Volume 2, Issue 4, April 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Analyzing Test Case Selection using Proposed Hybrid Technique based on BCO and Genetic Algorithm and a Comparison with ACO

Bharti Suri
Associate Professor
University School of Information Technology
Guru Gobind Singh Indraprastha University,
Dwarka,Delhi,India

Isha Mangal
Research Student
University School of Information Technology
Guru Gobind Singh Indraprastha University,
Dwarka,Delhi,India
ishamangal2006@yahoo.com

Abstract- **Regression Testing is primarily a maintenance activity that is performed to ensure the validity of modified software. But due to time and cost constraint entire test suite can't be executed. Thus, it becomes a necessity to reduce the test case suite and select a subset of test cases from test suite that can be executed in minimum time and has the ability to cover all the faults. A new hybrid approach based on bee colony optimization and genetic algorithm was proposed to reduce the test suite. This paper presents a tool named HBG_TCS that implements the proposed approach and the study is done to evaluate the correctness and efficiency of the proposed tool and a comparison of the proposed approach is made with the ant colony optimization.**

***Keywords*: Regression Testing, Test Case Selection, Genetic Algorithm (GA), Bee Colony Optimization (BCO), Ant Colony Optimization (ACO).**

I. INTRODUCTION

Regression Testing is an activity to make sure that modifications performed in the specific part of the software can be tested and do not incorporate other unexpected behavior [1-3]. In order to test for ambiguities in the software, we need to execute all the test cases that are prepared in the development phase, although this activity may be too time consuming and expensive.

Many techniques are proposed by researchers for reducing both time and cost. They include techniques regression test selection [4-14], regression test prioritization [15-17] and hybrid approaches [1, 2] etc. In the paper [18], the authors proposed a hybrid technique of test case selection using BCO and GA. In this paper, the above proposed technique is implemented and a comparison is provided with the regression test case selection using ACO technique [19].

Regression testing is generally done in a time constraint environment where the testing is done for a predetermined amount of time. Walcott et al [20] gave one such technique for time aware test case prioritization. Time aware prioritization intelligently schedules the test suite in terms of both the execution time and potential fault detection information. Singh et al in 2010, also proposed a time constraint prioritization using Ant Colony Optimization (ACO) and automated in [19]. In this paper a tool called HBG_TCS was developed for the proposed hybrid technique [18]. The outcomes of the execution provides near optimum results and further motivates to test the tool on various programs, to confirm the efficiency and correctness of the tool.

The paper is organized as follows. Section II discusses how the genetic algorithm works. Section III talks about the artificial bee colony optimization. Section IV illustrates the ACO technique. The proposed hybrid technique has been discussed in Section V. The experimental design of proposed approach is detailed in Section VI.

II. GENETIC ALGORITHM (GA)

Genetic algorithm, a adaptive search procedure is introduced by John Holland [21], broadly studied by Goldberg [22] and De Jong [23]. It is an optimization technique which provides near optimal solution to NP-hard problems. GA is applied to solve many problems like travelling salesman problem[24],knapsack problem[25],etc.

Genetic Algorithm is based on the idea on the natural evolution. The foundation of GA lies on the concept of the survival of fittest into a solution space. Each cycle of GA process includes initialization (encoding), selection based on fitness function, reproduction using crossover or mutation.

The cycle is repeated till a solution is found that satisfies the minimum criteria or a fixed number of generations has been reached.

III. BEE COLONY OPTIMIZATION (BCO)

Bee Colony Optimization is a swarm based metaheuristic algorithm introduced by Karaboga in 2005 [26-28].It is based on the foraging behavior of the honey bees.

BCO is an optimization technique which provides a population based search procedure, where the artificial bees modify the food positions with time [29]. The main aim of the bees' is to locate the food source positions with high nectar amount [30]. The colony of bees comprises of three groups of bees: employed bees, onlookers and scouts [29]. Employed bees forage in search of their food source and return to hive and perform a dance on this area. The employed bee who find abandoned food source becomes a scout and find a new food source again. Onlookers decided their food source depending upon the dance of employed bees.

A nectar source is selected by each bee by following a nest mate whose food source has already discovered. The bees dance on the floor area (hive), on discovery of nectar sources. This is how the onlooker bees convinced their nest mates to follow them. To get nectar ,rest of the bees follow the dancers to one of the nectar areas. On collecting the nectar they return back to their hive, relinquish the nectar to a food store. After relinquishing the food, the bee opts for one of the alternatives with a certain probability (a) abandon the food source and act as a uncommitted follower, (b) without enlisting the nest mates, continue to forage at the food source or (c) enlist the nest mates by dancing before returning to the food source. Different food areas are advertised by bee dancers within the dance area. The procedure by which the bee decides to follow a specific dancer, is not well understood but it is considered that "the recruitment among bees is always a function of the quality of the food source."

IV. ANT COLONY OPTIMIZATION (ACO)

Ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Marco Dorigo proposed ant colony Optimization in 2005[31]. ACO has its real strength in the overwhelming behavior of ants looking for a path between their colony and a source of food.

Ants are blind and they communicate with their colony by a chemical substance called pheromone. Ants while moving yields pheromone along the path traversed. This pheromone trail is sensed by other ants through which other ants follow the path with maximum pheromone trail. The process is repeated till sufficient amount of food has been gathered. This substance evaporates with time with some amount of percentage.

ACO technique has been already used in solving various combinatorial problem such as knapsack problem, travelling salesman problem, distributed network, telecommunication network, vehicle routing, test data generation.

The paper provides a comparison with the tool ACO_TCSP developed in Suri et al [19] with the tool HBG_TCS.

V. DESCRIPTION OF THE TECHNIQUE

In [18], a new approach to reduce the cost of regression testing by test case suite reduction was proposed. The proposed technique is based on concepts of BCO and GA. The technique selects the set of test case from the available test suite that will cover all the faults detected earlier in minimum execution time. Here bees are used as agents who explore the minimum set of test cases. Half of the bees will initially start foraging with randomly selected test cases. Now bees will add new test cases on her explored path if adding of a test case increases its fault detection capacity. After adding one more test case, the bees return to their hive, and exchange the information based on GA. The crossover operation is used to exchange the information. The new set of test case produced after crossover is used by new bees to forage. The process is repeated till any of the bees has discovered a set of test cases that covers all faults detection and starts to perform waggle dance.Initially the 'n/2'bees starts foraging. Each bee that have started forage will choose the test case randomly. The bees will select test case on the basis that adding test case will increase the fault detection capacity. The foraged bees will return to their hive and genetically exchange information by crossover method. Crossover will be performed between the bees whose total execution time is minimum and the bees with the next minimum execution time.If the resulted test cases total execution time is less than the maximum execution time available subset of test cases, the new bees will forage using those subset of test cases as its initial path. If the results produced after crossover does not produce new test cases or test cases which are superfluous, will not be considered. If both test sets produced after crossover will not produce new set, no new bee will forage. Now again the bees will choose the test case. Repeat The whole process is repeated till any of the bee has explored a set of test cases that can cover all the test faults. As soon as the minimum set of test case are produced, bee started to perform a waggle dance announcing her victory. So the other bees can follow this test case path.

VI. EXPERIMENTAL DESIGN

HBG_TCS is a tool developed in C++ language to implement the proposed algorithm that takes as input the test cases, their faults and their execution time and the time constraint and results in the minimum set of test cases.

A. Programs

For our analysis we used seven C++ programs and one java program. We engendered five to ten modified versions and black box test cases using fault based technique. Brief details about the programs , their sizes, versions, and test suits are provided in Table 1.

Program No	Program Name	Language Used	Size (LOC)	No.of Versions	No. of Test Cases	Total suite Execution Time (sec)

P1	Coll Admission	C++	281	5	9	105.32
P2	Hotel mangaemen t	C++	666	5	5	49.84
P3	Triangle	C++	37	6	19	382
P4	Quatratic	C++	38	8	19	441
P5	Cost_of_Pu b	C++	31	8	19	382
P6	Calculator	C++	101	9	25	82.5
P7	Prev_day	C++	87	7	19	468
P8	Raiway_boo k	Java	129	10	26	177

Table 1: Details of the selected 8 Test Programs

B. Variables

The independent variables manipulated by the experiment are:

1. Programs with five to ten faults each.
2. Various Time Constraints.

The analysis of the tool is done by calculating the ratio of the total execution time of the test suite to the reduced proposed algorithm execution time.

Execution of each program may lead to different path and is determined graphically by showing percentage reduction in test suit size and percentage reduction in execution time.

C. Design

For the correctness and efficiency of the proposed algorithm, we run each programs with same test suite for seven different time constraint values. Each run may yield a path or may not yield the path. If the test case selected covers all the faults, then our algorithm stops returning the subset of test cases chosen. The time constraint chosen is the other finishing criteria of the algorithm.

D. Data Analysis

For our testing the tool, out of the whole test pool, we randomly chose a test suite for programs P1 to P8 in which the number of test cases vary from five to twenty six test cases. The execution of each test case is recorded and is used as an input to the HBG_TCS tool.

E. Execution of HBG_TCS tool on Example

The output of HBG_TCS tool has been shown on the example[18].The test cases and their respective faults has been shown in Table 2 amd their execution time in Table 3.

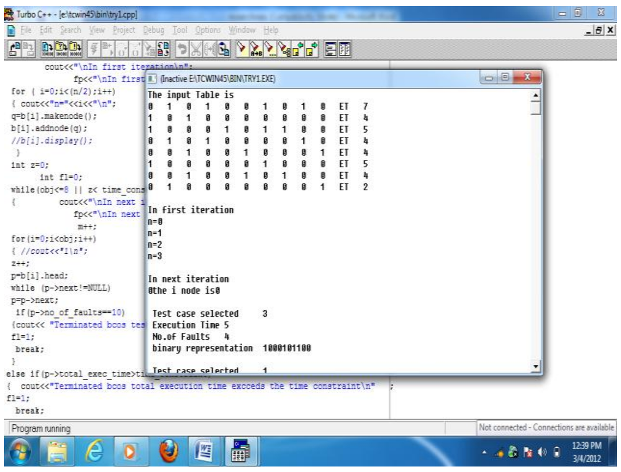
Test Case	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
T1	X		X			X			X	
T2		X						X		
T3		X			X		X			
T4				X		X		X	X	
T5	X					X				X
T6				X	X			X		
T7	X						X	X		
T8			X							X

Table 2: Sample data

Test Case	No. of faults covered	Execution Time(ET)
T1	4	7
T2	2	3
T3	3	5
T4	4	5
T5	3	3
T6	3	6
T7	3	3
T8	2	2

Table 3: Test cases, no. of faults covered, their execution time

The output screen looks like as shown in Fig 1:


Turbo C++ - [e:\tcwin45\bin\try1.cpp]
File Edit Search View Project Debug Tool Options Window Help
(Inactive E:\TCWIN45\BIN\TRY1.EXE)
Program running
Not connected - Connections are available
12:39 PM
3/4/2012

```
        cout<<"\nIn first iteration\n";
              fp<<"\nIn first
 for ( i=0;i<(n/2);i++)
 { cout<<"n="<<i<<"\n";
 q=b[i].makenode();
 b[i].addnode(q);
 //b[i].display();
  }
 int z=0;
       int fl=0;
 while(obj<=8 || z< time_cons
 {          cout<<"\nIn next i
              fp<<"\nIn next
               m++;
 for(i=0;i<obj;i++)
 { //cout<<"1\n";
 z++;
 p=b[i].head;
 while (p->next!=NULL)
 p=p->next;
  if(p->no_of_faults==10)
 {cout<< "Terminated bcos tes
 fl=1;
  break;
 }
else if(p->total_exec_time>ti
{  cout<<"Terminated bcos total execution time exceeds the time constraint\n"   ;
fl=1;
  break;
```

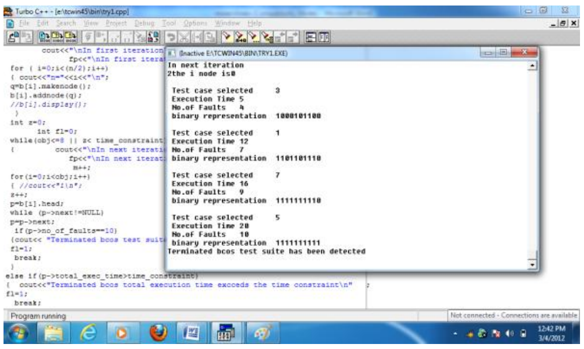
```
The input Table is
0   1   0   1   0   0   1   0   1   0   ET   7
1   0   1   0   0   0   0   0   0   0   ET   4
1   0   0   0   1   0   1   1   0   0   ET   5
0   1   0   1   0   0   0   0   1   0   ET   4
0   0   1   0   0   1   0   0   0   1   ET   4
1   0   0   0   0   0   1   0   0   0   ET   5
0   0   1   0   0   1   0   1   0   0   ET   4
0   1   0   0   0   0   0   0   0   1   ET   2

In first iteration
n=0
n=1
n=2
n=3

In next iteration
0the i node is0

 Test case selected     3
 Execution Time 5
 No.of Faults   4
 binary representation  1000101100

 Test case selected     1
```


Turbo C++ - [e:\tcwin45\bin\try1.cpp]
File Edit Search View Project Debug Tool Options Window Help
(Inactive E:\TCWIN45\BIN\TRY1.EXE)
Program running
Not connected - Connections are available
12:42 PM
3/4/2012

```
        cout<<"\nIn first iteration
              fp<<"\nIn first itera
 for ( i=0;i<(n/2);i++)
 { cout<<"n="<<i<<"\n";
 q=b[i].makenode();
 b[i].addnode(q);
 //b[i].display();
  }
 int z=0;
       int fl=0;
 while(obj<=8 || z< time_constraint
 {          cout<<"\nIn next iterati
              fp<<"\nIn next iterat
               m++;
 for(i=0;i<obj;i++)
 { //cout<<"1\n";
 z++;
 p=b[i].head;
 while (p->next!=NULL)
 p=p->next;
  if(p->no_of_faults==10)
 {cout<< "Terminated bcos test suit
 fl=1;
  break;
 }
else if(p->total_exec_time>time_constraint)
{  cout<<"Terminated bcos total execution time exceeds the time constraint\n"   ;
fl=1;
  break;
```

```
In next iteration
2the i node is0

 Test case selected     3
 Execution Time 5
 No.of Faults   4
 binary representation  1000101100

 Test case selected     1
 Execution Time 12
 No.of Faults   7
 binary representation  1101101110

 Test case selected     7
 Execution Time 16
 No.of Faults   9
 binary representation  1111111110

 Test case selected     5
 Execution Time 20
 No.of Faults   10
 binary representation  1111111111
Terminated bcos test suite has been detected
```

Fig1: Screenshots

Initially four bees started to forage. The time constraint is taken as 20 in above example. The test cases selected by each bee is shown in Table 4.

BEES	B0	B1	B2	B3
Test Case Selected	T3	T3	T7	T3
Total Execution Time	5	4	5	4

Table 4:Initial Selection of Test Cases

In II iteration, the bees chose the following test cases as in Table 5.

BEES	B0	B1	B2	B3
Test Case Selected	T3,T1	T3,T4	T7,T8	T3,T5
Total Execution Time	12	9	6	9

Table 5: Selection of Test Cases in Iteration II

Now Bees B1 and B2 perform crossing over of test cases so that new set of test cases generated will be followed by new bees as in Table 6.

BEES	B0	B1	B2	B3	B4	B5
Test Case Selected	T3,T 1	T3,T 4	T7,T 8	T3,T 5	T7,T 4	T3, T8
Total Execution Time	12	9	6	9	8	7

Table 6: Selection of Test Cases in Iteration III

In Iteration 3,we found in Table 7 that B1 has detected one path in 3 sec that can cover all the faults.

BEES	B0	B1	B2	B3	B4	B5
Test Case Selected	T3,T 1,T7	T3,T 4,T5	T7,T 8,T4	T3,T 5,T8	T7,T 4,T5	T3,T 8,T1
Total Execution Time	16	13	10	11	12	14

Table 7: Selection of Test Cases in Iteration IV

F. Time Analysis and Comparison

The execution time of each program is shown in Fig 2. using HBG_TCS tool.

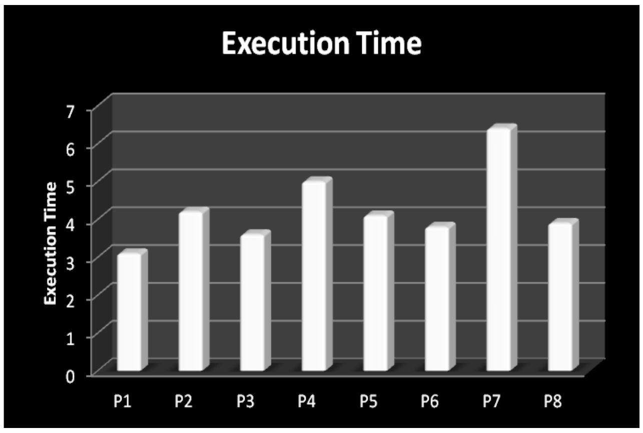


Fig2:Graph showing Execution Time of each program using HBG_TCS Tool

From Fig 3, it is clear that HBG_TCS tool based on the hybrid technique based on BCO and Genetic Algorithm of test case selection is much faster than ACO_TCSP tool for ACO. The same set of programs run on each tool. The execution time is tremendously reduced.

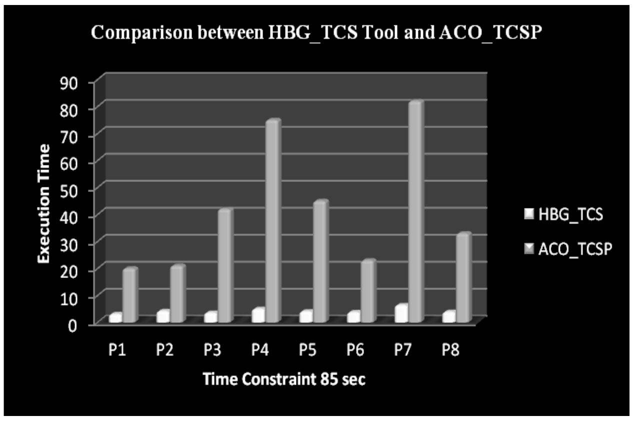


Fig3:Comparison between HBG_TCS Tool and ACO_TCSP

The amount of reduction in execution time by using HBG_Tool is shown in Fig 4 .

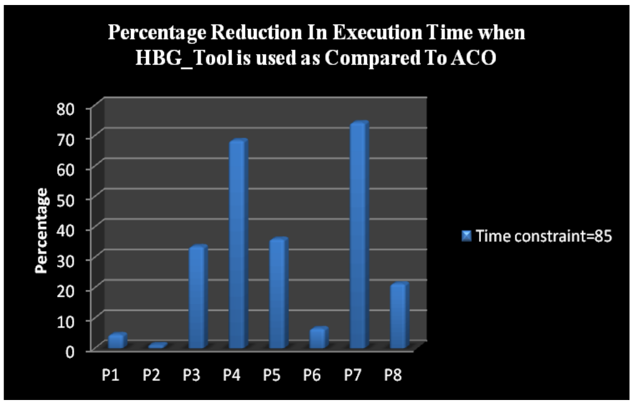


Fig4:% reduction in Execution time by HBG_TCS Tool

G. Cost Benefit Analysis

The results obtained in previous sections were further reviewed to find the cost benefits analysis of the technique.

The cost of running the HBG_Tool in addition to cost of running the selected test cases is compared with the cost of executing the complete test suite.

The percentage reduction in the size was of selected test suite using HGB_TCS Tool was calculated and is shown pictorially in Fig 5 .

If ‘n’ is size of a test suite and ‘s’ be the selected test suite using hybrid approach,then he value for percentage reduction in test suite has been calculated using:

%reduction in test suite=(n-s)/n*100

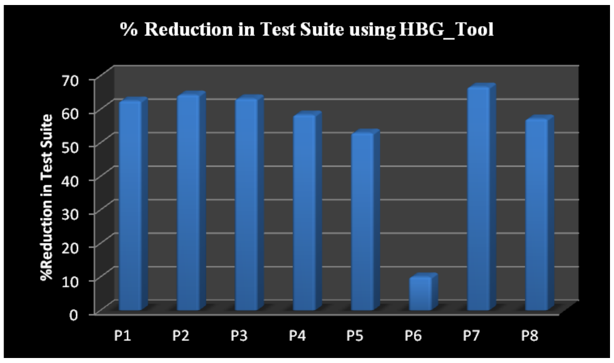


Fig 5: % Reduction in Test suite size using HBG_TCS Tool

The hybrid approach provides very high reduction in the size of test suite.

Similarly, we can calculate the percentage reduction in execution time of selected test suite .Let ‘T’ be the total execution time of the original test suite, ’Tb’ be the running time of HBG_Tool and ‘Ts’ be the total time foe executing the selected test suite .The percentage reduction in execution time using hybrid approach is computed as:

%reduction in Execution Time=(T-(Tb+Ts)/T*100

And is depicted in Fig 6.

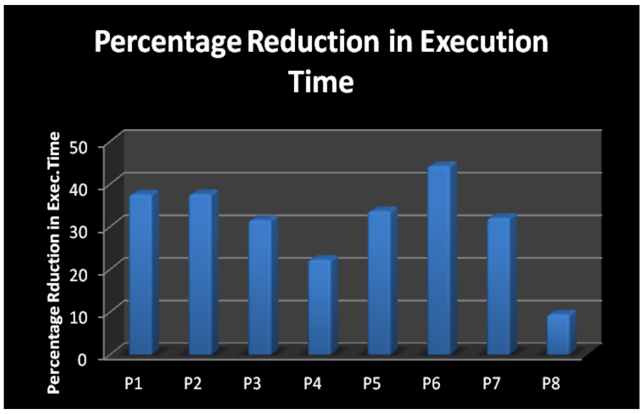


Fig 6: % Reduction in Execution Time using HBG_TCS Tool

6.8 Conclusion

In this paper, we implemented the technique proposed in [18]. The results achieved depicts that a huge amount of reduction in test suite takes place. Reduction in test suite reduces time as well as cost.

The proposed hybrid approach proves much faster than ACO technique. The tool developed runs much faster to provide the minimum subset of test cases. The tool can provide different results in each run.

REFERENCES

[1] G.Duggal, B.Suri,”Understanding Regression Testing Techniques”, COIT, 2008, India.

[2] W. E.Wong, J. R. Horgan, S. London and H.Agrawal, “A study of effective regression testing in practice,” In Proceedings of the 8th IEEE International Symposium on Software Reliability Engineering (ISSRE' 97), pages 264-274, November 1997.

[3] K.K.Aggarwal & Yogesh Singh, “Software Engineering Programs Documentation, Operating Procedures,” New Age International Publishers, Revised Second Edition – 2005.

[4] R.Rothermel , “Efficient Effective Regression Testing Using Safe Test Selection Techniques,” Ph.D Thesis, Clemson University, May, 1996.

[5] Y.Singh, A. Kaur, B.Suri, “A new technique for version-specific test case selection and prioritization for regression testing,” Journal of the CSI ,Vol. 36 No.4, pages 23-32, October-December 2006.

[6] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, “Prioritizing Test Cases for Regression Testing,” IEEE Trans. Software Eng., vol. 27, no. 10, pages 929-948, Oct. 2001.

[7] Y. Chen, D. Rosenblum, and K. Vo. TestTube, “A system for selective regression testing,” In Proceedings of the 16th International Conference on Software Engineering, pages 211-220, May 1994.

[8] K. Fischer, F. Raji, and A. Chruscicki, “A methodology for retesting modifed software,” In Proceedings of the National Telecommunications Conference B-6-3, pages 1-6, Nov. 1981.

[9] R. Gupta, M. J. Harrold, and M. Soffa, “An approach to regression testing using slicing,” In Proceedings of the Conference on Software Maintenance, pages 299-308, Nov. 1992.

[10] R. Gupta, M. J. Harrold, and M. Soffa, “An approach to regression testing using slicing,” In Proceedings of the Conference on Software Maintenance, pages 299-308, Nov. 1992.

[11] N.Mansour, and K. El-Faikh, “Simulating annealing and genetic algorithms for optimal regression testing,” Journal of Software Maintenance, Vol. 11, pages 19-34, 1999.

[12] M.J.Harrold, R.Gupta, and M.L. Soffa,“ A methodology for controlling the size of the test suite, ” ACM Transaction on Software Engineering and Methodology, pages 270-285, July 1993.

[13] H.Agrawal ,J.R. Horgan, and E.W. , Krauser, "Incremental regression testing," In: Proc. Conference on Software Maintenance, pages 348-357,1993.

[14] R.Bahsoon, N. Mansour, "Methods and metrics for selective regression testing," In Computer Systems and Applications, ACS/IEEE International Conference, pages 463-465, 2001.

[15] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Trans. Software Eng., vol. 27, no. 10, pages 929-948, Oct. 2001.

[16] S.Elbaum, Alexey G. Malishevsky, and G.Rothermel, "Test case prioritization: A family of empirical studies," IEEE Transactions on Software Engineering, vol. 28, NO.2, pages 159-182, Feb.2002.

[17] K. K. Aggrawal, Y. Singh, A. Kaur, " Code coverage based technique for prioritizing test cases for regression testing ," ACM SIGSOFT Software Engineering Notes , vol 29 Issue 5 September 2004.

[18] B.Suri, I.Mangal, V.Srivastava,"Regression Test Suite Reduction using a Hybrid Technique Based on BCO and Genetic Algorithm",Special Issue of International Journal of Computer Science and Informatics,ISSN:2231-5292,Volume II.

[19] B.Suri, S.Singhal, "Analyzing Test Case Selection and Prioritization using ACO",ACM SIGSOFT Volume 36 Issue 6, November 2011

[20] K.R.Walcott,M.L.Soa,G.M.Kapfhammer,and R.S.Roos","Time aware test suite prioritization",In Proceedings of ISSTA,pages 1-11,2006.

[21] J.Holland, "Adaption in Natural and Artificial Systems", Ann Arbor, MI: University of Michigan Press,1975.

[22] D. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning", New York,Addision Wesely, 1989.

[23] K.A.De Jong, " Analysis of Behaviour of a class of Genetic Adaptive Systems" .Phd Thesis,University of Michigan,Ann Arbor,MI, 1975.

[24] G.Vahadati, M. Yaghoubi, M.Poostchi, S.Naghibi, "A New Approach to Solve Traveling Salesman Problem Using Genetic Algorithm Based on Heuristic Crossover and Mutation Operator",IEEE 2009.

[25] P.C. Chu and J.E. Beasley, "A Genetic Algorithm for the Multidimensional Knapsack Problem", Springer, Journal of Heuristics Volume 4, Number 1, 63-86, DOI: 10.1023/A:1009642405419.

[26] D. Karaboga, B. Basturk, "A Powerful And Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm", Journal of Global Optimization, Volume:39 , Issue:3 ,pp: 459-471, Springer Netherlands, 2007.

[27] D. Karaboga, B. Basturk, "Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems", Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, Vol: 4529/2007, pp: 789- 798, Springer-Verlag, 2007, IFSA 2007.

[28] D. Karaboga, B. Basturk Akay, "Artificial Bee Colony Algorithm on Training Artificial Neural Networks, Signal Processing and Communications Applications", .SIU 2007, IEEE 15th. 11–13 June 2007, Page(s):1 - 4, 2007.

[29] Wikipedia; http://en.wikipedia.org/wiki/Bee

[30] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Technical Report-TR06, Erciyes University, Computer Engineering Department, Turkey, 2005.

[31] M.Dorigo,"Ant Colony Optimization: Artificial Ants as Computational Intelligence Technique",IEEE Computaional Intelligence Magazine,2006.