2012).
- [5] Mrs.B.Subashini #1, Dr.D.JeyaMala *2 Department of Computer Applications, K.L.N. College of Engineering, Pottapalayam, Sivagangai, Tamil Nadu, india. Department of Computer Applications Thiagarajar College of Engineering, Madurai, Tamil Nadu, india.2009
- [6] Roongruangsuwan, S., & Daengdej, J. (2010). Test Case Reduction Methods by Using CBR. In International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS2010) (p. 75).
- [7] Raamesh, Lilly, and G. V. Uma. "An efficient reduction method for test cases." IJEST 2 (2010).
- [8] Chauhan, Rashi, Pooja Batra, and Sarika Chaudhary. "An Efficient Approach for Test Suite Reduction using Density based Clustering Technique." International Journal of Computer Applications 97.11 (2014).
- [9] Raamesh, Lilly, and G. V. Uma. "Reliable Mining of Automatically Generated Test Cases from Software Requirements Specification (SRS)." arXiv preprint arXiv:1002.1199 (2010).
- [10] von Ronne, Jeffery. "Test suite minimization: An empirical investigation." University Honors College Thesis, Oregon State University, Corvallis, USA (1999).
- [11] Han, Jiawei, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.
- [12] Sharma, Sarita, and Anamika Sharma. "Amalgamation of Automated Testing and Data Mining: A Novel Approach in Software Testing." IJCSI (2011).
- [13] Raamesh, Lilly, and G. V. Uma. "Knowledge Mining of Test Case System" International Journal on Computer Science and Engineering Vol.2(1), 2009, 69-73 .
- [14] Rashi Chauhan, Pooja Batra and Sarika Chaudhary. Article: An Efficient Approach for Test Suite Reduction using Density based Clustering Technique. International Journal of Computer Applications 97(11):1-4, July 2014.
- [15] Lee, Chung-Hong. "Mining spatio-temporal information on microblogging streams using a density-based online clustering method." Expert Systems with Applications 39.10 (2012): 9623-9641.
- [16] Harrold, M. Jean, Rajiv Gupta, and Mary Lou Soffa. "A methodology for controlling the size of a test suite." ACM Transactions on Software Engineering and Methodology (TOSEM) 2.3 (1993): 270-285.
- [17] Saifan, Ahmad & Alsukhni, Emad & Alawneh, Hanadi & Sbaih, Ayat. (2016). Test Case Reduction Using Data Mining Technique. International Journal of Software Innovation. 4. 56-70. 10.4018/IJSI.2016100104.
- [18] PRASAD, M.L., KEERTHI, M., SRIKAR, K. and DIVYA, V., 2017. Generating Optimized Pairwise Test Cases by using K-Means Algorithm. International Journal of Advanced Technology and Innovative Research, Vol.09, Issue.05.
- [19] Khan, R. and Amjad, M., (2016) "Automatic generation of test cases for data flow test paths using K-means clustering and generic algorithm", International Journal of Applied Engineering Research, 11(1), pp.473-478.
- [20] Kumar, G. and Bhatia, P.K., 2013. Software testing optimization through test suite reduction using fuzzy clustering. CSI transactions on ICT, 1(3), pp.253-260.
- [21] Geetha, B. and Jeya Mala, D. (2016) "Automatic Test Case Reduction in Object Oriented System Using Clustering and Fuzzy Logic", Asian Journal of Information Technology, 15(20), pp.4071-4076.
- [22] Mohammed Akour, Iyad Alazzam, Feras Hanandeh and Iman Akour, A systematic Literature Review to Classify Pre and Post Test Suite Reduction Techniques, International Journal Of Mathematics And Computers In Simulation Volume 9, 2015 Issn: 1998-0159 181