Software Test Case Reduction and Prioritization

Gaurav Kumar [a*], *Vikas Chahar* [b#], *Pradeep Kumar Bhatia* [b†]
[a]Directorate of Education, GNCT of Delhi, India
[b]G. J. "University of Science & Technology, Hisar, Haryana, India"
[*]"er.gkgupta@gmail.com"
[#]"vikas.chahar@gmail.com"
[†]"pkbhatia.gju@gmail.com"

Abstract: Software testing can be made more effective by (i) eliminating the test cases which are either of no use or are redundant, (ii) prioritizing test cases on the basis of certain parameters/requirements. Reduction in Test Cse and Prioritization (TCRP) will reduce effort and cost engaged in performing execution of test cases up to a certain point. Without TCRP, there will be always much difference between estimated Testing effort and actual effort. TCRP is possible to achieve at a large extent with the help of soft computing techniques/methods. This paper examines the efficiency achieved by various existing approaches and proposes an approach in the combined form of Neural Network and Fuzzy Logic for the TCRP. Good results are achieved by this model in comparison of other existing methods used for test case minimization with requirement/fault coverage when applied on available data set..

Keywords: "Ant Colony Optimization (ACO)", "Artificial Neural Network (ANN)", "Fuzzy Logic (FL)", "Genetic Algorithm (GA)"

I. INTRODUCTION

Software Testing is very important in SDLC and consumes much of the time and cost as it identifies and resolve problems found in functionality/code of the system. Effectiveness of Software Testing may promoted by writing test cases in a way where it is capable of fault detection proficiently to produce a quality product. A deviation from specification called defect may result into system failure if it is not being identified while doing testing at development time. Size of the test suite developed early in the SDLC also grows with the growth of software resulting in the sharp increase of cost of running it (Singh et al., 2011). So, for reducing the cost, limited test-suite are required to be executed without affecting coverage of test and detection of fault capability for the test suite (Jones et al., 2003). Test cases of same type can be placed in an equivalent cluster and there should be no need to test each test case from this cluster. Although faults may be caused due to anomaly found in requirements, expected and actual outcomes, and non-functional faults such as "performance, security, scalability, or compatibility" etc. Because of limitations of resources, it seems impractical to execute program with all possible combination of values for all variables and checking coverage for all paths, conditions, code etc. (*Khan et al., 2009*).

Computers can learn faster and analyze numerous amounts of data in an easy manner with the application of soft computing techniques. A simple application of soft computing can be seen in online shopping of Amazon or Flipkart where a customer is recommended to choose the related product based on analyzing the preference of customer which in turn depends on the web pages that the customers visit, or what he/she search for, and what they buy on the platforms. In soft computing, human intelligence and effort can be proper utilized by identifying unique and innovative test environments [*Nag, 2017*]. Soft Computing techniques can be effectively utilized in optimizing and generating test cases, prioritizing and automating testing, UI testing enhancement, reduction in analysis tasks with very little maintenance.

A. Test Case Reduction (TCR)

With the growth of program size, we can add new test cases for test suite so that new functionality can be validated that may incur duplicate test cases. "A test suite reduction technique reduces the costs of executing, validating, and managing test suites" (*Sprenkle et al., 2005*).
(*Rothermel et al., 2002*). "Test cases should be minimized to increase efficiency of software testing by selecting a representative subset of test cases for the entire input domain satisfying requirements r_is, decisions, definition-use associations, or specification items. The test-suite reduction may stated as follows" (*Khan et al., 2009*):

*"*A test suite *T* for a set of test-case requirements" $r_1, r_2, \ldots, r_n$ should give "desired test coverage of the program and subsets of *T i.e.* $T_1, T_2, \ldots, T_n$ where test cases t_j belonging to T_i for one of the r_is can be used to test requirement r_i with the goal of same coverage" as was of the initial test suite.

Generic steps followed for reducing cases:

1. Write manually set of test cases or through automated tool.
2. For each test case Build coverage and dataset
3. Apply proposed test case reduction methodology and eliminate unneeded test cases
4. Analyze effectiveness of Test Case Reduction Method

B. "Test Case Prioritization (TCP)"

Test cases may be prioritized for reducing of regression testing cost such that test cases which are more important, can be executed earlier. "Due to time and resource constraints, it may not be possible to execute all tests in suites for each iteration of testing". *TCP* "can help in increasing rate of fault detection of a test suite by scheduling test case execution in an order to maximize some objective function *e.g.* number of faults detected", number of requirements covered, testing of most critical and complex functionality of the system (*Gandhi et al., 2014*). When not available to execute test cases then prioritized test cases should be executed (*Saini et al., 2012*). To do regression testing of the

software , test cases may be arranged in a way that test cases with highest priority should come first and so on giving high fault detection rate (*Khandai et al., 2011, Srivastava, 2008*).

TCP may be stated as follows:

$(\forall T)$, Find $T' \in PT$ such that $(T \neq T')$ and "[f(T')≥f(T)]" where,

"T" is given test suite,

"PT is set of all possible prioritizations of T",

"f is function" ;

Various techniques for TCP are as follows:

- *Requirement-based Prioritization:* "Test cases are prioritized based on the software requirements like requirement instability, requirement complexity, custom-priority" etc.
- *Cost Effectiveness based Prioritization:* "Test cases are prioritized based on cost of test results analysis *i.e.* defects discovered during previous test runs and cost of test execution".
- *Code/Statement coverage based Prioritization*: "Test cases are ordered based on the highest statement coverage and/or additional statements that were not covered initially".
- *Prioritization based over Chronographic history :* "Test cases are prioritized based on the history of test executions. The criteria can be maximum defect detection, failure rate or test complexity".

"*Total fault exposing potential Prioritization*": "Test case fault exposing potential is determined based on total faults seeded (through mutation) and total faults detected from the test case".

II. BACKGROUND

Test suites size increases with evolution of software, execution of test suites becomes costly. Reduction of test suit significantly reduce the size of test suites without degrading fault detection capability of reduced suites as compared to corresponding un-reduced suites (*Raamesh et al., 2010*). Redundant test cases should be transformed instead of discarding for effective fault detection capability (*Singh et al. 2011*).

Test data generation technique based on GA reduces "cost of software testing by more than 75% as compared to Random Testing (RT) technique". Fitness function evaluates after using relations among nodes of the program's control flow graph by using concept of dominance. (*Ghiduk et al., 2010*). Test cases can be selected, identified and reduced from a large test suite in optimum time by using a combined approach of genetic algorithms and bee colony optimizations (*Suri et al., 2006*). Test suite can be reduced by using data mining clustering techniques that cluster patterns wguch are similar in test cases to find redundancy incorporated by automatic generated test cases. (*Muthyala et al., 2011*).

A variation in testing cost and fault severities can be found on assessing a "metric showing rate of fault detection by prioritized test cases" (*Malishevsky 2006, Panigrahi et al., 2011*). Prioritization techniques for test cases may chosen based on time (execution and validation), defect (occurrence and impact), requirement (assigned by customer, implementation complexity, change of requirement, requirement coverage) or complexity(test case complexity, test impact etc.) so that "rate of fault detection" by test suites can be improvised (*Elbaum et. al., 2000, Muthusamy et al., 2013*). A test case that covers the maximum number of modified lines is assigned highest priority to firstly execute it while the test cases with least coverage of modified lines are executed in last because they are assigned lowest priority. Further, test cases can be prioritized using GA "that takes test case information from regression testing as input and produces a sequence of test case to be executed such that the maximum number" of MC/DC can be achieved (*Jones et al., 2003; Jacob et al., 2013*).

III. SOFT COMPUTING TECHNIQUES

Various soft computing techniques are available that can be used to optimize software testing by reducing and prioritizing test cases as given under (Bhateja, 2016):

A. "Artificial Neural Networks (ANN)"

This is a mathematical modelling of human NN which can learn from past experience that includes training phase and generating output for unknown inputs based on learning. In unsupervised Self Organizing Map (SOM) Neural Network, computational neurons are arranged into a grid that maps continuous low dimensional spatially discrete output space input space to the. A broad SOM algorithm can be given as under:

1. Initialize weight vectors w_j by randomized values.

2. Extract "training input vector x from the input space".

3. Calculate minimum value of $d_j(x) = \sum_{i=1}^{D}(x_i - w_{ji})^2$ to find "winning neuron I(x) with weight vector closest to the input vector".

4. Update "weight $\Delta w_{ji} = \eta(t) T_{j,I(x)}(t)(x_i - w_{ji})$ where $T_{j,I(x)}$ is a Gaussian neighbourhood and η(t) is the learning rate".

5. From step 2 repeat it until the feature map gets stable

B. Fuzzy Logic (FL)

For reducing the size of test suite *FL* can be used by help of "Fuzzy C-Means Clustering algorithm (FCM)" which is based on the principle that a *n* number of clusters can belong to dataset , where each data point in the dataset belongs to each cluster to a certain degree Any point of data close to the center of a cluster will have a high degree of belonging or membership to that cluster and another data point that lies far away from the center of a cluster will have a low degree of belonging or membership to that cluster" (*Alata et al., 2008*). The FCM algorithm (Kumar et. al., 2013) list of k number is returned of cluster centers $C = \{c_1, \ldots, c_k\}$, where, Partition "Matrix $M = u_{i,j} \in [0,1], i = 1, \ldots, n; j = 1, \ldots k$ ". The cluster centers and the membership grades for each data point are updated to minimize objective function and distance from any given data point to a cluster center.

$$Objective functon, J_m = \sum_{i=1}^{N}\sum_{j=1}^{k} u_{ij}^{m} \left\| x_i - c_j \right\|^2, 1 \leq m < \infty$$

$$Standard function, u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|}\right)^{\frac{2}{m-1}}}$$

$$Centroid\ of\ a\ cluster, c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} . x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

C. "Genetic algorithm (GA")

GA is used for optimization/searching tech. For minimizing the search for better performance. Using the popular concept of GA behaviour and Genetic structure behaviour of the chromosomes of individuals. Three operators are used on its population:

"i) Selection ii) Crossover and iii) Mutation".

"Genetic Algorithms work efficiently when the search space is large in size, composite and not clearly understood, when domain knowledge is inadequate or expert knowledge is difficult to encode". A general GA algorithm is given under (Bhawna et. al. 2016, Jeyaprakash et. al. 2015):

1. Identify total Chromosomes as total no. of Test Cases and Initialize the population;
2. Evaluate the population (Using the given fitness function); /* Find value of F*/
3. Find scaled fitness values; /*$F' = aF + b$ */

   3.1. $if\ F_{min} > \left(\frac{F_{avg} * C_{mul} - F_{max}}{C_{mul} - 1.0}\right)$

   {

   $\delta = F_{max} - F_{avg}$;

   if($\delta \leq 0.00001$)

   $a = 1; b = 0;$

   $else$

   {

   a = $\frac{(C_{mul} - 1.0) * F_{avg}}{\delta}$;

   b = $F_{avg} * \frac{F_{max} - C_{mul} * F_{avg}}{\delta}$;

   }

   }

   3.2. $else$

   {

   $\delta = F_{avg} - F_{min}$;

   if($\delta \leq 0.00001$)

   $a = 1; b = 0;$

   $else$

   {

   a = $\frac{F_{avg}}{\delta}$;

   b = $-F_{min} * \frac{F_{avg}}{\delta}$;

   }

   }

4. $while\ NotSatisfied(Termination\ Creiteria)$

   {

   Select parents for reproduction;

   Apply crossover and mutation by swapping of chromosomes;

   Evaluate population;

   Remove duplicate values and finalize best Chromosomes to find scaled fitness values.

   }

D. "Ant Colony Optimization (ACO)"

The real ants behavior is simulate to locate the shortest path using algorithm to the food source. For communication of information between the individuals, decision of path to follow is based on the pheromone trails. "The moving ant lays some pheromone on the path for marking which can be detected by another ant moving around and decides to follow it with high probability, and then reinforces the trail with its own pheromone to marking the path for other one. ACO technique has been applied to solve many optimization problems like travelling salesman problem, knapsack problem, distributed networks, telecommunication networks, and test data generation". A general ACO algorithm is given under (Dorigo et. al. 2005, Mohapatra et. al. 2015)):

"**Input**: An instance P of a CO problem model P=(S,f,Ω)
InitializePheromoneValues (Γ)
s_{bs} ←NULL
while termination conditions not met do
  G_{iter}←ϕ
  For j=1,...,n_a do
    s←ConstructSolution(Γ)
    if s is a valid solution then
      s←LocalSearch(s) {optional}
      if(f(s) < f(s_{bs})) or (s_{bs}=NULL) then s_{bs}←s
        G_{iter}←G_{iter} U {s}
      end if
    end for
  ApplyPheromoneUpdate(Γ, G_{iter}, s_{bs})
end while
output: The best-so-far solution s_{bs}"

IV. PROPOSED MODEL

The combined form of Neural Network and Fuzzy Logic acts as a powerful tool by integrating expert knowledge and numerical data toreduce as well as prioritize test cases. "Proposed Neuro-Fuzzy model has been derived from many existing approaches to get benefits of parent models. Model has been validated through data got from PROMISE Software Engineering Repository of NASA projects". The crisp data is converted to linguistic variable using Fuzzification which is then passed to Inference Engine. Triangular functions are used as the membership functions that is a 3 pt function defined by "minimum (α), maximum (β) and modal (m) values, i.e. T (α, m, β), where (α ≤ m ≤β). For defuzzification, Centeroide Method which calculates Centre of Gravity (COG)" area under the curve has been used.

$$T = \frac{w_1(a \propto^b) + w_2(am^b) + w_3(a\beta^b)}{w_1 + w_2 + w_3}$$

Here "w_1, w_2 and w_3" are "weights which is optimistic and are most likely and pessimistic. Here maximum weight is given for most expected cluster of test cases. ($a\alpha^b$) denotes optimistic cluster, (am^b) denotes most likely cluster and pessimistic clusteris denoted by ($a\beta^b$). "Proposed model has been implemented in MatLab R2013 using ANFIS (Adaptive Neuro-Fuzzy Inference System) with a hybrid learning algorithm of least-squares method and back-propagation gradient descent" (for samples with less test cases) and To identify parameters of Sugeno-type fuzzy inference systems

Resilient BPNN (for samples with large test cases) have been used as shown in Fig. 1. Finally, on the basis of evaluations, Effectiveness (η) of the proposed model can be calculated as:

$$\eta_{Proposed\ \ Model} = \left(1 - \frac{achieved\ values}{original\ values}\right) \times 100$$

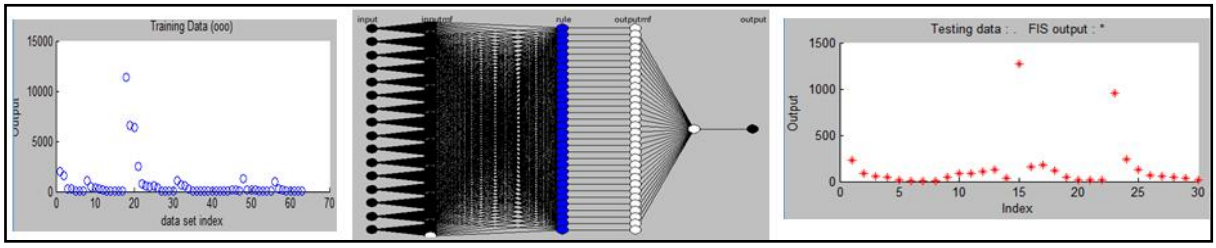

Fig. 1. ANFIS implementation of training data, model structure and testing

V. Evaluation & Result Analysis

The proposed Neuro-Fuzzy model is implemented in MATLAB to find out a representative set and is compared with existing approaches as in Table I and II. Data set for research work is taken from PROMISE data repositories. Proposed approach is helpful in regression testing to avoid execution of all test cases by finding optimum number of test cases in terms of reduced & prioritized test cases with fault/requirement coverage.

Fig. 2 represents graphical view for optimized test cases and its execution time based on the evaluation done on the datat for Table I and Table II.

Table I. Test case reduction % for proposed and existing approaches

Sample	Actual Test Cases	SOM NN		Fuzzy Clustering		GA		ACO		Proposed Neuro-Fuzzy	
		Result	Reduction %	Result	Reduction %	Result	Reduction %	Result	Reduction %	Result	Reduction %
1	15	13	13	8	46.67	12	20	8	46.67	8	46.67
2	20	17	15	13	35	16	20	12	40	13	35
3	50	31	38	27	46	40	20	30	40	26	48
4	65	40	38.46	33	49.23	56	13.84	35	46.15	30	53.84
Average Reduction %		26.11		44.22		18.46		43.20		45.87	
Effectiveness (η)		32.67		46		17.33		43.33		48.67	

Table II. Execution Time effectiveness for proposed and existing approaches

Sample	Actual Execution Time	SOM NN		Fuzzy Clustering		GA		ACO		Proposed Neuro-Fuzzy	
		Result	Reduction %	Result	Reduction %	Result	Reduction %	Result	Reduction %	Result	Reduction %
1	210	183	12.85	172	18.09	170	19.04	174	17.14	178	15.23
2	193	161	16.58	177	8.29	154	20.20	151	21.76	156	19.17
3	984	773	21.44	741	24.69	756	23.17	753	23.47	723	26.52
4	810	672	17.03	646	20.24	685	15.43	651	19.62	638	21.23
Average Reduction %		16.97		17.83		19.46		20.49		20.54	
Effectiveness (η)		18.57		20.98		19.66		21.30		22.84	

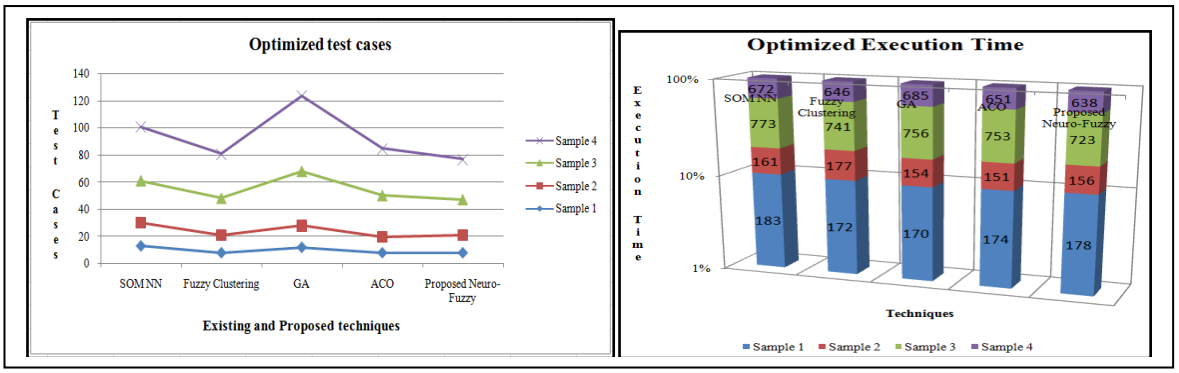

Fig. 2. Graphical view for optimized test cases and its execution time

VI. Conclusion

Soft computing techniques offer a good opportunity to optimize effort estimated and actually spent in software testing. In future, soft computing techniques offers human testers an interesting and valued aspect of testing. Here, a detailed evaluation and analysis of "TCRP with soft computing techniques (e.g. ANN, FL, GA, ACO) have been done". Final result indicates that in terms of reduction % and

effectiveness,. Keeping this in view, a combined form of NN & FL for TCRP has been proposed which gives better results as compared to existing approaches. However, proposed approach is for training or learning etc. In future, some improvements may be advised for development of TCRP model using combined approach of different techniques like "Ant Colony Optimization, Bee Colony Optimization" etc. which may overtake the performance given by proposed "Neuro-Fuzzy model".

Bibliography

[1] Acharya, A. A., Mohapatra, D. P., & Panda, N. (2010, December). Model Based Test Case Prioritization for Testing Component Dependency in CBSD Using UML Sequence Diagram. *International Journal of Advanced Computer Science and Applications*, *1*(6), 108-113.

[2] Akthar, S., & Rafi, S. M. (2010). Improving the Software Architecture Through Fuzzy Clustering Technique. *Indian Journal of Computer Science and Engineering*, 1(1), 54-57.

[3] Alata, M., Molhim, M., & Ramini, A. (2008). Optimizing of Fuzzy C-Means Clustering Algorithm Using GA. *World Academy of Science, Engineering and Technology*, 15, 224-229.

[4] Bhateja, N. (2016, May). Various Artificial Intelligence Approaches in Field of Software Testing. International Journal of Computer Science and Mobile Computing, 5(5), 278-280.

[5] Bhawna, Kumar, G., Bhatia, P.K. (2016, May). Software Test Case Reduction using Genetic Algorithm: A Modified Approach. *International Journal of Innovative Science, Engineering & Technology*, 3(5), 349-354.

[6] Dorigo M., & Blum C. (2005). Ant colony optimization theory: A survey, In Theoretical Computer Science, 344(2–3), 243-278.

[7] Keywords: Ant colony optimization; Metaheuristics; Combinatorial optimization; Convergence; Stochastic gradient descent; Model-based search; Approximate algorithms

[8] Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2000, Aug.). Prioritizing Test Cases for Regression Testing. *International Symposium of Software Testing and Analysis*, 102-112.

[9] Gandhi, S., & Gupta, D. (2014). Test Case Reduction and Prioritization. *International Journal for Scientific Research & Development*, 2 (6), 664-666.

[10] Gantait, A. (2011). Test case Generation and Prioritization from UML Models. *2nd IEEE International Conference on Emerging Applications of Information Technology*, 345-350.

[11] Ghiduk, A. S., & Girgis, M. R. (2010). Using Genetic Algorithms and Dominance Concepts for Generating Reduced Test Data. *Informatica Journal*, 34, 377-385.

[12] Jacob, T. P., & Ravi, T. (2013). Optimization of Test Cases by Prioritization. *Journal of Computer Science*, 9(8), 972-980.

[13] Jeyaprakash, S., Alagarsamy, K. (2015). A Distinctive Genetic Approach for Test-Suite Optimization. *Elsevier International Conference on Soft Computing and Software Engineering*, 62, 427-434.

[14] Jones, J. A., & Harrold, M. J. (2003, March). Test-Suite Reduction and Prioritization for Modified Condition/ Decision Coverage. *IEEE Transactions on Software Engineering*, 29(3), 195-209.

[15] Khan, S. U. R., Rehman, I. U., & Malik, S. U. R. (2009, October). The Impact of Test Case Reduction and Prioritization on Software Testing Effectiveness. *IEEE International Conference on Emerging Technologies*, 416-421.

[16] Khandai, S., Acharya, A. A., & Mohapatra, D. P. (2011). Prioritizing Test Cases Using Business Criticality Test Value. *International Journal of Advanced Computer Science and Applications*, 3(5), 103-110.

[17] Kumar, G., & Bhatia, P. K. (2013, September). Software Testing Optimization through Test Suite Reduction using Fuzzy Clustering. *CSI Transactions on ICT*, Springer India, 1(3), 253-260.

[18] Malishevsky, A. G., Ruthruff, J. R., Rothermel, G., & Elbaum, S. (2006, March). Cost-cognizant Test Case Prioritization. *Technical Report (TR-UNL-CSE-2006-0004)*, Deptt. Of CSE, University of Nebraska, U.S.A., 1-41.

[19] Mohapatra, S. K., & Prasad S. (2015, December). Test Case Reduction Using Ant Colony Optimization for Object Oriented Program. *International Journal of Electrical and Computer Engineering*, 5(6), 1424-1432.

[20] Muthusamy, T., & Seetharaman, K. (2013, December). A Test Case Prioritization Method with Weight Factors in Regression Testing Based on Measurement Metrics. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(12), 390-396.

[21] Muthyala, K., & Naidu, R. P. (2011, June-July). A Novel Approach To Test Suite Reduction Using Data Mining. *Indian Journal of Computer Science and Engineering*, 2(3), 500-505.

[22] Nag, P. (2017, September 26).How Artificial Intelligence is changing the Dynamics of Software Testing.Available: http://www.tothenew.com/blog/artificial-intelligence-in-software-testing/ [2017, October 31]. Panigrahi, C. R., & Mall, R. (2011). Test Case Prioritization of Object Oriented Programs. *SETLabs Briefings*, 9(4), 31-40.

[23] Raamesh, L., & Uma, G. V. (2010). An Efficient Reduction Method for Test Cases. *International Journal of Engineering Science and Technology*, 2(11), 6611-6616.

[24] Rothermel, G., Harrold, M. J., Ronne, J. V., & Hong, C. (1998, November). Empirical Studies of Test-Suite Reduction. *IEEE International Conference on Software Maintenance Proceedings*, 34–43.

[25] Saifan, A. A., Alsukhni E., Alawneh, H., & Sbaih A. A. (2016, October). Test Case Reduction Using Data Mining Technique. *ACM International Journal of Software Innovation*, 4(4), 56-70.

[26] Saini, R., Saini, S., Gupta, D., & Rana, A. (2012, July). Reduction in Test Cases using Regression testing approach and cost effective test prioritization testing techniques - APFD measure. *International Journal of Scientific and Research Publications*, 2(7), 1-8.

[27] Singh, N. P., Mishra, R., & Yadav, R. R. (2011, August). Analytical Review of Test Redundancy Detection Techniques. *International Journal of Computer Applications*, 27(1), 30-33.

[28] Sprenkle, S., Sampath, S., Gibson, E., Pollock, L., & Souter, A. (2005, September). An Empirical Comparison of Test Suite Reduction Techniques for User-session-based Testing of Web Applications., 21st IEEE International Conference on Software Maintenance Proceedings, 587 - 596.

[29] Srivastava, P. R. (2008). Test Case Prioritization. *Journal of Theoretical and Applied Information Technology*, 4(3), 178-181.

[30] Suri, B., Mangal, I., & Srivastava, V. (2006). Regression Test Suite Reduction using an Hybrid Technique Based on BCO And Genetic Algorithm. *International Journal of Computer Science & Informatics*, 2(2), 165-172.

[31] Yoo, S., & Harman, M. (2012, March). Regression Testing Minimisation, Selection and Prioritisation: A Survey. *Software Testing, Verification & Reliability Journal*, 22(2), 67-120.