Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Crossbreed Algorithm for Regression Testing by Comparing Bee Colony Optimization (BCO), Genetic Algorithm (GA) and Ant Colony Optimization (ACO)

Jasjeet Kaur
M.Tech Scholar
SBBSIET College, Jalandhar, India.

Tajinder Kaur
Asst.Prof(IT)
SBBSIET College, Jalandhar, India.

Abstract - Regression testing is a branch of testing in which we test the system in each and every manner. This is done to improve the performance of the system on the basis of faults found in the region. For better efficiency we combine the test case with other suitable test cases to enhance the performance. Now in this paper we have proposed a work in which we have combined BCO with COA to perform the optimization of the system as well as we would compare it with all the previous work done. COA is an optimization algorithm in which it is tested at each and every step that how the test cases should be combined to optimize the result .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.

Keywords - Regression testing; Genetic algorithm (GA); Bee colony optimization (BCO); Ant colony optimization(ACO);Continuous orthogonal algorithm(COA)

I. INTRODUCTION

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of obsolete capabilities[16,17]. Regression testing is performed in maintenance phase and is defined[18] 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[19] that can detect all the faults. There are various regression techniques: Retest all[14,16], Regression test selection, and test case prioritization[20] and hybrid approaches[16,17] . As the presented paper is based on test suite selection; the paper discusses it in detail.

Regression Testing Techniques: Regression testing techniques are categorized as: regression test selection techniques, regression test prioritization techniques, and hybrid techniques. Regression test prioritization techniques reorder test suite with a goal to increase the effectiveness of testing in terms of achieving code coverage earlier, checking frequently used features of software, and early fault detection. Hybrid techniques combine both selection and prioritization for regression testing.

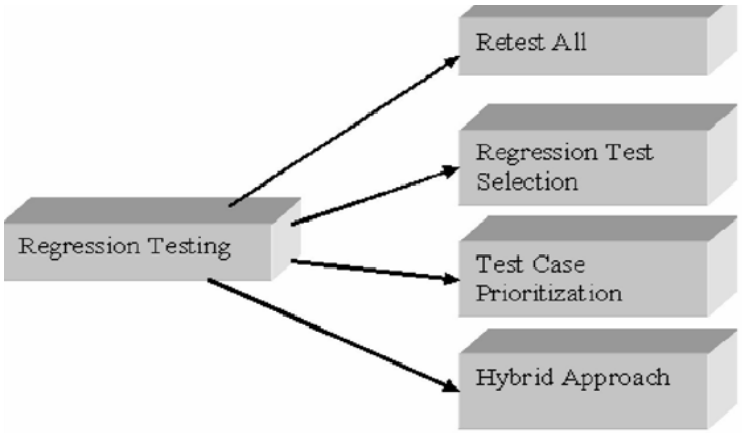

Figure 1.4: Various Types of Regression Testing Techniques

The paper is organized as follows. Sections II discuss the concept of Genetic algorithm. Section III discuss about the Swarm intelligence. Section IV describes about Bee colony optimization. Ant colony optimization is discussed in Section V. The proposed approach has been discussed in Section VI. Section VII summarizes the conclusion and future work

II. GENETIC ALGORITHMS

A genetic algorithm (GA) is an optimization technique[3] which can be applied to various problems, including those that are NP-hard.GA is adaptive search procedures which were introduced by John Holland in 1975 and extensively studied by Goldberg(1989), De Jong(1975) 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. 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.

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

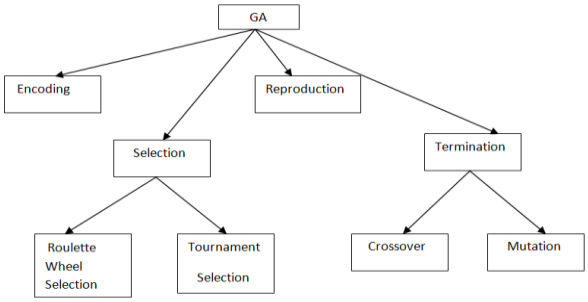

Fig 2: GA Architecture

Figure 4.1: GA Architecture[3]

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

III. SWARM INTELLIGENCE

Swarm intelligence (SI) is artificial intelligence based on the collective behavior of decentralized, self-organized systems. The expression was introduced by Gerardo Beni and Jing Wang in 1989, in the context of cellular robotic systems. SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents lead to the emergence of complex global behavior. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling.

IV. BEE COLONY OPTIMIZATION

Bee Colony Optimization is the name given to the colony formed from the mutual understanding and team work of the natural bees in the process of foraging food. It is a fact that all the creatures in this earth follow one or the other mechanism/process to find food sources that suite them. The build-up of honey bee comb and its management is a classic example of teamwork, synchronization experience and co-ordination. The way the bees find, build and maintain their comb and hive is remarkable. These are the factors which have given rise to interest of researchers to study this system and find solutions to their problems. In a natural honey bee hive there are a variety of bees with specific role(s) to play.

V. ANT COLONY OPTIMIZATION

Ant colony optimization is a technique for optimization that was introduced in the early 1990's. The inspiring source of ant colony optimization is the foraging behavior of real ant colonies. This behavior is exploited in artificial ant colonies for the search of approximate solutions to discrete optimization problems, to continuous optimization problems, and to important problems in telecommunications, such as routing and load balancing. First, we deal with the biological inspiration of ant

colony optimization algorithms. We show how this biological inspiration can be transferred into an algorithm for discrete optimization. Then, we outline ant colony optimization in more general terms in the context of discrete optimization, and present some of the nowadays best performing ant colony optimization variants. After summarizing some important theoretical results, we demonstrate how ant colony optimization can be applied to continuous optimization problems. Finally, we provide examples of an interesting recent research direction: The hybridization with more classical techniques from artificial intelligence and operations research.

VI. CONTINUOUS ORTHOGONAL ALGORITHM

COA along with ACO has been implemented to enhance the probability of the search areas of the algorithm so that we can get the maximum number of faults in each section. Also it has been used to perform that which test case matches with another test case so that we can find out the optimal result. The pheromone deposit mechanism of COAC is to enable ants to search for solutions collaboratively and effectively. By using an orthogonal design method, ants in the feasible domain can explore their chosen regions rapidly and efficiently, with enhanced global search capability and accuracy.

VII. PROPOSED APPROACH

The proposed technique is based on concepts of GA,BCO and ACO. The mechanism of COA is best suite with the concepts BCO and ACO to solve the real time problems .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. Bees will start foraging with randomly selected test cases. 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. ACO is used to improve the efficiency of the system. At last COA will help to search the sections till the time there is any margin of finding any fault it means that COA will decide which test case has the possibility to be combined with another test case so that we can search out maximum regions from where the faults can covered.

The proposed algorithm has 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:

i. Given the original test suite, T={t1,t2……tn}.

ii. Set of all faults, F={f1,f2,…….fk}.

iii. Each test case {t1,t2,…tn} in the original test suite covers some or all the faults from 'F'.

iv. Zi is the execution time for running each test case ti , where. 1_ i _ n.

v. 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 Fk else 0 and so on.

vi. Number of artificial bees 'B' to search through the test case space is n (number of test cases).

vii. It is not necessary that all bees will fly at the same time.

Viii. While foraging, bees try to cover more faults as much as possible.

ix. 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 .

x. The test cases are combined according to similarity in their properties. The data which is produced by BCO is transferred to the ACO.

X. Finally the COA (Continuous orthogonal algorithm) executes the test cases to a number generate by its algorithm , the number can be called as the threshold frequency of the region in which the test case has to be executed .This threshold value is a very critical number because it has to be decided very carefully. If you test the data again and again, it may lead to a time delay into the output as well as it may damage the data.

The steps for running the proposed algorithm are as follows:

Step 1: At the first stage, random test cases have been created to check out the system performance.

Step 2: Now bees will start foraging .each bee that have started forage will choose the test case randomly.

Step 3: The bees will select test cases on the basis of that addition of test cases will increase the fault detection capability or not. This can checked by the 'OR' operation of Boolean algebra

Step 4: The foraged bees will return to their hive and genetically exchange the information by using the crossover method. The data produced by the Bees using BCO will transfer to the Ants so that ACO can proceed.

Step 5: Now the bees whose execution time is minimum are selected for the crossover operation. The execution time is calculated by COA.

Step 6: Now the combinations of test cases are made to cover the maximum number of faults. These combinations are made by similarities in their properties.

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 efficiency will be improved and 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, T9,T10}. The prerequisite for this example assumes the knowledge of the faults detected by T as shown in Table2.Test case T1 can find six faults {F1,F3,F4,F5,F6,F7} in nine minutes, T2 finds four faults {F2,F3,F4,F8 } in three minutes, and T3 finds three faults {F2, F5,F7} in five minutes. Test cases T4 and T5 find four and five faults in five and three minutes {F4,F6,F8,F9} and {F1,F2,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 and T9 find two and five faults in two minutes {F3,F10}. Test case T10 find 6 faults {F1, F3,F4,F5,F6,F10} in two minutes.

Table 1: Sample data

Test case	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
T1	X		X	X	X	X	X			
T2		X	X	X				X		
T3		X			X		X			
T4				X		X		X	X	
T5	X	X		X		X				X
T6				X	X			X		
T7	X						X	X		
T8			X							X
T9			X							X
T10	X		X	X	X	X				X

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

Test case	No. of Faults Covered	Execution Time(ET)
T1	6	9
T2	4	3
T3	3	5
T4	4	5
T5	5	3
T6	3	6
T7	3	3
T8	2	2
T9	5	2
T10	6	2

a) Results using proposed method:

Now the algorithm will start its working as follows:

The binary representation of test cases is shown in Table 3.

Table 3: Binary representation of test cases

Test Case	Binary Form
T1	1010010010
T2	0100000100
T3	0100101000
T4	0001010001
T5	1000010001
T6	1001010001
T7	1000001100
T8	0010000001
T9	0010001101
T10	1110000001

In the below table 4 there are set of bees {B1,B2,B3,B4,B5,B6},and set of test cases {T1,T2,T3,T4,T5,T6}. test cases are chosen by the bees. here the B1 section executed for 9 seconds and number of faults covered by this is 6.B2 section executed for 3 seconds and the no of faults covered is 4 and so on.

Table 4: Test cases selected by bees initially.Round I

Bees	Test case selected	Execution Time(ET)	No. of Faults Covered	Binary representation of faults covered
B1	T1	9	6	1010010010
B2	T2	3	4	0100000100
B3	T3	5	3	0100101000
B4	T4	5	4	0001010110
B5	T5	3	5	0001010110
B6	T6	6	3	0001010110

Now in next table for more rigorous testing of the data, the test cases are required to combine with some other test cases.

Table 5: Combination of test cases using COA. Round II

Bees	Test case selected	Execution Time(ET)	No. of Faults Covered	Binary representation of faults covered
B1	T1,T9	9	6	1010010011
B2	T2,T10	3	4	0101100100
B3	T3,T7	5	3	1100101100
B4	T4,T8	5	4	0011010111
B5	T5,T6	3	5	0110101001

The combination of the test cases has been done on the basis of the features of the test cases. If T1 has been combined with T9, it means that the basis extraction properties of T1 and T9 are approximately matching but it does not mean at that T1 and T9 are same in all manners. If T1 gets combined with T9 we do get number of faults as 6 although the system has made a run for the same time. The result can conclude two things. Either T1 was alone sufficient enough to calculate the faults or number of faults in that region is very limited and there is no margin to find anymore fault in that particular bee area.

Table 6: Faults covered by combining other test cases

Bees	Test case selected	Execution Time(ET)	No. of Faults Covered	Binary representation of faults covered
B1	T1,T9	9	4	1010010011
B2	T2,T5	3	3	0101100100
B3	T3,T7	5	1	1100101100
B4	T4,T8	5	2	0011010111
B5	T5,T6	3	2	0011010111

By not keeping any doubt in mind that whether more faults can be detected or not, the test cases are combined with some other test cases also, but result goes down in such a manner. Hence proven that for now previous test scenario is better one in comparison to the current test scenario.
Now in table 7 Now put the test cases in a group of three test cases

Table 7: selecting group of test cases by bees. Round III

Bees	Test case selected	Execution Time(ET)	No. of Faults Covered	Binary representation of faults covered
B1	T1,T5,T9	9	6	1010010011
B2	T2,T6,T10	3	4	0101100100
B3	T3,T7,T8	5	3	1100101100
B4	T4,T8,T7	5	4	0011010111
B5	T5,T6,T3	3	5	0110101001

Table 8:Foraged bees and their execution time.Round IV

Bees	Test case selected	Execution Time(ET)	No. of Faults Covered	Binary representation of faults covered
B1	T1,T5, T2,T6	13	9	1111110111
B2	T2,T6, T5,T3	12	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

In table 9 Foraged BEES and their execution time

Table 9: Foraged BEES and their execution time

BEES	B1	B2	B3	B4
Test case selected	T1,T5	T2,T6	T3,T7	T4,T8
Total Execution Time	9	3	5	5

b) Final Results

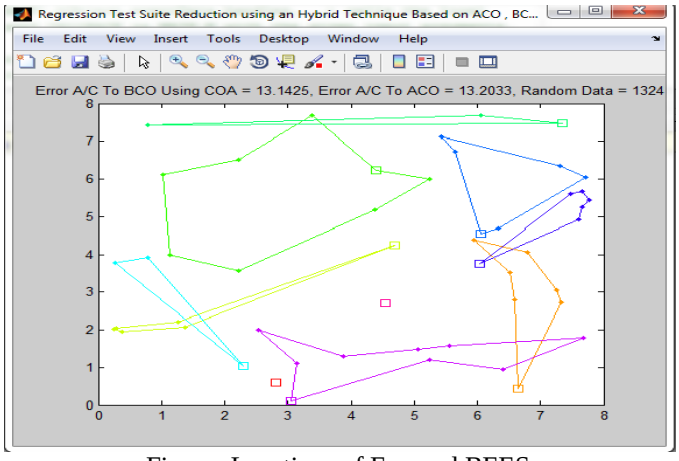

Figure: Locations of Foraged BEES

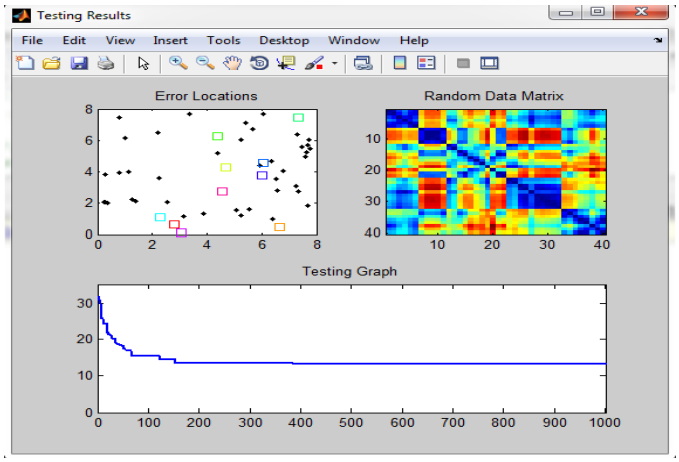

Figure: Testing Results

VIII. Conclusion And Future Work

In our proposed work we have combined BCO with COA to perform the optimization of the system as well as we would compare it with all the previous work done. COA is an optimization algorithm in which it is tested at each and every step that how the test cases should be combined to optimize the result. In Future, combination of BCO or ACO with the Bacterial foraging optimization (BFO) method of neural, if used consciously can be useful in overcoming the challenges faced by RT. The two Optimization techniques BCO and ACO can be combined with BFO to give appreciable benefits to software industry in terms of reduce the cost of software testing as compared to the traditional testing methods.

References

[1]. S. Raju and G. V. Uma *" Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm"* ISSN 1450-216X Vol.74 No.3 (2012), pp. 389-402 EuroJournals Publishing, Inc. 2012

[2]. Ali M. Alakeel"A Fuzzy Test Cases Prioritization Technique for Regression Testing Programs with Assertions" 2012

[3]. Bharti Suri, Isha Mangal & Varun Srivastava "Regression Test Suite Reduction using an Hybrid Technique Based on BCO And Genetic Algorithm". ISSN (PRINT) : 2231–5292, Vol.- II, Issue-1, 2 2011.

[4]. Mazeiar Salehie, Sen Li, Ladan Tahvildari , Rozita Dara, Shimin Li, Mark Moore "Prioritizing Requirements-Based Regression Test Cases: A Goal-Driven Practice" 15th European Conference on Software Maintenance and Reengineering IEEE 2011.

[5]. Hagai Cibulski, Amiram Yehudai " Regression Test Selection Techniques for Test-Driven Development" Published in: · Proceeding ICSTW '11 Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops.

[6]. Sheng Huang, Zhong Jie Li, Ying Liu, Jun Zhu **"**Regression Testing as a Service" Published in Proceeding SRII '11 Proceedings of the 2011 Annual SRII Global Conference IEEE Computer Society Washington, DC, USA ©2011

[7]. Arvinder Kaur and Shivangi Goyal "A Survey on the Applications of Bee Colony Optimization Techniques" ISSN : 0975-3397 Vol. 3 No. 8 August 2011

[8]. "HwaYou Hsu and Alessandro Orso""MINTS: A General Framework and Tool for Supporting Testsuite Minimization""Vol 5 2010".
[9]. Praveen Ranjan Srivastava"TEST CASE PRIORITIZATION" Journal of Theoretical and Applied Information Technology 2008".
[10]. "S.Roongruangsuwan et all: ""TEST CASE PRIORITIZATION TECHNIQUES" Journal of Theoretical and Applied Information Technology 2008".
[11]. S. Yoo, M. Harman"Regression Testing Minimisation, Selection and Prioritisation : A Survey""Vol5.24 September 2007". IEEE Transactions on Software Engineering, vol. 33, n. 4, pp. 225-237 ,
[12]. R. Pressman. Software Engineering: A Practitioner's Approach. McGraw-Hill, New York, 2002.
[13]. G. Kapfhammer. The Computer Science Handbook, chapter on Software testing. CRC Press, Boca Raton,FL, 2nd edition, 2004.
[14]. H. Leung and L. White. Insights into regression testing. In *Proceedings of the Conference on Software Maintenance*, pages 60–69, 1989.
[15]. G. Rothermel and M. Harrold. Selecting tests and identifying test coverage requirements for modified software. In *Proceedings of the International Symposium on Software Testing and Analysis*, pages 169–184, August 1994.
[16]. G.Duggal, B.Suri,"Understanding Regression Testing Techniques", COIT, 2008, India.
[17]. 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.
[18]. K.K.Aggarwal & Yogesh Singh, "Software Engineering Programs Documentation, Operating Procedures," New Age International Publishers, Revised Second Edition – 2005.
[19]. 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.
[20]. G. Rothermel, R.H. Untch, C. Chu, and M.J.Harrold, "Prioritizing Test Cases for RegressionTesting," IEEE Trans. Software Eng., vol. 27, no.10, pages 929-948, Oct. 2001.

