International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 16 (2017) pp. 5578-5586

ACO based test case prioritization for fault detection in maintenance phase

Sushant Kumar[1] and Prabhat Ranjan

Department of Computer Science, Central University of South Bihar, India.

[1]Orcid id: 0000-0002-7577-9819

Abstract

The regression testing in maintenance phase is costly and time consuming phase of software development life cycle (SDLC). The cost of testing in maintenance phase has almost eighty percent of the total cost. The testing is performed in maintenance phase is called regression testing. The regression testing is unavoidable maintenance activity that is performed several times in SLDC. Therefore, optimization of regression testing like test case prioritization is highly required for reducing the testing time in maintenance phase. The two popular techniques for optimization of testing techniques are test case selection and test case prioritization. These techniques have integrated with different soft computing algorithms for getting fruitful regression test cases for making test suites. In this paper test case with prioritize order using modified ant colony optimization (ACO) is used better results. ACO will find the best test cases that have find the maximum fault in minimum time. The proposed approach has validated with benchmark example. The result shows the effective test case has selected with high APFD score.

Keyword: Regression Testing, Maintenance Phase, ACO, APFD, Test Case, Test case prioritization

INTRODUCTION

Software system is continuous grow during development and maintenance phase. This has changed with much cause such as adding new functionality, predicting and correcting errors, and enhances the performance [7]. When the software is changed for better application or modification it is compulsory to perform regression testing. The intention of regression testing is to ensure that software still behaves like intended and modification not adversely impact its quality [3].The result from previous paper shows that regression testing has utilized 80 percent of the testing cost therefore it is highly costly operation in maintenance phase [4][5][6].The organization cannot bear this huge amount of cost in maintenance. Researcher and industry is taking more concern to reduce the cost of maintenance and testing, one of the effective and well proven methods are regression testing that is used in testing for modified version of software. Since major cost in modified part is link with testing so regression testing has crucial role in version testing. There are many regression approaches are follow for regression testing techniques some of them are re-test all data, test case selection and test case prioritization. Testing again with all data is not possible and it is also not much desirable. In regression testing, the role of selection and prioritization are major factor for making new test suites. Selection and prioritization both have optimization technique and work for some objective that decides the quality test data find out from test suite in early. In testing activity software quality has to be checked with different testing techniques. The most used testing criteria are statement coverage, branch coverage, path coverage, and fault coverage. This has good criteria for software testing and used many papers and covers one given objective fruitfully. However the new era is moving towards more on computation so need more quality and reliable product in the market because it directly links with human safety in future. The need for quality test data is needed after every modification of software. So it is highly recommended have to full fill more and more objective with quality test case. Some authors have already started the work on this field to find the test data that is solve the many purpose of testing. The work performed is not up to the mark in this field so more research is going on this field. The motive in this work is to select and prioritize the test case with multi objective approach concept. This paper used two objectives to select test data has maximum coverage of fault with minimize the execution time. The fitness function has to certify the given two objectives. It has proven that random approach for selection is obsolete and other method like greedy approach is still using for test data selection. The problem in greedy is that it only search for next better solutions. It is effective method for nearest solutions but it has not successful for global solutions. Some modified greedy algorithm has used like optimal and 2- optimal greedy it has perform better than greedy and random algorithm however it is still suffering for global result. Some researcher has start working on handling of test data problem using artificial intelligence. Search based software engineering (SBSE) is one of the fruitful applications for software optimization. SBSE has fruitful applied in different phases of the SDLC however the analysis of paper [23] shows that search based method is most effective in software testing compare than other phases. Phil McMinn [22] has defined SBST (search based software testing) method that in enhancement for SBSE. Now many soft computing algorithms are applying on software testing to find quality data. Some conventional methods like genetic algorithm, hill

climbing are simulated annealing are using in test case optimization it performance is much better than greedy and random algorithm however it is still suffering for global solutions. Some advanced meta-heuristic algorithm has applied on SBST namely Cuckoo search (CS) Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO) and Harmonic Search (HS). All above algorithm have nature inspired and work in optimized way in nature. The goal is to solve the multi-objective test optimization that give the accurate result and satisfy the objective like code coverage, take less time and max fault coverage.

The hybrid approach of advanced soft computing techniques useful for coverage of the many objectives like Fuzzy- PSO, NFPSO, FACO, CSPSO and many more that are used in the test case optimization. Soft computing meta-heuristic is comparable new in software engineering field so mapping of test case is major role to select the meta-heuristic algorithm or hybrid algorithm approach for selection of test case.

The structure of the paper includes test case prioritization, ant colony optimization, with modified algorithm, and last part is conclusion.

TEST CASE PRIORITIZATION

Test case optimization for regression test optimization has used different technique like selection [27], minimization and prioritization [24]. Rothermal has mentioned that they are all having same working rule and roughly same meaning [17]. Test case generation is another aspect for test case optimization, detail categorization has mentioned by Sushant et.al.[21],[25] to classification of work in this area. It is not possible to check every test case after every time change in code. It is proved that testing task is impractical for re-test all. Some optimization technique is important to increase the quality of software and reduce the testing process. The test case optimization is being achieved by test case minimization (TCM), test case selection (TCS), test case prioritization (TCP), test case classification (TCC) and test case reduction (TCR) [19]. S.Yoo et.al [19] has suggested that all the optimization approach has slightly related to each other i.e. one approach cover other approach. The survey shows that [19] selection and prioritization have most popular work for test case optimization, sometimes they are working like hybrid approach. One of the optimization approaches like classification is important in test case optimization but has given very little concern in software testing. Test case optimization has critical problem in software development due to large number of test case in software it is not possible to test all the data .If any test suite has 20 test case then 20! Possible combination has to checked in software .This is even not possible for powerful supercomputer to test the data in given time frame. There are many conventional approaches and heuristic approaches are used for test case optimization like greedy search and many more but all are suffer from local optima. It has been observed that soft computing meta-heuristic application has most effective in software testing phase in software development [23]. There are many soft computing technique apply in different phases of software development. Some of the fruitful approach is genetic algorithm, fuzzy logic, artificial intelligence and many more. The objective of testing solved by some adequacy criteria must be satisfied. For any testing techniques at least one adequacy criteria must be checked and verified for testing. A good approach to enhance the quality of selected test suite by making of test case that have used for multiple purpose i.e. one test case has can do many operations. The multi-objective criteria is satisfied when one test case have at least solve two objective. The selection of objective is also the important task and it will depend upon software to software because some software is more concern about cost and some software more concern about time and quality. There are many objectives that is use for checking the adequacy of the software like max branch coverage, max requirement coverage, min cost, min mutation score, and max fault. Selection of adequacy criteria is another conceptual approach that may vary according to software to software every adequacy has weight according to software and depending upon their critical condition. Here we select three objectives that are covered by author either in single objective or multiple objectives. Annibale et.al. [2] is given two objective problems and three objective problem. It include maximum coverage, minimum cost, and maximum fault. In the test suite every test case have important power to solve test case in our problem the two test case objective has been used from the different objective some of the important category are given below.

1) Code coverage
2) Execution cot of test cases
3) Fault Coverage

Code Coverage: The code coverage is defined by path coverage, branch coverage and statement coverage the selection depends upon the data sets you have used if the program is small then we can we used any of the operations but if the data sets is taken very large like SIR data sets then path coverage is not possible then statement coverage or branch coverage has used.

$$\mathrm{Cov}(X) = \frac{1}{n}\sum_{c=1}^{n} c_c \qquad (1)$$

Here n, the total code coverage and cc is equal to 1 then the code is covered by at least one test case if not then it will 0.

Execution cost: The execution cost measure the execution time of the test case. The actual performance depends upon so many parameters and it is not easy task, we concern about executing code in the program.

$$\mathrm{Cost}(X) = \sum_{c=1}^{n} x_c \,.\, \mathrm{cost}(t_c)_{\ c} \qquad (2)$$

Here cost (tc) represents the execution of the individual test case.

Fault coverage: The fault coverage is another important objective that is satisfied for test case. The test case that cover maximum fault is must be the part of test suite.

$$\text{fault(X)} = \frac{1}{f}\sum_{c=1}^{n} c_c \qquad (3)$$

Here f defined the number of test case and cc represents the fault and the value is 1 if any test case detects it. The problem formulation is designed for the test case optimization to select some adequacy criteria. In this paper, test case prioritization has performed for maximum fault in less time execution, this decrease the cost and effort.

Ant Colony Optimization

The ACO is family of swarm intelligence techniques. It solves the meta- heuristics problem .This is proposed by Marco Dorigo in 1992 [9] the first algorithm is used optimization of graph based on behavior of ants. There are many application of ant colony optimization in engineering problem like traveling salesman problem, test data generation, combinatorial problems, quadratic problem assignment problem, protein problem, routing vehicles problem and many more. Ant colony optimization finds the food using searching pattern of real ant colonies [9].

The procedure of ACO Meta-heuristic algorithm can be defined as given steps.

ACO ALGORITHM
While(Not termination)
Generate solutions()
daemon Actions
pheromone Update
end while

Processing of ant colony optimization

The basic functions used to compute best path in the traveling problem to source to destination. The function of ant colony optimization has defined in below section.

The selection of node

In ACO, the optimization takes the n number of value (test case) for performing the solutions. Every ant has many (different or similar) test cases for solutions. The test case is selected by randomly by any ant the next node is chosen based on his own intelligence or pheromone result [9].The equation 4 shows the basic definition of ACO.

$$p(ACO)\frac{k}{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}} \qquad (4)$$

The defined parameters α and β has fixed and used for pheromone trail and heuristic information. τ_{ij}^{α} shows the pheromone value of edge i to j and η^{β}_{ij} represents the heuristic information of ants. $p(ACO)^{k}_{ij}(t)$ is the probability value to choose path of ant k from i to j at time t.

$$h_{ij} = \frac{1}{\eta_{ij}^{\beta}} \qquad (5)$$

Heuristic information h_{ij} can be calculated from the equation 5.

Ant pheromone update:

The pheromone update is performed after collecting the information from entire colony. The equation 6 shows the pheromone update for ACO

$$\tau_{j0}(t+1) = (1-\rho)\tau_j(t) + \rho\rho_t^k best \qquad (6)$$

The τ_{j0} (t) and τ_{j0} (t + 1) denote the old and new pheromone result of the j^{th} node.

There are different versions of ACO and they are applied in engineering problem like GAACO, FUZZYACO, NFACO and many more.

The basic formulation of ant colony to find the fault has given below.

$$\text{probality } y_j = \frac{(errorscovered)\beta \ *(phermonetrail\ i\ j)}{\sum_{k=1}^{n}(errorscovered)^{\beta} * (pheromonetrail\ ik)^{\alpha}} \qquad (7)$$

The equation 7 is used to select the test case based upon errors covered without the information of execution time. J, K is the non visited test cases

Motivation and Problem formulation

The paper is formulated to solve the two objective test case problems. The test suite generates only those test cases that have two objectives. The fitness function designed according so that it covers the multi-objective. The first objective is that to cover maximum fault from list and the second is to take less time to find fault for intention to minimize the effort and cost. The equation 8 and 9 shows the fault and execution time (cost) objective.

$$fault(X) = \frac{1}{f}\sum_{c=1}^{n} c_c \quad (8)$$

$$Cost(X) = \sum_{c=1}^{n} x_c \,.\, cost(t_c)_{\ c} \quad (9)$$

Fitness function for Fault based Test case Prioritization

Input: Test Suite = t_{001}, t_{002}, t_{003},…….. t_{00n},

Output: A set of sub set S

Initialization

1: C <- ϕ ={Covered Elements}

2:F <- ϕ ={Covered Past Faults}

3:S <- ϕ ={Selected Test Cases}

4: LOOP Process

5: While (C ⊂ P)

6: for each ti∑ T do

7: $fi = \frac{fault(ti)-F}{Cost(Ti)}$

8 Tj -> test case in T with minimum fi

9 S -> S ∪tj //add tj to solutions

10 F-> F ∪ fault (tj)

11 T-> T- (Tj)

12 :endfor

13:end while

Return P

Assumptions for Test case prioritization

The effective method for test case optimization is prioritization of test case. Here mapping of test case is to discover new test suites which contains m test cases, where (m ≤ n). The reduced test suite has same power to detect the fault like original test suite. The artificial ant is used to represent the mapping of test case with ant colony.

1. T = $t_{001,}$ $t_{002,}$$t_{003,}$ ….T_{00n} //The test suite
2. F = f_{001}, f_{002}, $f_{003…}$ f_{00n} //The Set of all faults
3. The ant is moving one position node to another position node //node is test case in TCO
4. The elapsed time is Ei after moving of ant.
5. The total time constraint is defined for prioritization is the maximum time for test case.
6. t_{max} is the total time then it can be define by equation is ..

$$constraint : \sum_{i=1}^{n} t_i x_i \leq t_{max}$$

7. The total number of ant is n
8. For each ant from path i to j a list is selected that contains the test case
9. The edge weight i has defined W_i // which is the level of pheromone amount on that edge.
10. Pheromone deposited rate is mapped with 1 or 0 // if 1 then 100 % deposition rate on path.
11. The evaporation rate to be r % of W_i

ACO to maintain Diversity

The proposed diversity based ACO has some modification in ant colony optimization to maintain the diversity in the next iteration of the food search. Where the ant finds the food source and if the path of food source is small then all ant attract with that path and follow same mark for all ant, this will be the problem of diversity. So if in the next iteration the path is small we provide the diversity of the ant by change the direction for some ants to cover different direction and follow the route to cover all the software this is also solve the exploitation in original. The modified ACO algorithm enhances the heuristic information for every ant to select the path in next iteration. The randomness approach of ant is still maintain even shorter path is explored. Exploration is managed by some ant to find the other possible solution in minimum time.

Proposed test case prioritization approach using modified ACO

1. Define the code modules with faults

2. Define faults with time

3. load the fault and time of module

4. Assign the stating position

pos (p) = rand(x), rand(y)

5. Start loop

6. Solutions for free ants (nearest food source)

7. Unique food source = food uniqueness

8. Compare (unique food source, random

Number (heuristic search))

9. Pheromone level <- food quality

9.1. Food quality = NU-RT

9.2. N = number of ants that reached to food source

9.3. U= fault uniqueness

9.4. R= evaporation rate

9.5. T= Time duration for journey of food/fault

10. Perform global search

11. Perform decision using pheromone level

Decision = follow the trail (food source/fault available)

12. Update pheromone

13. Stop (all food source/fault cover)

14. End loop

The figure 1 shows the path of the ant. The moving from nest to food source has path. The figure 2 shows the flowchart of the ACO.

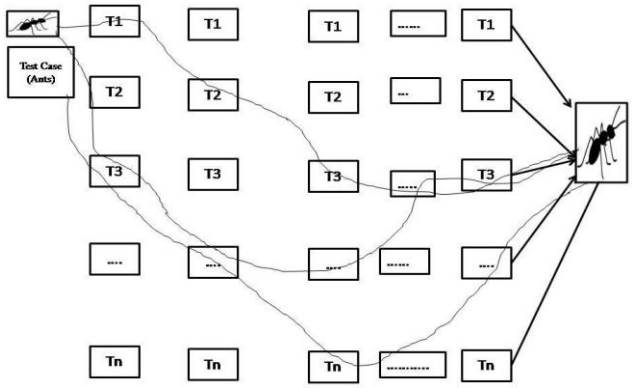

Figure 1: ACO Construction solutions.

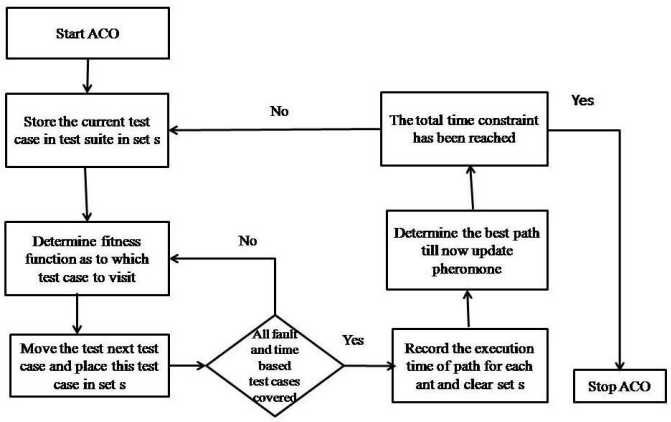

Figure 2: Flow Chart for ACO of test case prioritization

The modified formula used for find the probability of ACO for finding the fault has given below.

$$\text{probality yj} = \frac{\left(\frac{errorcovered}{executiontime}\right)\beta \quad * (phermonetrail\ i\ j)}{\sum_{k=1}^{n}\left(\frac{errorscoverd\ j}{executiontime\ k}\right)^{\beta} * (pheromonetrail\ ik)^{\alpha}} \quad (10)$$

The equation 10 has used to select the test case based upon errors covered in define execution time. J, K is the non visited test cases.

RESULTS AND ANALYSIS

There are many possible goals for optimization of regression testing. Our goal is to enhance the fault finding rate and minimize the time for cost reduction.

The most useful technique APFD (Average percentage faults detection) is to be introduced by Rothermal et.al. [18]. The APFD equation is used for measurement the percentage faults that are detected by every test suite. The best result has best test suite and take less time to cover the fault. The given formula is used for fault detection.

$$\text{APFD} = 1 - \frac{T_{f1} + T_{f2} + T_{f3} + \cdots + T_{fm}}{mn} + \frac{1}{2n} \quad (11)$$

$$\text{The Rate of Fault} = \frac{\text{No of Faults}}{\text{Exection Time}} \quad (12)$$

The equation 11, T is the test suite and used for fault detection evaluation, where m is number of faults test case is defined with n in a test suite. The equation 12 represents the rate of fault for test case.

The case study [10], [11], [12],[26] has used to validate the ant colony optimization approach.

Case Study 1

The Case study 1 has taken the data from college program for admission purpose. The table 1 shows that the test case and failure data information. The ten test case and 10 faults are defined. The initial order of test case is T_{01}, T_{02}, T_{03}, T_{04}, T_{05}, T_{06}, T_{07}, T_{08}, T_{09}, T_{10}

Table 1: Faults detected by test case and test suite for case study 1

Test Cases /Faults	T_{01}	T_{02}	T_{03}	T_{04}	T_{05}	T_{06}	T_{07}	T_{08}	T_{09}	T_{10}
F_{01}	√					√				
F_{02}				√			√	√	√	
F_{03}		√			√	√				
F_{04}							√			
F_{05}		√						√	√	
F_{06}				√						
F_{07}				√	√					
F_{08}		√	√							
F_{09}						√				
F_{10}	√									√
ET	5	7	11	4	10	12	6	15	8	9

Table 2: Test cases with fault detected and Time

Test Case	No of Faults covered	Execution Time
T_{01}	2	5
T_{02}	3	7
T_{03}	1	11
T_{04}	3	4
T_{05}	2	10
T_{06}	3	12
T_{07}	2	6
T_{08}	2	15
T_{09}	2	8
T_{10}	2	9

The test suite has information about fault and executed time. For the finding of APFD value, the fault vs execution time has calculated for each test case in test suite. The value of rates of faults detection of test case T_{01} is 0.4. Similarly the value of next test case is 0.42, 0.99, 0.75, 0.2, 0.25, 0.33, 0.133, 0.25 and 0.22. The priority has assigned in decreasing order. The test case has directly proportional to the rate of fault detection. The prioritized test suite are T_{04}, T_{02}, T_{01}, T_{07}, T_{06}, T_{09}, T_{10}, T_{05}, T_{08} and T_{03}.

The APFD for prioritized test suite after assigning the value of T_{f1}, T_{f2},…T_{fm}.

$$APFD = 1 - \frac{3 + 1 + 2 + \ \ldots \ + 3}{10 * 10} + \frac{1}{2 * 10}$$

APFD = 1.05 -0.24

AFFD =0.81

The APFD of non prioritized order is given below.

$$APFD = 1 - \frac{1 + 4 + 2 + \ \ldots \ + 1}{10 * 10} + \frac{1}{2 * 10}$$

APFD = 1.05 - 0.38

APFD = 0.72

Hence the prioritized value is 0.81. The non prioritized value is 0.72. The test suite is executed using ant colony optimization is 0.87 APFD

Case Study 2

Table 3: Faults detected by test case and test suite for case study 2

Test Case/ Faults	T_{01}	T_{02}	T_{03}	T_{04}	T_{05}
F_{01}	√		√		
F_{02}	√				√
F_{03}					√
F_{04}	√	√	√		√
F_{05}			√	√	√
ET	12	10	10	7	9

The above given test suite has five faults and five test case. The initial test case has T_{01}, T_{02}, T_{03}, T_{04} and T_{05}. The fault vs execution time has been calculated for each test case. The values are 0.25, 0.1, 0.28, 0.14 and 0.40 respectively. Priority is set in decreasing order because the lesser the value of fault vs execution time has less effective test case. The prioritized test suites are T05, T03, T01, T04 and T02 respectively. The APFD value for prioritized test case is shown below.

$$APFD = 1 - \frac{2 + 1 + 1 + 1 + 1}{5 * 5} + \frac{1}{2 * 5}$$

APFD = 1.1 – 0.24 = 0.86

The APFD value of non prioritized value has defined below

$$APFD = 1 - \frac{1+1+5+1+3}{5*5} + \frac{1}{2*5}$$

APFD = 1.1 - 0.44

APFD = 0.66

Hence the prioritized values of test suites are 0.86 and non prioritized test suites are 0.66. The test suites are implemented with ant colony optimization has 0.88 APFD value.

Case Study 3

Table 4 : Faults detected by test case and test suite for case study 3

Test Case/ Faults	T_{01}	T_{02}	T_{03}	T_{04}	T_{05}	T_{06}	T_{07}	T_{08}	T_{09}
F_{01}	√	√	√	√	√	√	√	√	
F_{02}	√	√							
F_{03}	√		√			√			√
F_{04}				√				√	
F_{05}	√		√	√		√			
ET	11	11	12	10	15	8	15	10	11

The above test suite has nine test cases and five faults. The rates of fault detection of test cases are 0.34, 0.17, 0.24, 0.28, 0.067, 0.36, 0.067, 0.3 and 0.09. The priority is set in decreasing order. The prioritized order of test suites are T06, T01, T08, T04, T03, T02, T09, T05 and T07.

The APFD for prioritized order has given below.

$$APFD = 1 - \frac{1+2+2+3+1}{5*9} + \frac{1}{2*9}$$

AFPD = 1.056 – 0.2

APFD = 0.86

The APFD of the non-prioritized order is given below

$$APFD = 1 - \frac{1+1+1+4+1}{5*9} + \frac{1}{2*9}$$

APFD = 1.056 – 0.178

APFD = 0.88

The test suite is executed under ACO has 0.88 APFD value.

Case Study 4

Table 5: Faults detected by test case and test suite for case study 4

Test Case/ Faults	T_{01}	T_{02}	T_{03}	T_{04}	T_{05}	T_{06}	T_{07}	T_{08}
F_{01}		√	√			√		
F_{02}	√			√				√
F_{03}		√			√		√	
F_{04}	√			√				
F_{05}			√					
F_{06}					√		√	
F_{07}	√		√			√		
F_{08}			√				√	
F_{09}	√			√				
F_{010}					√			√
ET	7	4	5	4	4	5	4	5

The case study 4 has test suites 8 test cases and 8 faults. The rates of faults of test cases are 0.57, 0.5, 0.8, 0.75, 0.75, 0.4, 0.75 and 0.75.

The APFD value for non prioritized order is given below.

$$APFD = 1 - \frac{2+1+2+1+3+5+1+3+1+5}{10*8} + \frac{1}{2*10}$$

APFD= 1.05 – 0.3

APFD = 0.75

The APFD values for prioritized order are given below. The prioritized sequences are T_{03}, T_{04}, T_{05}, T_{07}, T_{08}, T_{01}, T_{02} and T_{06}.

$$APFD = 1 - \frac{1+1+1+1+1+1+1+1+1+1}{10*8} + \frac{1}{2*10}$$

APFD= 1.05 – 0.12

APFD = 0.93

The APFD value for prioritized order is 0.93.

CONCLUSION

The modified ant colony optimization solves the problem of test case prioritization fault detection rate. The diversity maintain by the ACO has better exploration in later iteration. This work helps to control the convergence to achieve better exploration. This work helps in regression testing operation to reduce the cost and effort. The selection of test case in every iteration takes cost and it will take our crucial time. This work helps to select those test cases that are essential for regression testing in minimum time.

REFERENCES

[1] Pavan kumar Chittimalli and Mary Jean Harrold, ”Recomputing Coverage Information to Assist Regresssion Testing” IEEE Transaction on Software Engineering, vol. 35, No.4 pp. 452-459 July 2009

[2] Annibale Panichella ,Rocco Oliveto,Massimiliano Di Penta and Andrea De Lucia, “Improving Multi –Objective Test Case Selection by Injecting Diversity in Genetic algorithms”, IEEE Transaction on Software Engineering, vol. 41, no 4, pp.358-383, 2015

[3] Beena Raman and Sarala Subramani, “An Efficeint specific update search domain based glowwarm swarm aptimization for test case prioritization”, The International arab journal of information technology, vol 12, no 64, pp. 748-754, 2015

[4] B.Beizer, Software Testing Techniques. Van Nostrand Reinhold, 1990

[5] A.P.Mathur. Foundations of Software testing, China Machine Press, 2008

[6] G.Myers. The Art of Software testing, NY, USA, John Wiley, 1979

[7] Yogesh Singh, Software Testing ,Cambridge Publication, 2010

[8] http://mute-net.sourceforge.net/howAnts.shtml

[9] Marco Dorigo and Thomas Stutzle, ”Ant Colony Optimization”, The MIT Press London, 2004

[10] Bharti Suri and Shweta Singhal, ”Implementing Ant Colony Optimization for Test Case Selection and Prioritization”, International Journal on Computer Science and Engineering, Vol. 3, No. 3, pp. 1924-1932, May 2011

[11] Yogesh Singh, Arvinder Kaur and Bharti Suri ”Test Case Prioritization using Ant Colony Optimization” ACM SIGSOFT Software Engineering Notes, Vol. 35,no.4, pp. 1-7, July 2010

[12] Bharti Suri, Isha Mangal and Varun Srivastava, “Regression Test Suite Reduction using an Hybrid Technique Based on BCO And Genetic Algorithm”, International Journal of Computer Science and Informatics, Vol.- 2, No.1, 2011

[13] S.Yoo ,M.Harman,and S.ur, ”Highly scalable multiobjective test suite minimization using graphics cards”, In proceedings of the third nternational conference on serach based software engineering. Springer-Verlag, pp.219-326, 2011

[14] Kalyanmoy Deb, Amrit Pratap, Sameer agarwal and T. Meyarivan ,”A Fast and Elitlist Multiobjective Genetic algorithm: NSGA-II” IEEE Transactions on evolutionary computation , vol 6, pp. 182-197, 2002

[15] Priyanka Chawala, Inderveer and Ajay Rana ”A novel strategy for automatic test data generation using soft computing technique” Front.Comput.Sci, vol 9, pp. 346-363, 2015

[16] Sapna P G and Arunkumar Balakrishnan ”An Approch for generating minimal test cases for regression testing”, Procedia computer science , vol47, pp. 188-196, 2015

[17] Gregg Rothermel,Roland H. Untch and Mary Jean Harrold ”Prioritizing Test Cases For Regression Testing” Ieee Transactions on Software Engineering, Vol. 27, pp. 929-948, 2001

[18] Sebastian Elbaum, Gregg Rothermel and Satya Kanduri, ”Selecting a Cost-Effective Test Case Prioritization Technique” Software Quality Journal, Vol 12, pp. 185-210, 2004

[19] S. Yoo, M. Harman, ”Regression Testing Minimisation, Selection and Prioritisation : A Survey”,Softw. Test. Verif. Reliab., vol. 1, pp. 1-60, 2007

[20] Fabricio Gomes de Freitas,camila loiola brito maia,Gustavo augusto llima de campos and Jerffeson teixeira desouza”Optimization in software testing using metaheuristics”Sistemas de informacao, vol 5 , pp. 3-13, 2010

[21] Sushant Kumar,Prabhat Ranjan and R.Rajesh,”An overview of test case optimization using Meta-heuristic approach”, International Conference on Recent Advances in Mathematics, Statistics and Computer Science, pp.475-484, 2015

[22] P. McMinn, ”Search-Based Software Testing: Past, Present and Future” IEEE Fourth International on Software Testing, Verification and Validation Workshops (ICSTW), pp. 153-163, 2011

[23] Luciano S.de Souza,Ricardo B.C.Prudencio ,Flavio de A.Barros and Eduardo H.da S.Aranha ”Search based contrained test case selection using execution effort”Expert systems with applications ,vio 40 ,4887-4896 ,2013

[24] Sreedevi Sampath ,Renee Bryce and Atif M.Memon ”A Uniform Repersentation of Hybrid Criteria for Regression Testing”, IEEE Transactions on Software Engineering, vol. 39, pp. 1326-1344, 2013

[25] Sushant Kumar, Prabhat Ranjan, “A Comprehensive Analysis for Software Fault Detection and Prediction using Computational Intelligence Techniques, International Journal of Computational Intelligence Research, vol. 13, pp. 65-78, 2016

[26] Kamna Solamki, Yudhvir Singh and Sandeep Dalal, “A coprative evalution of m-aco technique for test suite prioritization”, Indian Journal of science and technology, vol 9, pp. 1-10, 2016

[27] Manoj Kumar, Arun Sharma and Rajesh Kumar, “An emprical evalution of a three – tier conduit framework for multifaceted test case classification and selection using fuzzy – ant colony optimization approach” software practice and experience, vol 45, pp. 949-971, 2015