Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Integrated Approach to Test Suite Selection Using ACO and Genetic Algorithm

Gurinder Singh[1], Dinesh Gupta[2]
M. Tech Student Deptt. of CSE.
Asst Prof. Deppt. of Information Technology, India

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. This research paper presents a new test case reduction hybrid technique based on Genetic algorithms(GA) and Ant colony optimization (ACO) .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. ACO is a swarm intelligence algorithm. The proposed approach adopts the behaviour of ants 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); Ant colony optimization (ACO).

I. INTRODUCTION

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of obsolete capabilities 00. Regression testing is performed in maintenance phase and is defined 0 as "the process of retesting the modified parts of the software". It is an expensive activity and consumes significant amount of effort and cost 0. The test suits 0 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 0 that can detect all the faults. There are various regression techniques: Retest all 0, Regression test selection 0, and test case prioritization 0 and hybrid approaches 00. As this presented paper is based on test suite selection and discusses the proposed technique 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, chooses 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 0 as in Fig 1: Coverage techniques 0, Minimization techniques and Safe techniques 00.

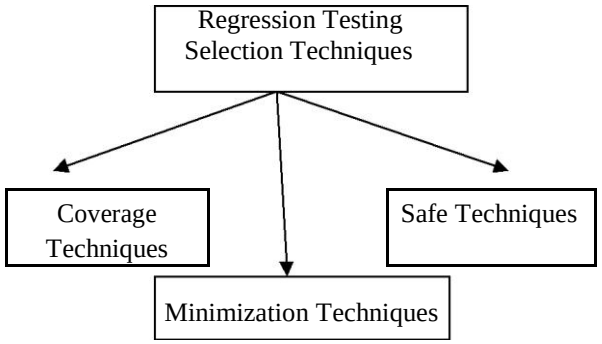

Fig. 1: Regression Testing Selection Techniques

The work is an attempt to develop an algorithm for reduction of test cases from a large test suite using genetic algorithm and Ant colony optimizations that will reduce both time and effort and produce optimal results 0. 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 discuss about the artificial ant colony optimization. The proposed approach has been discussed in Section

V and Section VI. Section VII summarizes the conclusion and the future work.

II. Related Work

Several algorithms based on genetic algorithm 0 and swarm intelligence 0 i.e. ant colony optimizations and bee colony optimizations have been proposed for test case selection and prioritization from a large test suite. A Strategy for using GA to automate branch and fault-based Testing 0 and automatic structural testing using genetic algorithms 0 is done to reduce the input domain using search based technique 0. The algorithms based on genetic algorithm 00 and swarm intelligence 00 ie.ant colony optimizations and bee colony optimizations have been proposed for test case selection and prioritization from a large test suite. Sthamer 0 and Pargas et al 0 applied GA for automatic test data generation in their thesis. A Strategy for using GA to automate branch and fault-based Testing 0 and automatic structural testing using genetic algorithms 0 is done by Jones et al. Lin and Yeh worked on GA for automatic test data generation based on path based testing 0. The concept of artificial ant Colony algorithm was introduced by Marco Dorigo. The Ant colony optimization algorithm is a probabilistic technique for solving computational problems which can be used for finding good paths through graphs.

III. Genetic Algorithm

A genetic algorithm (GA) is an optimization technique which can be applied to various problems, including those that are NP-hard 0. GA is adaptive search procedures 0 which were introduced by John Holland 0 and extensively studied by Goldberg 0, De Jong 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 2. Encoding is done for the solution to the problem. Using fitness-based function 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. Ant Colony Optimization

Ant colony optimization is a meta-heuristic technique that uses artificial ants to find solutions to combinatorial optimization problems 0. ACO is based on the behaviour of real ants and possesses enhanced abilities such as memory of past actions and knowledge about the distance to other locations. As regression testing is any type of testing that seeks to uncover new type of bugs in software 00. In nature, an individual ant is unable to communicate or effectively hunt for food, but as a group, ants possess the ability to solve complex problems and successfully find and collect food for their colony. The idea is that good path are shorter, thus ants can travel these path faster and there will be more pheromone on these path. Ants find and collect food for their colony. Ants communicate using a chemical substance pheromone. As an ant travels, it deposits a constant amount of pheromone that other ants can follow. Each ant moves in a somewhat random fashion, but when an ant encounters a pheromone trail, it must decide whether to follow it. If it follows the trail, the ant's own pheromone reinforces the existing trail, and the increase in pheromone increases the probability of the next ant selecting the path. Therefore, the more ants that travel on a path, the more attractive the path becomes for subsequent ants. ACO is based on the behaviour of real ants and possesses enhanced abilities such as memory of past actions and knowledge about the distance to other locations. In nature, an individual ant is unable to communicate or effectively hunt for food, but as a group, ants possess the ability to solve complex problems and successfully find and collect food for their colony. In artificial term, the optimization method uses this intuition in the following way. Ants construct solution by making a number of decisions probabilistically. In the begging there is no constructed solution, pheromone information guides other ants in their decision making. The idea is that good path are shorter, thus ants can travel these path faster and there will be more pheromone on these path. However over time

pheromone on trail that is not reinforced will evaporate. Thus, over time shorter path will be reflected in pheromone information.

The ACO algorithm mainly consists of the iteration of two steps:

a) Generation of solution by ants according to private and pheromone information.

b) Updating of the pheromone information.

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 ACO 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 ants are used as agents who explore the minimum set of test cases. Half of the ants will initially start foraging with randomly selected test cases. Now ants will add new test cases on her explored path if adding of a test case increases its fault detection capacity. The crossover operation is used to exchange the information. The new set of test case produced after crossover is used by new ants to forage. The process is repeated till any of the ants has discovered a set of test cases that covers all faults detection.

Hybrid algorithm is proposed to reduce the cost of regression testing by test suite reduction and the output of the ACO is given as an input to the GA and proposed algorithm as follow: -

ALGORITHM:-

-Initially 'n/2' ants are selected for foraging.

-Repeat (until termination condition satisfied)

- Each ant select test case randomly and based on fitness function they will select those test case that increases the fault detecting capability.
- The foraged ant returned and performed genetic algorithm operation i.e. selection, crossover and mutation.
- If resulted test case total execution time is less than maximum-time, then the new ants will forage with that test case in next iteration cycle.

In the proposed algorithm, the ACO is used to find shortest path with help of distance value, the output of ACO is given as input to the GA algorithm. The set of path are obtained during the ACO process are input to GA and genetic algorithm undergoes the selection, crossover and mutation process and it give the result. The result contain only single path which is optimal among the shortest path.

The various operations performed by Genetic algorithm are:-

Selection:-Selection operator also known as reproduction operator. It selects individual genes from the population by a probability.

Crossover:- It selects two individuals by the crossover probability p and Crossover or recombination operator combines sub parts of two parent chromosomes. Crossover is mainly of two types namely single point crossover and multipoint crossover

Mutation:- It may be possible that crossover operation may produce degenerate population. In order to undo this, mutation operation is performed. Mutation operation can be inversion, insertion, reciprocal exchange or others.

GA is well suited for solving problems where solution space is huge and time taken to search exhaustively is very high. Another advantage of genetic algorithm is that it has ability to solve problems with no previous knowledge. For this reason we hybrid ACO and GA to find shortest path for generation of test cases.

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. 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.

The 'X' symbol represents the ability of each test case to cover fault throughout the software code. The goal of our research

work is to select those test cases from test suite that cover all fault of the software with minimum execution time. For generation of result we consider 10 test cases that cover 10 type of fault in software product.

Table1: Sample data of test cases.

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

In this paper, hybrid technique is proposed which accept input data which is collected from previous execution of same software product and generate table in matrix form and encoding of this data is done because genetic algorithm accept data in binary form.

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

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: Binary Representation of Test cases

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

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.

Let number of artificial ants in this example are A :{ A1, A2, A3, A4, A5, A6, A7, A8} that is equal to the number of test cases. Initially 'n/2' i.e.4 ants (8/2=4) A1, A2, A3, A4 will forage.

Table 4: Test cases selected by ants initially

Ants	Test Case Selected	Total Execution Time	No. Of Faults Coveredion	Binary Representation of Fault Covered
A1	T1	7	4	1010010010
A2	T2	3	2	0100000100
A3	T3	5	3	0100101000
A4	T4	5	4	0001010110

In II Round, ants will select those test cases that will increase its faults covering capacity. Suppose A1 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 A1 by choosing {T1, T5} is 5.We will calculate this for all Ants and the calculated values are shown in table 5.

Table 5: Selecting TC by ants in Round II

Ants	Test Case Selected	Total Execution Time	No. of Faults Covered	Binary Representation of Fault Covered
A1	T1,T5	10	5	1010010011
A2	T2,T6	9	4	0101100100
A3	T3,T7	8	5	1100101100
A4	T4,T8	7	6	0011010111

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

Table 6: Foraged Ants and their execution time

Ants	**A1**	**A2**	**A3**	**A4**
Test Case Selected	T1,T5	T2,T6	T3,T7	T4,T8
Total Execution Time	10	9	8	7

Here, Ants A3 (ET: 8), A4 (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 Ants will be added B5 and B6.

Table 7: New Ants will be added after Round II

Ants	**Test Case Selected**	**Total Execution Time**	**No. of Faults Covered**	**Binary Representation of Fault Covered**
A1	T1,T5	10	5	1010010011
A2	T2,T6	9	4	0101100100
A3	T3,T7	8	5	1100101100
A4	T4,T8	7	6	0011010111
A5	T3,T8	7	5	0110101001
A6	T4,T7	8	6	1001011110

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

Ants	Test Case Selected	Total Execution Time	No. of Faults Covered	Binary Representation of Fault Covered
A1	T1,T5,T2	13	7	1110010111
A2	T2,T6,T5	12	7	1101110101
A3	T3,T7,T8	10	7	1110101101
A4	T4,T8,T7	10	8	1011011111
A5	T3,T8,T1	14	8	1110111011
A6	T4,T7,T3	13	8	1101111110

In this round A3 (ET: 10) and A4 (ET: 10) do a crossover operation.
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

Ants	Test Case Selected	Total Execution Time	No. Of Faults Covered	Binary Form of Fault Covered
A1	T1,T5, T2,T6	19	9	1111110111
A2	T2,T6, T5,T3	17	8	1101111101
A3	T3,T7, T8,T4	15	10	1111111111
A4	T4,T8, T7,T6	16	9	1011111111
A5	T3,T8, T1,T2	17	9	1110111111
A6	T4,T7, T3,T8	15	10	1111111111

Table 9: Foraged Ants return with their execution Time

Chosen by ants in Table 9 are as shown above:
As we have seen that A3and A6 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 this technique 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 is very beneficial during regression testing because test cases selected from test suite 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 Ant 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 data. We also aim to compare it to Bee colony optimizations algorithms and other hybrid algorithms.

References:-

[1] G.Duggal,B.Suri,"Understanding RegressionTesting 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] R.Kavitha,V.R.Kavitha and Dr.N.Suresh Kumar ,"Requirement Based Test Case Prioritization," Communication Control and Computing Technology IEEE, pp.826-829, 9Oct.2010

[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] Gregg Rothermel, Roland H. Untch and Chengyun Chu and Mary Jean Harrold, "Test case prioritization: An Empirical Study", IEEE international Conference on Software Maintenance, pp.179-188, Dec.1999.

[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] Bharti suri and Sheta Singhal, "Implementing Ant Colony Optimization for test case selection and prioritization" International Journal on Computer Science and Engineering, vol.3, pp.1924-1932, 5May2011.

[16] 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.

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

[18] 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.

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

[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] 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.

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

[24] 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.

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

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

[29] 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.

[30] 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,pages159-182, Feb.2002.

