ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 6, Issue 7, July 2017

A CRITICAL REVIEW ON TEST CASE PRIORITIZATION AND OPTIMIZATION USING SOFT COMPUTING TECHNIQUES

Renuka singh[1] (M.tech scholar)
Email id-rssinghrenuka@gmail.com
Sania Thomas[2](Assistant Professor)
computer science engineering

Subharti Institute of Technology and Engineering (Meerut, India)

ABSTRACT

Software testing plays a crucial role in software quality assurance. It is a core phase of software development life cycle at a same time it is very expensive and time consuming process too. It becomes very inefficient for a tester to re-execute large number of test cases again and again for small changes. Hence, test case prioritization and optimization techniques are used to schedule the test cases and select the optimum subset of relevant test cases from pool of test suit. This paper gives the insight into existing test case optimization and prioritization techniques such as greedy algorithms, Genetic Algorithm, Ant Colony Optimization, Bee Colony Optimization and hybridization of these algorithms. We focus on the limitations of existing techniques to make it clear that which technique is more suitable for test case optimization and prioritization by a comparative review of these algorithms.

Key-words: Test case prioritization techniques, regression testing, Soft computing technique etc.

I. INTRODUCTION

In this competitive, demanding and growing world of business the key factor for achieving success is quality maximization with time and cost minimization. Customer satisfaction is taken as a crucial parameter for quality measurement [1, 7].Software testing is an important and necessary step that plays a crucial role in software development life cycle as it is done to give assurance about the quality of software. Testing is a process of identifying and uncovers the errors before delivering the software to the end user. Software testing contains three main steps: selection of test inputs, execution of inputs on the software under test and result evaluation [2]. Testing is agrueling, strenuous, time consuming and expensive process. Software testing aims to design minimum number of test cases which can reveals as many faults as it can [3]. We can conclude that software testing is basically an inspection or investigation process for providing quality assurance of software. Testing is necessary as it assure that application will not result in any kind of failure because the process of correcting it will become very expensive in the later stage.

1.1 Regression Testing

Regression test is a black box testing technique that tests the system to guarantee that it is working as per the requirement even after the modifications [8, 9]. The most appropriate test case order is the one that uncover errors at the early stage but the location of errors are not known in advance that's why test case prioritization techniques depends on some available substitutes or expedient for prioritization criteria[10]. It is difficult and inefficient to execute test cases again and again even for small change occurring in the software [11]. Regression testing is important as alteration in any software can cause more harm to it.Techniques for performing regression testing [12]:

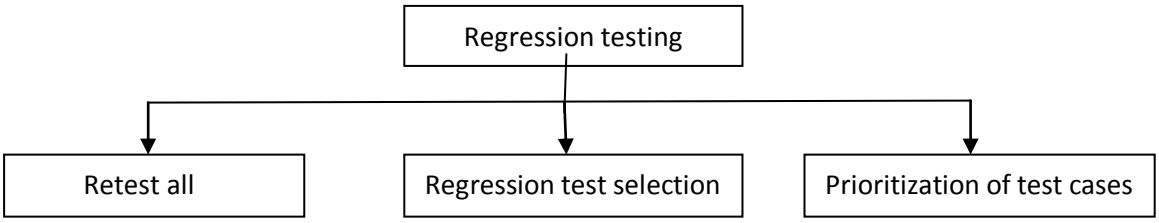

Fig 1: Classification of Regression testing

Retest all: this is the expensive and time consuming method as it involves re-executing of all the test cases. It is not applicable for minor changes in the system.

Regression test selection: this involves re-execution of a selected part of test suits.

Prioritization of test cases: this includes priority based selection of test cases which will as a result reduce the regression test suit.

1.2 Test Case

Test cases are those statements, clauses or terms which checks that whether software is working correctly or not. Most of the companies usually save the test suits for re-executing it. Large numbers of test cases are build for testing any application thus testing is a time consuming process. For previous version of application several test cases are developed which can be used to test the new version of application but if we apply these test cases for latest version than it will take significant amount of time therefore a tester should select the appropriate test cases and arrange them in proper order for execution [4]. For example, executing complete test suits for a product having 20,000 lines of code can take more than six weeks [1].The selection and order of test cases are based on different projects goals and performance criteria. Rothermer et al. (1999) summarized and classified the testing objectives into following types in test case prioritization [5].

- Increase the rate of defect detection-tendency of identifying defect at early stage.
- Increase the rate of detection of high-risk defects-tendency of identifying more critical defect at early stage.
- Increase the rate of regression errors discovered- the tendency of identifying defects at early stage, whenever code changes.
- Increase the rate of the code coverage-large amount of code should be covered.
- Increase the confidence in the reliability of the system- tester should have confidence in the reliability at faster rate.

1.3 Test Case Optimization:

Three main phases of software testing are: selection of test inputs, processing of the inputs on the software which has to be tested and comparison of the result. Software testing is a continuous activity which is performed during the development process of the software so that the errors that are making software to run in different way can be detected. Test suits once developed are updated and reused frequently even when software is modified as a result some of the test cases become antiquated and redundant. It is necessary to optimize the available test suites due to time constraint for re-executing the large test suites. Therefore, test case prioritization, test case selection and test suit minimization are use for the optimization of test suites [2].

1.4 Test Case Prioritization:

The primary goal of test case prioritization is to have a higher fault detection rate so that confidence can be achieved in terms of reliability of the system [13, 9, 14, and 15]. While performing test case selection and prioritization some key problems are identified in software testing with respect to adequacy criteria and influence of test cases on software quality. Adequacy criteria are the prototype that determine whether software is tested adequately [2]. Two basic advantages of test case prioritization are[6]:

- Large numbers of bugs are found under resource constraint condition.
- Time limit expend for fixation of bugs.

Basically test case prioritization is scheduling the test cases such that the highest priority test cases are executing before the lower priority test cases and test case optimization means selecting the best subset of test cases from the large number of pr-defined test cases. Test cases selected should be in such a way that it should meet the following conditions like code coverage should be maximum, large number of requirement should be covered and fault detection rate should be high[2].

Test case prioritization problem is defined as follow [13,9,14]

Given: test suites T, PT refers to the number of ways chosen, f is a function whose value depends on permutation of these T to some real numbers.

Problem: we have to find T’ such that T' E PT for all T, (T’EPT) where (T! =T’)and f (T’)>=f (T).

Basically there are two categories in which test case prioritization is divided [ieee1]:

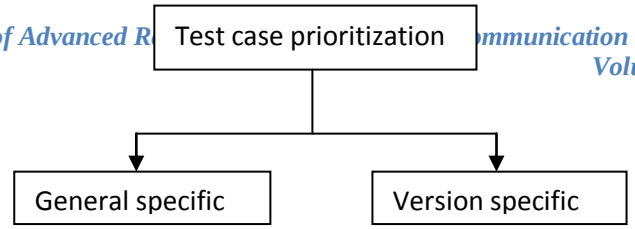

Fig 2: Classification of Test case prioritization

Considering a program P having a test suits T, in general specific test case prioritization knowledge of modified program is not required for calculating the prioritized test suits T' whereas in version specific test case prioritization the knowledge of modified program is needed for calculating prioritized test suits T'.

II. RELATED WORK

A tabular representation of existing approaches for test case prioritization and optimization is represented below in table 1.

Author	Approach used	features	Limitations	Conclusion
G.V.Uma	Genetic algorithm	Defined various concrete prioritization factors to identify more severe faults at earlier stage.	not efficient for complex code	Better rate of fault detection and reduces the execution time.
Arup BbhinavAcharya	Greedy approach	Prioritized the test cases for testing component dependency.	Limited component coverage	Implemented in dynamic environment and was found very efficient.
Eldon Y.Li	ACO	Higher-risk test cases are identified to improve the performance.	Not effective for large program	Improvement in execution time and budget.
Praveen RanjanSrivastava	Variable length genetic algorithm	Improve the testing efficiency by selecting the most critical path	Intractable for medium size software	With the help of critical path, testing efficiency is improved.
Suman	Genetic algorithm	Dynamically test cases are prioritized based on code coverage	More defected code may cover at last	Introduced cyclic crossover to make testing process effective and efficient.
BhartiSuri/ IshaMangal	Hybrid approach	Hybrid of BCO and genetic algorithm to reduce the testing time and cost	Less efficient for complex projects	Introduced HBG-TCS tool and the proposed hybrid approach were much faster than ACO.
R. Krishna Moorthi	GA	Prioritize the test case to increase the rate of fault detection.	Fixed for short testing time constraints	Using structure based criteria; APFD metric was generated that proves the effectiveness of the proposed algorithm.
E.Ashraf	novel	Clustering of effective factor	Tested on limited data set	APCC metric was used to show that proposed algorithm was more

				efficient.
Ashimasingh	GA	Uses an intelligent dynamic approach that will prioritize the test cases on the basis of priorities.	No cluster based prioritization approach	CMP metric proves the effectiveness of the algorithm developed.

Above table summarizes the existing work that identifies the various test case prioritization and optimization techniques. Large number of authors have proposed various different types of techniques for ordering test suits that results in high fault detection rate, time saving, cost minimization and quality assurance.

In [10], the author have developed requirement based system level test case prioritization technique that uses set of some concrete or abstract prioritization factors (PF) for detecting or revealing the most severe faults at early stage and improves the quality of software using genetic algorithm.

In [11], the author have used greedy approach to developed a method for prioritizing test cases for testing component dependency in component based software development (CBSD) environment. This technique was found very effective when applied for component developed in java.

In [5], the author hasdefined a process of test case prioritization using ant colony algorithm for optimizing defect detection rate and regression test performance. It proves that maximum defects are discovered with minimum number of test cases.

In [3], the author have defined a variable length genetic algorithm for optimizing the efficiency of software testing by selecting most critical cluster path that is those parts which contain large amount of error. So that number of test cases is not needed to run in complete program.

In [30], the author has introduced a new genetic algorithm that prioritizes test cases dynamically based on complete code coverage. Main aim is to reduce the number of test cases. In the end analysis was also done on the basis of process cost and test cost.

In [18], the author haveproposed a hybrid approach of bee colony and genetic algorithm for reduction of number of test cases and developed a tool HBG-TCS to implement the proposed approach. This proposed technique showed large amount of reduction in test suits and the tool presented above run much faster than ACO technique.

In [21], the author hasproposed a new technique based on genetic algorithm that increases the rate of fault detection. Basically, this paper is based on time constrained that is it says if time taken for execution of test cases is known in advance than test cases will be efficiently prioritized.

In [24], the author has proposed a technique based on genetic algorithm that generates test case from UML state diagram that is algorithm based on heuristic technique is presented. This technique is reducing the effort of generating test cases.

In [28], the author hasproposed an value based algorithm based on SIX factors that are priority of customer, requirement change, complexity of implementation, requirement traceability execution time and fault impact. The result shows that proposed value based algorithm generate better results than existing one that select random values.

In [27], the author haspresented an approach that provides sequence and reduction of test cases by using intelligent dynamic approach. They have also analysed the effect of genetic algorithm that how fast it find the faults. Effectiveness of Sequenced test cases was evaluated using CMP cumulative mutation probability.

In [25], the author has proposed a genetic algorithm that will perform test case prioritization with in a time constrained environment. This algorithm automates the process of test case prioritization. He alsoproposed a hybrid algorithm based on particle swarm optimization and genetic algorithm. Effectiveness was measured using APFD/APCC, efficiency, saving, reduction. Particle swarm optimization is blended with a crossover operator of genetic algorithm.

In [4], the author has presented different strategies of test case prioritization for web application. They used factors for prioritization like test lengths, frequency of appearance of request sequences, coverage of parameter value and their interaction.

III. TEST CASE PRIORITIZATION TECHNIQUES

3.1Greedy Algorithm [16]

The principle on which greedy algorithm works is that the highest weight element is taken first than the second highest weight element followed by third highest weight element and this continues till the suboptimal and complete solution is not produced.

Let's take an example, based on statement coverage in table 2:

TABLE 2

Statement / Test case	1	2	3	4	5	6	7	8
A	X	X	X			X	X	X
B	X	X	X				X	X
C	X	X	X	X				
D					X	X	X	X

In the above example, maximum statements are covered by test case A (i.e., six statements) therefore, test case A will be selected first. Test case B will be selected next as it covers five statements. Now, equal number of statements are covered by test cases C and D therefore, greedy algorithm will produce the test case sequence as A,B,C,D or A,B,D,C.

3.2Additional Greedy Algorithm [16]

It has the different strategy from greedy algorithm that it iteratively chooses the highest weight element of the problem from that part which was not considered by previously chosen elements.

For example, consider table 1 with four test cases and eight statements, test case A is selected as it covers maximum number of statements. Now we can see that test case A has not covered statement 4 and 5 therefore instead of selecting test case B we will select test case C or D as the cover the uncovered statements 4 and 5. Therefore, the sequence of test case will be either A, C, D, B or A, D, C, B.

3.3 2-Optimal Algorithms[16]

It is refer as K-optimal greedy approach that usually selects the K elements that covers the largest part of the problem.

For example, in table 1 we can see that test cases C and D if taken together than they will cover all the statements (i.e. 1 to 8) which in result covering the maximum number of statements. Therefore, the test case sequence produced by 2-optimal approach will be C, D, A, B.

3.4 Ant Colony Optimization:

It was proposed by Dorigo in 1992, it is based on the food searching process of ants. Whenever ant's searches for food, they leave pheromone on the travelled path and the shortest path will be identified by the evaporating process of pheromone and by the team work of ants. Initially ant choose their path randomly even when fork in the road come in between. Evaporation of pheromone is in different evaporation rate depending upon the length of paths that result in different residual pheromone. Therefore the longer path will leads to the less residual pheromone because of high evaporation of pheromone [5]. From source to destination path various edges are chosen by ants to construct an optimal path so the next edge that an ant choose is given as:[17]

$$\rho_{edge} = \frac{[\tau(e)]^{\alpha}.[\eta(e)]^{\beta}}{\sum edge_{ij}\,[\tau(e')]^{\alpha}.[\eta(e')]^{\beta}} \quad (1)$$

Where e is the nodes visited, e' is the nodes not visited, α and β are the pheromone rates (fixed as 1), τ is the amount of pheromone laid and η isfavourable edge to move.

3.5 Genetic Algorithm:

It was invented by John Holland in 1960. It is based on the guesses and improving them through evaluation process, "survival of the fittest". It uses the concept of fitness function to choose the strongest amongst all. Fitness value of an individual is determined as [13]:

Fitness=2*(pos-1)/ (nind-1) (2)

Where, pos refers to position of individual and nind refers to population size

Method involves the following [10]:

Initialization: initial population is generated from many individual solutions. Size of population depends upon problem nature that contains thousands of possible solution.

Selection: during each generation, a proportion of population is selected to form a new generation. With the help of fitness function the better solution is selected.

Crossover: it is done by exchanging the segments between pair of chromosome that is by switching substring of one chromosome with another chromosome.

Mutation: it alters an individual in the population by swapping the strings.

The optimal solution is searched on the basis of desired population. Initialization of test cases is done depending upon the problem. Fitness function will be use to select the suitable population of problem after that operation like crossover and mutation will be applied and finally solution is checked that weather it is optimized or not. Flow chart for the same is given below [10]:

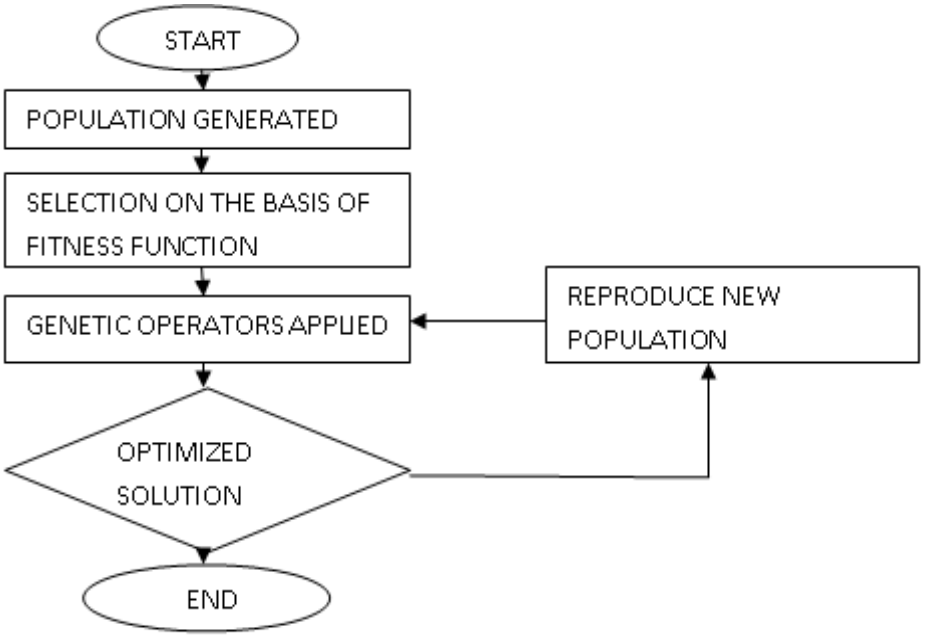

Fig 3:Flow of Genetic algorithm

3.6 Bee ColonyOptimization:

It is an optimization technique where food positions are modified with time by artificial bees. The colony comprises of three types of bees [18]:

Employed bees: they hunt for their food source and after returning performs dance in the area

Escort bees:The employee bees after finding the abandoned food becomes escort and find the new source of food.

Onlooker's bees: depending upon the dance of employed bees they decide their food source.

The below table shows the comparison among all the algorithm describes above:[16]

Table 3

Algorithms	approach	advantage	Drawback
Greedy algorithm	Incrementally selects the highest weighted element.	Simple and inexpensive	Provides Local optimal solution
Additional greedy algorithm	Iteratively chooses the highest weighted element from the part that was not	Very Effective and cheaper in implementation and execution.	It requires coverage information to be updated for the remaining test case.

	considered earlier.		
2-optimal algorithm	Selects the elements together that cover the largest part of the problem.	Fast and Effective in terms of performance.	Gives suboptimal results.
Ant colony optimization algorithm	With help of pheromone value ant chooses the optimal path from source to destination	It runs continually even after any dynamical change in the graph.	
Genetic algorithm	Individuals are selected from the population based on the fitness function and then they combine and mutate to offer a new generation.	Finds the solution in reasonable computation cost	Not significant for small programs.

IV. PRIORITIZED TEST SUIT EFFECTIVNESS:

For evaluating the performance of prioritization technique, it is required to measure the effectiveness of the sequenced/ordered test suit. APFD metric is used to measure the effectiveness [31].

Average Percentage of Faults Detected (APFD) Metric [31].

APFD metric measures the rate of fault detection per percentage of test suite execution. APFD is measured by taking weighted average of the percentage of faults detected during the execution of the test suit. Range of APFD values is from 0 to 100; higher the value better will be the fault detection rate. APFD is calculated as:

APFD=1-{(tf1+tf2+……+tfm/mn)}+(1/2n)

Where n are the number of test cases, m are the number of faults and (tf1,……,tfm) are the position of first test T that exposes the fault.

V. Conclusion:

One of the key issues in software testing is test data generation that is selection of optimal subset of test cases from the pool of test suit. Properly generated optimal test cases helps in the reduction of cost, time and maintenance with maximum assurance about the quality and high rate of fault detection. Large number of test case prioritization and optimization techniques has been developed with the objective of increasing the fault detection rate, maximum code and customer requirement coverage and minimum efforts for testing an application. However the above presented critical review of developed techniques have done a great contribution for achieving these objectives but still the problem like not obtaining efficient/ effective result in the case of complex and large size of software is faced sometime. Therefore, techniques like genetic algorithm, ant colony optimization, bee colony optimization or hybrid of these techniques may be well suited for test case prioritization and optimization.

REFERENCES

[1] Amit Kumar, Karambir SinghCOMPUSOFT, An international journal of advanced computer technology, 3 (5), May-2014 (Volume-III, Issue-V) ‘A Literature Survey on test case prioritization’ Research Scholar, CSE Department, UIET Kurukshetra University Asst. Prof., CSE Department, UIET Kurukshetra University

[2]ArunSharma,manojkumar,rajeshkumar,wseas transaction on information science and application. “Optimization of test cases using soft computing techniques: a critical review”

[3]Srivastava, Praveen Ranjan, and Tai-hoon Kim. "Application of genetic algorithm in software testing."*International Journal of software Engineering and its Applications* 3.4 (2009): 87-96.

[4]Sampath, Sreedevi, et al. "Prioritizing user-session-based test cases for web applications testing." *Software Testing, Verification, and Validation, 2008 1st International Conference on*.IEEE, 2008.

[5]Shen, Chien-Li, and Eldon Y. Li. "APPLY ANT COLONY ALGORITHM TO TEST CASE PRIORITIZATION."

[6]mssunita, msmamtagulia “study of regression test selection technique” 2014 international journal

of advance research in computer science and software engineering.

[7]J. Karlsson and K. Ryan, "A Cost-Value Approach for Prioritizing Requirements," *IEEE Software*, vol. 14, no. 5, pp. 67-74, Sep-Oct 1997.

[8]A.Ansari, K.Devadkar and Dr.prachiGharpure,"optimization of test suits-test case in regression test,"IEEE 2013.

[9] G. Rothermel. R.H. Untch. C. Chu. andMJ.Harrold. "PrioritizingTest Cases for Regression Testing," IEEE Trans. Software Eng., vol. 27,no. 10, pp. 929-948, Oct. 200 1.

[10]S.Raju and G.V.Uma," Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm," European Journal of Scientific Research.

[11]Arup acharya R. Kavitha and Dr. N. Suresh Kumar, "Model Based Test Case Prioritization for TestingComponent Dependency in CBSD Using UML Sequence Diagram," *International Journal ofAdvanced Computer Science and Applications*, Vol. 1, No. 6, pp.108-113, Dec 2010.

[12] *International Journal of Computer Applications (0975 – 8887) Volume 116 – No. 16, April 2015* A Review on Various Techniques for Regression Testing and Test Case Prioritization

[13] N Sharma, G.N. Purohit and Sujata,"test case prioritization techniques an empirical study", IEEE 2014.

[14] S. Elbaum. A. Malishevsky. andG.Rothermel ' Test case prioritization: A family of empirical studies". IEEE Transactions on Software Engineering, February 2002.

[15] Yu. Yuen Tak. and Man Fai Lau: "Fault-based test suite prioritization for specification-based testing" Information and Software Technology, vol. 54, no. 2, pp. 179-202, 20 12.

[16] search algorithm for regressioin test case prioritizationbyzheng li.

[17] 2014 IEEE International Conference on Advanced Communication Control and Computing Technologies (lCACCCT)Parallelized ACO Algorithm for Regression TestingPrioritization in Hadoop FrameworkNandhiniElanthiraiyan\Chamundeswari Arumugam 2I Master of Software Engineering,2ProfessorDepartment of Computer Science and Engineering,SSNCollege of Engineering,Chennai, India

[18] Bhartisuri and ishamangal,"Analyzing Test Case Selection using Proposed Hybrid Technique based on BCO and Genetic Algorithm and a Comparison with ACO",Volume 2, Issue 4, April 2012 International Journal of Advanced Research in Computer Science and Software Engineering

[19] SangeetaSabharwal, RituSibal and Chayanika Sharma," Applying Genetic Algorithm for Prioritization of Test Case Scenarios Derived from UML Diagrams", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011

[20] Dr. ArvinderKaur et al," A BEE COLONY OPTIMIZATION ALGORITHM FOR CODE COVERAGE TEST SUITE PRIORITIZATION", International Journal of Engineering Science and Technology (IJEST)

[21] R.Krishnamoorthi1 and S.A.Sahaaya Arul Mary2," Regression Test Suite Prioritization using Genetic Algorithms"International Journal of Hybrid Information Technology
Vol.2, No.3, July, 2009

[22] Dr.arvinderkaur and shubragoyal,"A GENETIC ALGORITHM FOR REGRESSION TEST CASE PRIORITIZATION USING CODE COVERAGE", International Journal on Computer Science and Engineering (IJCSE)

[23]Dr. arvindourkaur and shivangigoyal."A Bee Colony Optimization Algorithm for Fault Coverage Based Regression Test Suite Prioritization", International Journal of Advanced Science and Technology Vol. 29, April, 2011

[24] ChartchaiDoungsa-ard, KeshavDahal, AlamgirHossain, and TaratipSuwannasart
"An Automatic Test Data Generation from UML State Diagram using Genetic Algorithm"

[25]Dr. Arvindkuamr and divyabhatt," Particle Swarm Optimization with Cross-Over Operator for Prioritization in Regression Testing:", *International Journal of Computer Applications (0975 – 8887)Volume 27– No.10, August 2011*

[26] Arvinderkaur and shubragoel," A Genetic Algorithm for Fault based Regression Test Case Prioritization," *International Journal of Computer Applications (0975 – 8887) Volume 32– No.8, October 2011* 30

[27] Ashiamsingh and Kamal pralkash," Fault Based Analysis to Perform Test Case Prioritization in Regression Testing", International Journal of Advanced Research in Computer Science and

Software EngineeringVolume 2, Issue 9, September 2012

[28] E.Ashraf," Value based Regression Test Case Prioritization", Proceedings of the World Congress on Engineering and Computer Science 2012 Vol I WCECS 2012, October 24-26, 2012, San Francisco, USA

[29] AncaDeak, Tor Stalhane,"organization of testing activities in Norwegian software companies", IEEE sixth international conference on software testing 2013.

[30] Suman, Seema,"a genetic algorithm for regression test sequence optimization", international journal of advance research in computer and communication engineering 2012.

[31] *International Conference on Research Trends in Computer Technologies (ICRTCT - 2013) Proceedings published in International Journal of Computer Applications® (IJCA) (0975 – 8887)* 1 Effectiveness of Testcase Prioritization using APFD Metric: Survey R. Pradeepa Assistant Professor, Department of Computer Applications, Sri Ramakrishna Engineering College, Coimbatore, Tamilnadu, India K. VimalaDevi, Ph.D. Professor, Department of Computer Science, Kalasalingam University, Krishnankovil, Tamilnadu, India