Incorporating Unsupervised Machine Learning Technique on Genetic Algorithm for Test Case Optimization

Maragathavalli Palanivel and Kanmani Selvadurai
Department of Information Technology, Pondicherry Engineering College, India

Abstract: *Search-based software testing uses random or directed search techniques to address problems. This paper discusses on test case selection and prioritization by combining genetic and clustering algorithms. Test cases have been generated using genetic algorithm and the prioritization is performed using group-wise clustering algorithm by assigning priorities to the generated test cases thereby reducing the size of a test suite. Test case selection is performed to select a suitable test case in order to their importance with respect to test goals. The objectives considered for criteria-based optimization are to optimize test suite with better condition coverage and to improve the fault detection capability and to minimize the execution time. Experimental results show that significant improvement when compared to the existing clustering technique in terms of condition coverage up to 93%, improved fault detection capability achieved upto 85.7% with minimal execution time of 4100ms.*

Keywords: *Test case selection and prioritization, group-wise clustering.*

Received August 14, 2014; accepted August 31, 2015

1. Introduction

A criteria-based optimization with multiple objectives concentrates on satisfying a number of objectives simultaneously. The multi-criteria optimization problem uses a set of solutions and each are satisfied the objectives at an acceptable level. The multiple attributes considered for optimization are fault detection capability, test suite size, cost and execution time [7, 13].

Genetic algorithm is a most commonly used test case generation technique in software development environment; an advanced heuristic search technique applied in the area of software testing. For a large search space and getting optimal set of solutions genetic algorithm is the best choice. In this, testing of software can be done with a single objective or with multiple objectives like minimizing the execution time and number of test cases simultaneously maximizing the coverage (i.e., the test requirements) would be considered [4].

Test case prioritization is a process of executing the beneficiary test cases while testing the software [6].Test case prioritization addresses the following objectives namely:

1. Software testers intend to increase the rate of fault detection.
2. Detecting the faults earlier in testing life cycle.
3. To enhance the condition coverage at a faster rate.

Test suite prioritization algorithm prioritizes the test cases with a goal of maximizing the fault detection ability that is likely to be found during the constrained execution [7, 10]. By using effective prioritization techniques, test cases can be reordered by the testers in order to obtain improved fault detection rate of the systems. A prioritized test is more effective; if execution needs to be stopped after a period of time, it can also be achieved using a random ordering [3].

Finally, optimization of genetic algorithm generated test cases and producing ordered test cases from a set of clusters have been taken as a new proposal. And also, fault detection by using reduced test set is considered as main testing criteria.

2. Existing Methods

The concept of multi-objective optimization used for test case generation with the help of genetic algorithms is taken from the earlier researchers [4, 13]. An Elitist Non-dominated sorting multi-objective Genetic Algorithm (ENGA)-based Automatic Test Pattern Generation (ATPG) for crosstalk induced delay faults has been introduced later. [13] Used an objective boundary for calculating fitness whereas [4] introduced multi-stage concept for automatic test data generation. [11] Suggested Unified Modelling Language (UML) diagrams for representing input to genetic algorithm; the idea of activity diagram of Control Flow Graph (CFG) representation is learnt from this paper. Machine learning techniques used for providing network security have been discussed in [1] and for object-oriented software fault prediction is reported by [6]; in particular for criteria-based test case prioritization, unsupervisored learning techniques are generally used. Clustering has been used for prioritization with the help of expert knowledge presented by [14, 15]. The genetic algorithm is applied for prioritization of test cases derived from UML diagrams by [11].

In [9] introduced the concept of optimizing the test suite size in object-oriented programming where methods are introduced for performing the optimization task [12]. A genetic algorithm-based community detection algorithm has been used to obtain the optimized package structures for object-oriented software. Multi-criteria optimization is used as test case selection in size-constrained regression testing and was reported by [7]. A cumulative mutation probability metric is used to determine the effectiveness of a new test case [3]. The concept of multi-criteria optimization for selection and ordering is learnt from papers [7, 9].

Test case prioritization techniques including random and selective prioritization are used by [10, 15]. Existing clustering techniques use pair-wise combination of test cases in order to reduce the execution time. If the cluster consists of more than two test cases, then the no of clusters will be reduced thereby execution time will also be reduced. The idea of group-wise combination is experimented using intra and inter-clustering mechanisms in [14, 15].

Prioritizing test cases using string distances has been attempted in [5] and the comparison of fault detection is done with random ordering; the results are more efficient in detecting the strongest mutants and have a better Average Percentage of Fault Detected (APFD). Early fault detection model using integrated and cost-effective test case prioritization is proposed [8] which increase the test suite's fault detection rate. APFD metric is used to measure the test suite's fault detection rate. The importance of early fault prediction is learnt from [8]. Test prioritization using hierarchical clustering applied to industrial case study has been found in [2].

From the literature, in most of the systems decision making is done using clustering. Multi-criteria optimization techniques suitable for different stages of testing are analysed and an attempt of genetic algorithm with clustering for object-oriented test case generation, selective prioritization with multiple objectives without using a separate selection mechanism has been taken as a proposal. In each stage, the target goal is to reduce the size of a test suite without affecting the fault detection capability.

3. Proposed Work

The main objective of the work is to reduce the test suite size; the number of clusters formed in the existing hierarchical clustering is tried to reduce the execution time. The entire system is shown in Figure 1 which has two main processes:

1. Test case generation using Genetic Algorithm (GA).
2. Test case prioritization using Group-Wise Clustering Algorithm (GWCA).

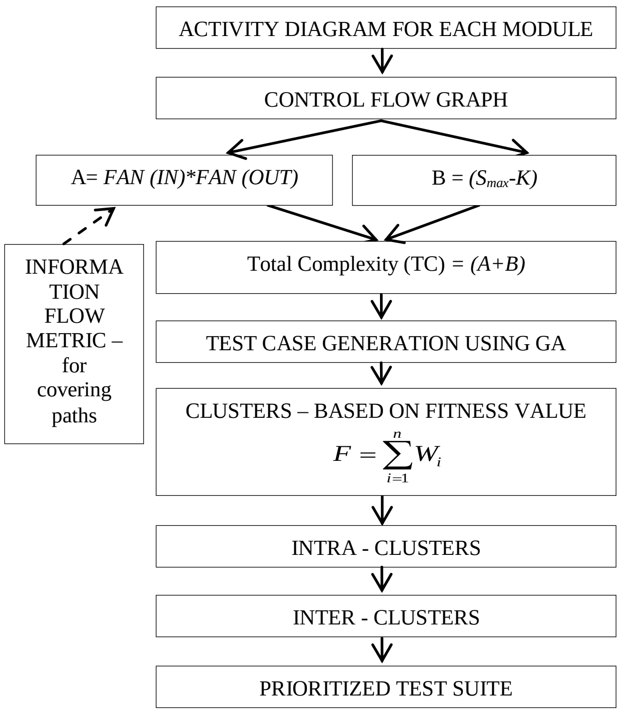

ACTIVITY DIAGRAM FOR EACH MODULE
CONTROL FLOW GRAPH
A= FAN (IN)*FAN (OUT)
B = (S_{max}-K)
INFORMATION FLOW METRIC – for covering paths
Total Complexity (TC) = (A+B)
TEST CASE GENERATION USING GA
CLUSTERS – BASED ON FITNESS VALUE
$F = \sum_{i=1}^{n} W_i$
INTRA - CLUSTERS
INTER - CLUSTERS
PRIORITIZED TEST SUITE

Figure 1. System design.

3.1. Test Case Generation using Genetic Algorithm

The input program is represented as activity diagram and then converted into the CFG of activity diagram. The nodes are numbered from initial to final state by numbers 1, 2, 3 etc; node weight is calculated by using the value of maximum stack height and number of nodes present in the stack. Maximum stack height value is equivalent to the number of nodes in Control Flow Graph except decision nodes. The total complexity of the node is calculated using the formula,

$$A = FAN(IN) \times FAN(OUT) \tag{1}$$

The nodes are pushed into the stack and the total complexity of the node is calculated (TC=A+B). The GA parameters used for test case generation including crossover, mutation and fitness range is given in Table 1.

Table 1. GA parameters.

GA Parameters		
Name of the Parameter	**Initial Value**	**Range**
Fitness Value	40	40-200
Crossover Probability, C_p	0.95	0.1-1.0
Mutation Probability, M_p	0.95	0.1-1.0
Stopping Criteria	55	≤1000 test cases

3.1.1. Algorithm for Test Case Generation using Activity Diagram

The steps involved in test case generation using Genetic Algorithm by taking activity diagram of Control Flow Graph as an input is given in existing algorithm [11].

Suppose the initial population is 0011, then the test case satisfies only last two conditions / decisions (true-1) and first two conditions value assigned 00 (false-0) which is not satisfied by the test case.

3.1.2. Operations used in Genetic Algorithm

The operations used for test case generation are mentioned in Figure 2; Selection of test case is performed using high-low fit method, crossover using two-point, and mutation using bit-inversion.

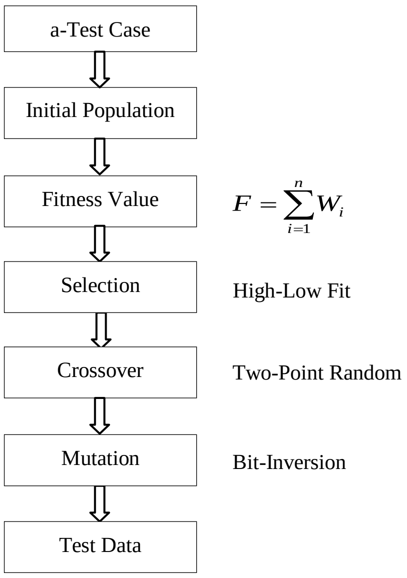


Figure 2. Operations involved in genetic algorithm.

1. *Selection*: In high-low fit method, the test cases selection is done using the combination of more high fitness values with less low fitness values. So, the resultant chromosomes with better fitness's are taken for next, next generations.
2. *Two-point crossover*: The crossover facilitates us to combine individuals i.e. reproduction chromosomes that were selected. Two-point crossover which calls for two points to be selected from the parents; the values between two points are swapped together, to get two off springs. Consider this example,
   Testcase1: 1 3| 2 5 4| & 7 8 3
   Testcase2: 1 2| 4 3 5| & 9 1 2 3 3
   Offspring1: 1 3 4 3 5 & 7 8 3
   Offspring2: 1 2 2 5 4 & 9 1 2 3 3
3. *Bit-inversion mutation*: Mutation alters one or more gene values in a chromosome from its initial value. Value Occurrences mutation attempts to replace a duplicate value of an individual with a missing value to improve the individual's fitness. For example, the value 4 is repeatedly coming in a particular test case that value alone will be replaced by a new value and is shown below:
   Testcase1: 1 4 4 & 5 9
   Offspring: 1 4 2 & 5 9
4. *Fitness function*: The fitness function used is weighted-sum approach. The evaluation of fitness gives better individuals. These metrics translate objectives such as quality objectives (usability, reliability), organizational objectives (scalability), and environmental objectives (security, privacy) into some measurable attributes of a candidate solution. The FF formula is given below:

$$F = \sum_{i=1}^{n} W_i \tag{2}$$

Where, W_i - weight of a ith node and n - number of test cases.

3.2. Test Case Prioritization

For the generated test cases, priorities have to be assigned and this is done using prioritization; it also checks the frequency of occurrences. By considering the execution time of an algorithm and to minimize the resource cost, the prioritization is usually being performed only for limited test cases from the generated test set.

Covering more number of conditions is the main criteria to be considered for the entire testing process; which in turn increasing the fault detection capability. Clustering technique is chosen because it is an unsupervised machine learning technique for prioritization. The parameters considered and their range of values is tabulated in Table 2.

Table 2. Test case prioritization parameters.

S. No.	Parameter	Range
1	Condition Coverage	70-95%
2	Number of Faults Detected	5-12
3	Execution Time	(1000-6000)ms
4	Number of Clusters	12-40

3.2.1. Implementation of Clustering Algorithm

In the existing, they use only pair-wise combination whereas the proposed technique named GWCA reduces the number of clusters by combining four test cases altogether to form a cluster thereby reducing the execution time of an algorithm; using both Intra and Inter clustering. Intra-cluster is prioritization of test cases within a cluster. The criteria taken here is Fault Detection Capability (FDC). Inter-cluster is prioritizing the clusters and the criterion is Condition Coverage (CC). After applying the prioritization technique, each cluster will represent a single test case; all the test cases represented by each and every

cluster are combined to form an optimal prioritized test suite.
The pseudo-code for group-wise clustering algorithm is given in Algorithm 1.

Algorithm 1. Pseudo-code for group-wise clustering algorithm.

```
Input: A set of n test cases, T
Output: A dendrogram, D, representing the clusters
(1) Form n clusters, each with one test case
(2) C! {}
(3) Add clusters to C
(4) Insert clusters n as leaf node into D
(5) When there is more than one cluster
(6) Find a group of test cases with similar fitness value
(7) Merge the group for new cluster $C_{new}$
(8) Remove the group of test cases from C
(9) Add $C_{new}$ to C
(10) Insert $C_{new}$ as a parent node of the group into D
(11) Return D
```

- *Cluster formation*: Test cases with similar fitness are combined together to form a cluster. Four test cases are chosen maximum for a cluster. Representation of a test case from each cluster with assigned priority value is shown in Figure 3.

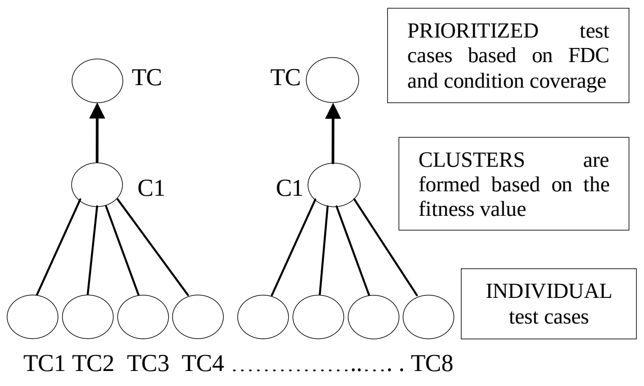

TC
TC
PRIORITIZED test cases based on FDC and condition coverage
C1
C1
CLUSTERS are formed based on the fitness value
INDIVIDUAL test cases
TC1 TC2 TC3 TC4 ……………..…. . TC8

Figure 3. Cluster formation.

Instead of assigning priorities to individual test cases, priority is being assigned to a cluster. Once the clusters are formed, intra clustering is performed, prioritization of the test cases inside the clusters. The fault detection capability, and condition coverage are considered as the multi-attributes for test case prioritization. Intra clusters are formed by taking fault detection capability as criteria and inter clusters are formed based on the condition coverage of the test cases. Finally inter clustering is performed where the individual test case represented by a cluster with its priority. Here, group-wise reduces the no of levels required. When compared to existing clustering no of levels are reduced, and time taken for prioritization is also reduced. The process of selecting test cases for prioritization is shown in Figure 4.

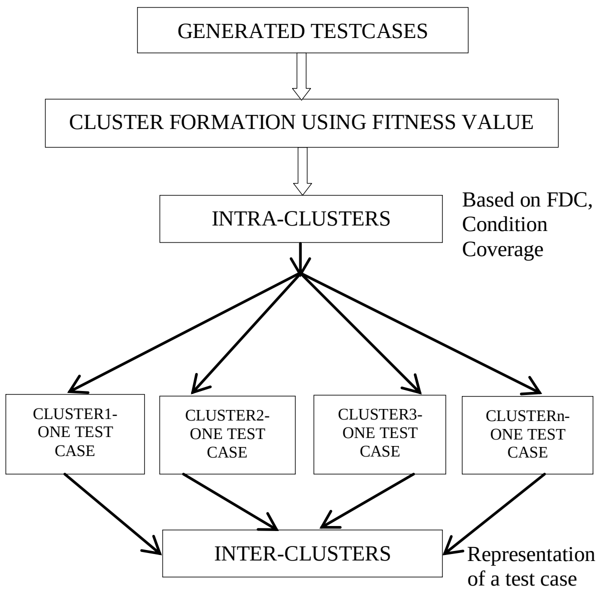


Figure 4. Selecting test cases for prioritization.

Fault Detection Capability: The ability of a test case to detect the faults or errors in programs. Fault detection capability is calculated using the formula,

$$FDC = \sum_{i=1}^{n} T_i (C_1 + C_2 + ... + C_m) \tag{3}$$

Where, m - number of conditions
n - number of test cases
C_1, C_2 ... C_m – set of conditions and
T - test case

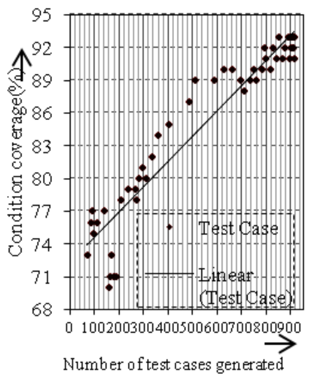
4. Result Analysis

Data set consists of SIR and existing java programs. Lines of code 70-240 with no of conditions range from (18-32) is taken for experimentation. Genetic algorithm generated test cases are taken for analysis. Fitness value of test cases depends on the no of conditions covered (by the test case). ge CC is represented in % for each test case which depends on fitness value i.e., fitness purely depends on the weightage (weighted sum approach) given to the individual nodes in the activity diagram. The test cases with totally different set of values give better results i.e., covering all paths. CC is directly proportional to fault detection capability and number of test cases to execution time. If CC is more, then FDC will also be more i.e., increase in coverage is nothing but increase in FDC. Test set with different combinations have been generated using GA (which uses a strategy for test case generation). Maximum of 1000 generations have been considered as stopping criteria for genetic algorithm.

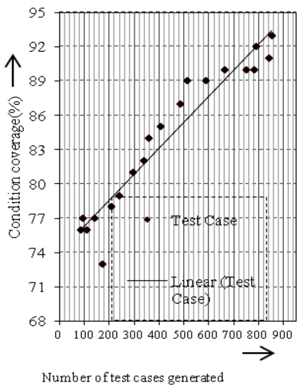
Initial testing results in coverage which is above 70% shown in Figure 5. Variance in test value is measured using the hamming distance which produces the dissimilar test cases; the variance in terms of condition coverage (%) is shown in Figure 5.

Information Flow Metric (IFM) gives the value of how much the current value converges from previous one i.e., convergence. Convergence and diversity are the commonly used metrics in GA.

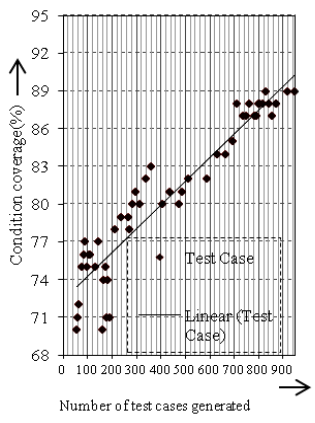
Test cases grouping used for selection and prioritization is resultant in optimized test suite shown in Figure 5. Multi-criteria optimization, GWCA used for taking the decision in test case selection and execution with improved FDC.



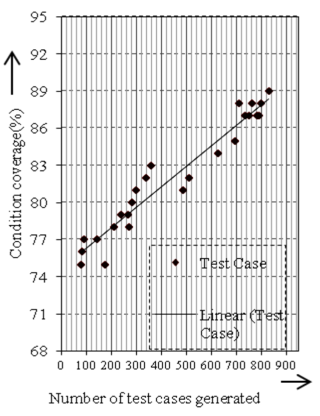
a) ELEVATOR - GA generated test cases.



b) ELEVATOR - clustered and prioritized test cases.



c) CHESS PLAYING - GA generated test cases.



d) CHESS PLAYING – clustered and prioritized test cases.

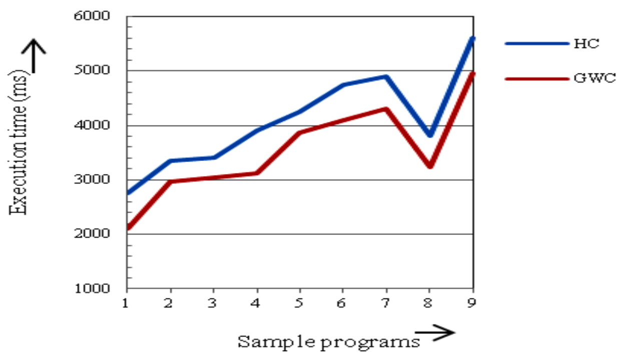
Figure 5. Initial generated test cases and improved results in reduced test suite after applying prioritization technique, group-wise clustering algorithm.

In elevator and shipping payment programs the condition coverage achieved is 91% by using GWCA whereas for library management system, CC is 94%. Number of conditions covered in GWCA is comparatively more than Hierarchical Clustering Algorithm (HCA) and the test case efficiency in terms of fault detection capability achieved upto 88.7% shown in Table 3. Cluster covering conditions decides the priority and it has been improved in group-wise clustering.

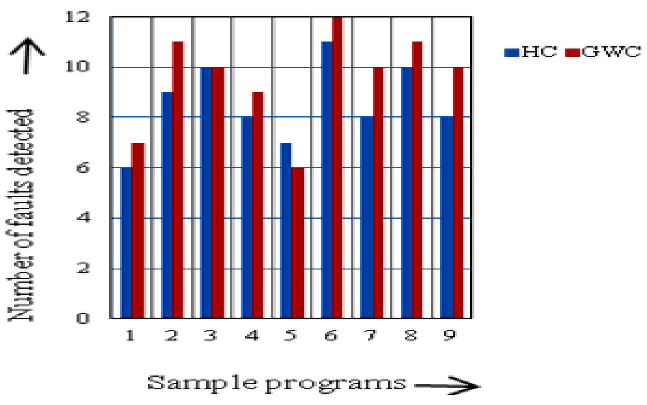
By combining more than two test cases together to form a cluster reduces the number of levels required for testing when compared to hierarchical combination. For example, programs elevator and alarm clock, number of clusters is reduced up to 12, a significant reduction. An important parameter, number of faults detected at particular time is improved very much by prioritization as proven in alarm clock, 11 faults can be detected at 2970ms whereas in previous predictions it is 9; library management system execution time was 3240ms and for soda vending machine and atm systems 4300 and 4960ms. Thus by using new clustering algorithm the execution time is reduced significantly and the results are presented in Figure 6.

Table 3. Results obtained in hierarchical and group-wise clustering algorithms.

S. No.	**Program Name**	**Number of Conditions**	**Number of Test Cases Generated**	**Number of Clusters Formed**		**Condition Coverage (%)**	
				HCA	**GWCA**	**HCA**	**GWCA**
1	Elevator	23	240	28	12	80	91
2	Alarm Clock	19	305	34	18	78	90
3	Shipping Payment	18	290	32	16	79	91
4	Stock Management	18	170	20	12	82	91
5	Student Enrolment	20	185	22	12	81	90
6	Chess Playing	32	260	30	14	76	89
7	Soda Vending Machine	28	210	24	12	73	88
8	Library Management System	24	250	30	16	85	94
9	Automated TellerMachine	35	315	36	18	78	89



a) Minimized execution time in GWCA.



b) Improved fault detection effectiveness in GWCA.

Figure 6. Improved results of group-wise clustering compared to hierarchical clustering.

5. Conclusions

Thus, the clustering technique named GWC Aperforms better in reducing number of levels for test case prioritization. In test case generation using GA, the fitness is calculated based on the weight of individual nodes. The multiple-criteria for optimization considered are condition coverage in terms of fault detection, no of clusters formed and execution time requirement. Number of clusters formed using group-wise clustering is comparatively less compared to pair-wise and hierarchical. Prioritization uses fault detection effectiveness for assigning priorities and produces ordered test cases as output. The condition coverage achieved using group-wise clustering has been improved up to 93% comparatively better than hierarchical clustering.

In future, the algorithm will be experimented with large programs by varying parameters. Similar type of criteria-based prioritization techniques can be attempted for testing. Different clustering criteria in addition to fault detection effectiveness can be considered for testing object-oriented programs.

References

[1] Abdulla S., Ramadass S., Altaher A., and Al-Nassiri A., "Employing Machine Learning Algorithms to Detect Unknown Scanning and Email Worms," *The International Arab Journal of Information Technology*, vol. 11, no. 2, pp. 140-148, 2014.

[2] Carlson R., Do H., and Denton A., "A Clustering Approach to Improving Test Case Prioritization: An Industrial Case Study," *in Proceedings of 27th IEEE International Conference on Software Maintenance*, Williamsburg, pp. 382-391, 2011.

[3] Do H. and Rothermel G., "On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques," *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 733-752, 2006.

[4] Ghiduk A., "Automatic Generation of Object-Oriented Tests with a Multistage-Based Genetic Algorithm," *Journal of Computers*, vol. 5, no. 10, pp. 1560-1569, 2010.

[5] Ledru Y., Petrenko A., Boroday S., and Mandran N., "Prioritizing Test Cases with String Distances," *Automated Software Engineering*, vol. 19, no. 1, pp. 65-95, 2012.

[6] Malhotra R. and Singh Y., "On the Applicability of Machine Learning Techniques for Object Oriented Software Fault Prediction," *An International Journal*, vol. 1, no. 1, pp. 24-37, 2011.

[7] Mirarab S., Akhlaghi S., and Tahvildari L., "Size-Constrained Regression Test Case Selection using Multi-Criteria Optimization," *IEEE Transactions on Software Engineering*, vol. 38, no. 4, pp. 936-956, 2012.

[8] Pandey A. and Shrivastava V., "Early Fault Detection Model using Integrated and Cost-Effective Test Case Prioritization," *International Journal of System Assurance Engineering and Management*, vol. 2, no. 1, pp. 41-47, 2011.

[9] Raamesh L. and Uma G., "An Efficient Reduction Method for Test Cases," *International Journal of Engineering Science and Technology*, vol. 2, no. 11, pp. 6611-6616, 2010.

[10] Roongruangsuwan S. and Daengdej J., "Test Case Prioritization Techniques," *Journal of Theoretical and Applied Information Technology*, pp. 45-60, 2010.

[11] Sabharwal S., Sibal R., and Sharma C., "Applying Genetic Algorithm for Prioritization of Test Case Scenarios Derived from UML Diagrams," *International Journal of Computer Science Issues*, vol. 8, no. 2, pp. 433-444, 2011.

[12] Singhal A., Chandna S., and Bansal A., "A Novel Approach for Prioritization of Optimized Test Cases," *International Journal on Computer Science and Engineering*, vol. 4, no. 5, pp. 795-802, 2012.

[13] Sukstrienwong A., "Solving Multi-Objective Optimization under Bounds by Genetic Algorithms," *International Journal of Computers*, vol. 5, no.1, pp. 18-25, 2011.

[14] Upadhyay A. and Misra A., "Prioritizing Test Suites Using Clustering Approach in Software Testing," *International Journal of Soft Computing and Engineering*, vol. 2, no. 4, pp. 222-226, 2012.

[15] Yoo S. and Harman M., "Clustering Test Cases to Achieve Effective and Scalable Prioritization Incorporating Expert Knowledge," *in Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*, Chicago, pp. 201-212, 2009.

Maragathavalli Palanivel received her B.E. degree in Computer Science and Engineering from Bharathidasan University, Trichirappalli in 1998 and M.Tech. degree in Distributed Computing Systems from Pondicherry University, in 2005. She joined Pondicherry Engineering College in 2006 and currently working as Assistant Professor in the Department of Information Technology. Now she is pursuing her PhD degree in Computer Science and Engineering. She has published several research papers in various refereed journals and international Conferences. She is a Life member of Indian Society for Technical Education.

Kanmani Selvadurai received her B.E and M.E in Computer Science and Engineering from Bharathiar University and Ph.D from Anna University, Chennai. She has been the faculty of department of Computer Science and Engineering, Pondicherry Engineering College from 1992. Presently she is working as Professor in the department of Information Technology. Her research interests are in Software engineering, Software testing and Data mining. She is a member of Computer Society of India, ISTE and Institute of Engineers India. She has published more than 120 papers in international journals and conferences.