International Journal of Computer Science and Informatics

Volume 3 | Issue 1 Article 5

July 2013

Regression Test Suite Reduction using an Hybrid Technique Based on BCO And Genetic Algorithm

Bharti Suri
University School of Information Technology, Guru Gobind Singh Indraprastha University, Dwarka, Delhi-110075, India, bhartisuri@gmail.com

Isha Mangal
University School of Information Technology, Guru Gobind Singh Indraprastha University, Dwarka, Delhi-110075, India, ishamangal2006@yahoo.com

Varun Srivastava
University School of Information Technology, Guru Gobind Singh Indraprastha University, Dwarka, Delhi-110075, India, varun0621@gmail.com

Regression Test Suite Reduction using an Hybrid Technique Based on BCO And Genetic Algorithm

Bharti Suri, Isha Mangal & Varun Srivastava

University School of Information Technology, Guru Gobind Singh Indraprastha University,
Dwarka, Delhi-110075, India
E-mail : bhartisuri@gmail.com, ishamangal2006@yahoo.com, varun0621@gmail.com

Abstract - Regression testing is a maintenance activity that is performed to ensure the validity of modified software. The activity takes a lot of time to run the entire test suite and is very expensive. Thus it becomes a necessity to choose the minimum set of test cases with the ability to cover all the faults in minimum time. The paper presents a new test case reduction hybrid technique based on Genetic algorithms(GA) and bee colony optimization (BCO) .GA is an evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. BCO is a swarm intelligence algorithm. The proposed approach adopts the behavior of bees to solve the given problem. It proves to be optimistic approach which provides optimum results in minimum time.

Key words - *Regression testing; genetic algorithm(GA); beecolony optimization(BCO).*

I. INTRODUCTION

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of obsolete capabilities [1, 2]. Regression testing is performed in maintenance phase and is defined [3] as "the process of retesting the modified parts of the software and ensuring that no new errors have been introduced into previously tested code". It is an expensive activity and consumes significant amount of effort and cost. The test suits are already available from previous stages of software development life cycle (SDLC). Regression testing does not involve rerunning the entire suite but selecting a subset of it [4] that can detect all the faults. There are various regression techniques: Retest all [1, 5], Regression test selection [6-16], and test case prioritization [8, 17-19] and hybrid approaches [1,2,7]. As the presented paper is based on test suite selection; the paper discusses it in detail.

Regression Test Case Selection (RTS):

"Retest all" is very expensive technique, so regression test selection is performed to reduce the cost. The technique, instead of rerunning the whole test suite, choses a part of test suite such that cost of selecting a part of test suite is less than the cost of running the tests. RTS techniques are broadly classified into three categories [3] as in Fig 1: Coverage techniques [11], Minimization techniques and Safe techniques [3].

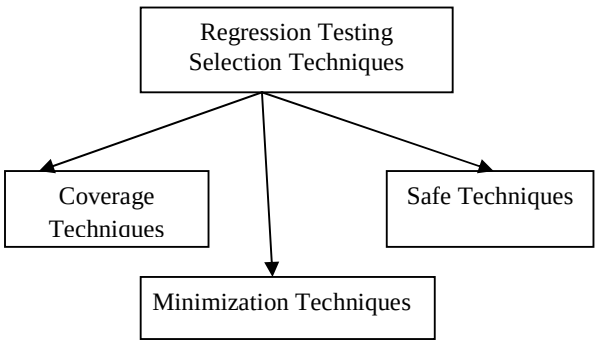

Fig. 1: Regression Testing Selection Techniques

The work is an attempt to develop an algorithm for reduction of test cases from a large test suite using genetic algorithm and bee colony optimizations that will reduce both time and effort and produce optimal results.

The paper is organized as follows. Section II pertains to related works and research in this specific area. Section III discusses how the genetic algorithm works. Section IV talks about the artificial bee colony optimization. The proposed approach has been discussed in Section V and illustrated with an example. The application of proposed approach is given in Section VI. Section VII summarizes the conclusion and the future work.

II. RELATED WORK

Several algorithms based on genetic algorithm [20, 21] and swarm intelligence [22, 23] ie.ant colony optimizations and bee colony optimizations have been proposed for test case selection and prioritization from a large test suite. Sthamer[24] and Pargas et al [25] applied GA for automatic testdata generation in his thesis. A Strategy for using GA to automate branch and fault-based Testing [26] and automatic structural testing using genetic algorithms [27] is done by Jones et al. Lin and Yeh worked on GA for automatic test data generation based on path based testing [28]. An evolutionary approach is developed to dynamic test data generation by Anastasis and Andreas [29]. Harman et al proposed an approach to reduce the input domain using search based technique [30]. In fact, the genetic algorithm is also used to generate test data automatically [31].A lot of work is done by researchers on optimization of test cases. Mala et al has developed a hybrid genetic algorithm based approach for quality improvement and optimization of test cases[32] and Eric et al analyzed the effect of fault detection of test set when its size is minimized [33]. The concept of Artificial Bee Colony algorithm was introduced by Karaboga [34-36]. Chong et al [37] applied honey bees foraging behavior model to the job scheduling problem. McCaffrey et al [38] generates pair wise test sets using a simulated bee colony algorithm. Mala et al [39] presented a new, non pheromonen based test suite optimization approach inspired by the behavior of biological bees. Dahiya et al [40] presented an ABC algorithm based approach for automatic generation of structural software tests.

III. GENETIC ALGORITHM

A genetic algorithm (GA) is an optimization technique which can be applied to various problems, including those that are NP-hard [41]. GA is adaptive search procedures which were introduced by John Holland [20] and extensively studied by Goldberg [21], De Jong [42] and many other researchers. It uses a "survival of the fittest" technique, where the best solutions survive and are varied until we get a good product [43].

To apply GA, two main requirements are to be satisfied:

(a) An encoding is used to represent a solution of the solution space, and

(b) An objective function i.e a fitness function which measures the goodness of a solution [44].

The proposed technique makes use of genetic operations. The GA process consists of the various steps as shown in fig 2.

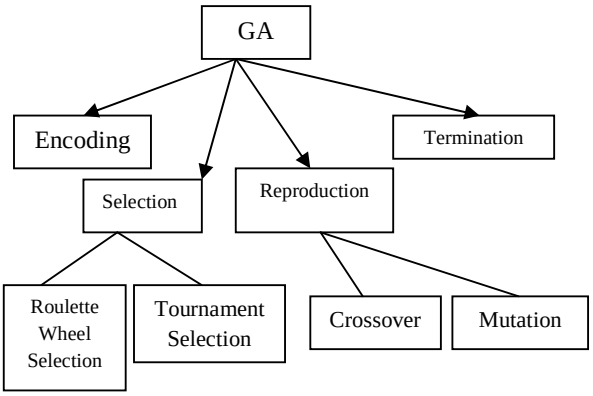

Fig. 2 : GA Architecture

Encoding is done for the solution to the problem. Using fitness-based function like roulette wheel selection and tournament selection, initial population is chosen. The second generation population of solutions is generated from first generation using genetic operators like crossover and mutation. New population will be chosen and further take part in generating the next generation. The process is repeated until a termination condition is reached (i.e. the result has been found or, fixed number of generations reached).

In the proposed technique, crossover operation is applied at the second step of GA life cycle. This is the method of merging the information units of two individuals that will produce two more new children (information units).Here cutting of the two strings at the user crossover point and swapping the two. The outcome of this process is the new population.

Take two strings and perform a 2-point crossover on them.

10111 | 00101
11111 | 00001

The new population or strings generated after applying crossover are: 1011100001 and 1111100101.

IV. ARTIFICAIL BEE COLONY OPTIMIZATION

Artificial Bee Colony (ABC) algorithm is a swarm based meta-heuristic algorithm introduced by Karaboga in 2005 [22, 23], and simulates the foraging behavior of honey bees. BCO is an optimization tool which provides a population based search procedure, where the artificial bees modify the food positions with time [45]. The main aim of the bees' is to locate the food source positions with high nectar amount [46]. The colony of bees in ABC algorithm comprises of three

groups of bees: employed bees, onlookers and scouts [45]. 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 and find a new food source again. Onlookers decided their food source depending upon the dances of employed bees.

A nector source is selected by each bee by following a nest mate whose food source has already discovered. The bees dances on the floor area(i.e.) hive, on discovery of nector sources. That is the how bees convinced their nestmates to follow them. To get nector, bees follow the dancers to one of the nectar areas. On collecting the nector they return back to their hive, relinquish the nectar to a food storer bee. 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 nestmates by dancing before the return to the food source.Differerent 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" .It is also noted that not all bees start foraging simultaneously.

V. PROPOSED APPROACH

In this paper, we proposed a new approach to reduce the cost of regression testing by test case suite reduction. 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.

The prerequisite for the proposed algorithm is a test suite 'T' of 'n' test cases. The result is subset 'S', which consists of m test cases(m<=n),such that the test cases are selected on the basis of maximum fault coverage capacity in minimum execution time.

The assumptions taken for the proposed algorithm is as follows:

- Given the original test suite, T={t1,t2……tn}.
- Set of all faults, F={f1,f2,…….fk}.
- Each test case {t1,t2,…tn} in the original test suite covers some or all the faults from 'F'.
- Zi is the execution time for running each test case ti , where. $1 \leq i \leq n$.
- Each test case will be represented in binary form. Each test case is of 'k' bits (k is the total number of faults).Each bit of the test case depends upon the capacity of detecting that fault. Starting from the leftmost bit, the bit is 1 if it detects Fault F_k else 0 and so on.
- Number of artificial bees 'B' to search through the test case space is n (number of test cases).
- It is not necessary that all bees will fly at the same time. If we have 'n' bees, then 'n/2' bees will start flying initially.
- Initially number of faults covered by bees is zero.
- While foraging, bees try to cover more faults as much as possible.
- Bees while foraging, will add the test case only if it increases its number of faults covered capacity by the set of test cases , otherwise the bee searches for the other test case .

The steps for running the proposed algorithm are:

Step 1: Initially the 'n/2' starts foraging. All the bees will not initially start foraging.

Step 2: Each bee that have started forage will choose the test case randomly.

Step 3: The bees will select test case on the basis that adding test case will increase the fault detection capacity. This can be checked by 'OR' operation of Boolean algebra. The number of 1's present in the result is the number of faults covered.

Step 4: The foraged bees will return to their hive and genetically exchange information by crossover method.

Step 5: Crossover will be performed between the bees whose total execution time is minimum and the bees with the next minimum execution time. We assume that the bees that discover a set of test case in minimum time will produce a new set of test case that will be executed in minimum time and helpful in future to predict better and optimum result.

Step 6: If the resulted test cases's 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.

Step 7: 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.

Step 8: Now again the bees will choose the test case. Repeat From step 3 to 6 till any of the bee has explored a set of test cases that can cover all the test faults.

Step 9: 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.

Step 10: As the number of iterations increase the system will give the optimum result.

Example:

Consider a test suite with test cases in it, covering a total of 10 faults. Test case selection selects a subset of test cases which will cover all the faults in minimum execution time. In this section we demonstrate the working and efficiency of proposed approach using the given example.

The regression test suite 'T' as given in Table 1, contains eight test cases {T1, T2, T3, T4, T5, T6, T7, T8}. The prerequisite for this example assumes the knowledge of the faults detected by T as shown in Table2.Test case T1 can find four faults {F1,F3,F6,F9} in seven minutes, T2 finds two faults {F2,F8 } in three minutes, and T3 finds three faults {F2, F5,F7}in 5 minutes. Test cases T4 and T5 find four and three faults in 5 and three minutes {F4,F6,F8,F9} and {F1,F6,F10} respectively. Test cases T6 and T7 find three faults in six and three minutes {F4,F5,F8} and {F1,F7,F8}.Test case T8 find two faults {F3,F10} in two minutes.

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 1: 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 2: Test cases, no. of faults covered, their execution time

The algorithm works as follows:

The binary representation of test cases is shown in Table3.The bit id is 1 (high) if it covers the respective fault starting from the leftmost position, otherwise 0.

Test Case	Binary Form
T1	1010010010
T2	0100000100
T3	0100101000
T4	0001010110
T5	1000010001
T6	0001100100
T7	1000001100
T8	0010000001

Table 3: Binary Representation of Test cases

Let number of artificial bees in this example are B :{ B1, B2, B3, B4, B5, B6, B7, B8} that is equal to the number of test cases. Initially 'n/2' i.e.4 bees (8/2=4) B1, B2, B3, B4 will forage. The number of fault covered by each bee is zero initially. Suppose the test case chosen by B1, B2, B3, and B4 is T1, T2, T3, T4 respectively as shown in Table 4.

BEES	Test Case Selected	Total Execution Time	No. of Faults Covered	Binary Representation of Fault Covered
B1	T1	7	4	1010010010
B2	T2	3	2	0100000100
B3	T3	5	3	0100101000
B4	T4	5	4	0001010110

Table 4: Test cases selected by bees initially

In II Round, bees will select those test cases that will increase its faults covering capacity. Suppose B1 selects T5.check the number of faults by OR function of Boolean algebra.

T1: 1010010010

T5: 1000010001

OR: 1010010011

So the number of faults detected by B1 by choosing {T1, T5} is 5.We will calculate this for all bees and the calculated values are shown in table 5.

BEES	Test Case Selected	Total Execution Time	No. of Faults Covered	Binary Representation of Fault Covered
B1	T1,T5	10	5	1010010011
B2	T2,T6	9	4	0101100100
B3	T3,T7	8	5	1100101100
B4	T4,T8	7	6	0011010111

Table 5: Selecting TC by bees in Round II

Now all the four bees as in Table 6 will return to their place from where they have started and genetically do a crossover operation.

BEES	**B1**	**B2**	**B3**	**B4**
Test Case Selected	T1,T5	T2,T6	T3,T7	T4,T8
Total Execution Time	10	9	8	7

Table 6: Foraged BEES and their execution time

Here, Bees B3 (ET: 8), B4 (ET:7) will be genetically crossover as their execution time is the minimum from others.

B3:T3 | T7

B4: T4 | T8

After applying crossover the new test cases produced are {T3,T8} and {T4,T7} whose total execution time is 7(5+2) and 8 (5+3) and faults covered are 5 and 6.As the execution time of new generated test set is less than the maximum(ie.10), two bees will be added B5 and B6 (as shown in Table 7 which will forage using those test cases initially.

BEES	**Test Case Selected**	**Total Execution Time**	**No. of Faults Covered**	**Binary Representation of Fault Covered**
B1	T1,T5	10	5	1010010011
B2	T2,T6	9	4	0101100100
B3	T3,T7	8	5	1100101100
B4	T4,T8	7	6	0011010111
B5	T3,T8	7	5	0110101001
B6	T4,T7	8	6	1001011110

Table 7: New Bees will be added after Round II

In Round III, the test case chosen by bees (Table 8) are as follows:

BEES	Test Case Selected	Total Execution Time	No. of Faults Covered	Binary Representation of Fault Covered
B1	T1,T5,T2	13	7	1110010111
B2	T2,T6,T5	12	7	1101110101
B3	T3,T7,T8	10	7	1110101101
B4	T4,T8,T7	10	8	1011011111
B5	T3,T8,T1	14	8	1110111011
B6	T4,T7,T3	13	8	1101111110

Table 8: Selecting TC by bees in Round III

Now all the six bees will return to their place after selecting test cases and genetically do a crossover operation for exchanging information. In this round B3 (ET: 10) and B4 (ET: 10) do a crossover operation.

B3: T3 | T7 T8

B4: T4 | T8 T7

After applying crossover the new test cases produced are {T3,T8,T7} and {T4,T7,T8} whose total execution time is 10 and 10 and faults covered are 7 and 8 .Since the results produced generate no new set of test cases so no new will forage.

Now, Round IV starts. In this Round, the test case chosen by bees in Table 9 are as follows:

BEES	**Test Case Selected**	**Total Execution Time**	**No. of Faults Covered**	**Binary Form of Fault Covered**
B1	T1,T5, T2,T6	19	9	1111110111
B2	T2,T6, T5,T3	17	8	1101111101
B3	T3,T7, T8,T4	15	10	1111111111

B4	T4,T8, T7,T6	16	9	1011111111
B5	T3,T8, T1,T2	17	9	1110111111
B6	T4,T7, T3,T8	15	10	1111111111

Table 9: Foraged BEES return hive & their execution time

As we have seen that B3and B6 has covered all the faults in 15 units. So test case set of {T3, T4, T7, T8} is the first minimum set of test case produced. To produce more optimal efficient results the above process should be iterated.

VI. APPLICATION OF PROPOSED APPROACH

Software practitioners may use the technique developed to reduce time and effort required for selection of test cases from a large test suite. This approach may start to produce better results. But as the iterations proceeds, the system will produce optimum results. This approach proved very beneficial during regression testing. The test cases selected, will cover all the faults. Hence the proposed approach may prove to be useful in real life situation.

VII. CONCLUSION AND FUTURE WORK

We have proposed test case selection approach from a large test suite using hybrid technique based on genetic algorithms and bee colony optimizations. This approach has been tested for several examples. One of these examples has been shown in this paper. The technique developed using this approach identifies and reduces the test data. The approach provides better results in the initial iteration of the whole process. It provides positive feedback and hence it can lead to better solutions in optimum time.

Issues of future research include automation of the technique and applying it on large and complex software. We also aim to compare it to ant colony optimizations algorithms and genetic algorithms.

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] B.Suri, P. Nayyar ,"Coverage Based Test Suite Augmentation Techniques-A Survey", International Journal of Advances in Engineering & Technology, May 2011, IJAET ISSN: 2231-1963.

[5] H. Leung and L. White, "Insights into regression testing," In Proceedings of the Conference on Software Maintenance, pages 60-69, Oct. 1989.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

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

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

[22] Wikipedia; http://en.wikipedia.org/wiki/ Swarm_intelligence.

[23] Scholarpedia;http://www.scholarpedia.org/ article/Artificial_bee_colony_algorithm.

[24] H.H. Sthamer, "The automatic generation of software test data using genetic algorithms, Ph.D thesis, University of Glamorgan 1996.

[25] R.P Paragas, M. Harrolg and R.Peck , "Test data generations using genetic algorithms", Software testing verification and reliability, vol.9, no4, pp263-282,1999.

[26] B .Jones, D.Eyres and H .Sthamer ,"A strategy for using genetic algorithms to automate branch and fault based testing", the computer journal , vol 41, no.2pp. 98-107,1998.

[27] B.F Jones, H.H Sthamer and D.Eyres, "Automatic structural testing using genetic algorithms", Software engineering journal, vol.11,no.5,pp.229-306, 1996.

[28] J.C. Lin, P.L. Yeh," Automatic test data generation for path testing using Gas", Department of Computer Science and Engineering, Tatung University,1999.

[29] A.Anastasis, A. S. Andreou," Automatic, evolutionary test data generation for dynamic software testing", Journal of Systems andSoftware Volume 81, Issue 11, Pages 1883-1898, November 2008.

[30] M.Harman,Y.Hassoun,K.Lakhotia,P.McMinn, J.Wegener," The impact of input domain reduction on search-based test data generation",in the proceedings of ACM SIGSOFT, ISBN: 978-1-59593-811-4,2007.

[31] Marc Roper, Iain Maclean, Andrew Brooks, James Miller and Murray Wood. Genetic Algorithms and the Automatic Generation of Test data,1995.

[32] D.J.Mala , V.Mohan, "Quality Improvement and Optimization of Test Cases-A Hybrid Genetic Algorithm Based Approach", ACM SIGSOFT ,May 2010.

[33] W.W.Eric, ,R.H.Joseph, L.Saul and Aditya P.Mathur,"Effect of Test Case Minimization of Fault Detection Effectiveness",Software pPractice and Experience,Vol.28,No.4, pp. 347-369, 1998.

[34] 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.

[35] 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.

[36] 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.

[37] C.S. Chong, M.Y.H. Low, A.I Sivakumar, K.L Gay," A Bee Colony Optimization Algorithm to Job Scheduling Simulation" In Proceedings of the Winter Conference, Washington, DC, pp. 1954-1961,2006.

[38] J.D.McCaffrey, "Generation of Pairwise Test Sets using a Genetic Algorithm", Proceedings of the 33rd IEEE International Computer Software

and Applications Conference, pp. 626-631, July 2009.

[39] D. Jeya Mala, V. Mohan, "ABC Tester - Artificial Bee Colony Based Software Test Suite Optimization Approach", Int.J. of Software Engineering, IJSE Vol.2 No.2 July 2009.

[40] S.S.Dahiya, J.k.Chhabra, S.Kumar, "Application of Artificial Bee Colony Algorithm to Software Testing", Software Engineering Conference (ASWEC), 21st Australian IEEE Conferences,2010.

[41] NP-Hard Problems", ACM SIGACT, Volume 28 Issue 2, June 1997.

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

[43] http://en.wikipedia.org/wiki/Genetic_algorithm

[44] H. Zakir Ahmed,"Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator", International Journal of Biometrics & Bioinformatics (IJBB) Volume (3),Issue (6).

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

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

◈◈◈