© July 2016 | IJIRT | Volume 3 Issue 2 | ISSN: 2349-6002

Hybrid Approach for Optimizing Test Suite Based on GA & ACO

Khushboo Arora[1], Mamta Arora[2]
[1]*M.Tech(CSE),Scholar,*
[2]*Asst. Professor, CST Dept.,*
Manav Rachna Unviserity, Faridabad, India

Abstract- **Nowadays, Software testing is the most important part of a successful software product. It is a process with the intent of detecting as many errors in the software process. Software testing takes an input which executes the process and then produces an output. This output mainly depends upon the software testing. The quality and stability of a software is analyzed using software testing and is achieved by suitable test suite. Manual testing is a very difficult and expensive process and also takes a lot of time. The main problem of manual testing is the problem of code coverage, which is not performed at a regular interval. Thus there is a necessity to choose the best and minimized test suite which maximizes the fault coverage in minimum time. The paper presents a new hybrid approach for optimizing the software test suite by combining two main algorithms: Genetic algorithm and Ant Colony Optimization. Genetic algorithm, an optimization algorithm is based on natural evolution, which optimizes the solutions using different operators such as selection, crossover and mutation whereas Ant Colony Optimization algorithm is a meta-heuristic technique. The proposed methodology adopts the behaviour of ants and applies some genetic operator i.e. crossover operator to solve a problem. The paper also provides a comparison of the above hybrid technique with Genetic Algorithm and Ant Colony Optimization based on the number of test cases.**

Index Terms– **Ant Colony Optimization, Genetic Algorithm, Optimization, Software Testing, Test Suite.**

NOMENCLATURE

ACO	Ant Colony Optimization
GA	Genetic Algorithm

I. INTRODUCTION

Software testing is a most important feature of software engineering. It is a process of identifying errors as soon as possible. Software testing also detects the difference between the input which is given for testing and the output. Generally, the essential role of software is to check the quality of the given software product.

Software testing is a process of executing software with the intent of finding errors. It is known as one of the most time consuming and costly phases in software development life cycle(SDLC). Software testing also consists of verification and validation phase. In software development large amount of time is spent in doing software testing. So the main goal of software testing is to produce a set of minimum test cases which covers maximum faults in minimum time.

The given paper is organized in the following manner. Section II describes genetic algorithm and how it works. Section III talks about Ant colony optimization algorithm. Section IV discusses the various related work and research done in the given area. The proposed methodology for hybrid approach is explained in Section V. Section VI summarizes the work in conclusion and future scope.

II. GENETIC ALGORITHM

Genetic Algorithm was provided through John Holland [4] alongside his acquaintances and understudies in the mid-Nineteen Seventies. Genetic relies on the likelihood of survival of the fittest. It is often called growth process which is thoroughly contemplating common development.

Genetic Algorithm is used as a computational strategy, for characteristic determination and is utilized to reproduce the development forms. The calculation begins with haphazardly selecting answer for the underlying populace. GA takes after a succession as first creating beginning populace, assessing the populace, choice, crossover, transformation lastly re-produce the populace. GA then chooses the fittest arrangement from the given arrangements and applies hybrid and change to produce new era. The fundamental thought of GA is to deliver another era superior to the past era. This technique proceeds until a halting rule is met.

Genetic Algorithm has been with success accustomed automates the generation of check knowledge. It will defeat random search to find solutions to complicated issues. The execution of GA begins with a group of random initial population sampled for a selected downside domain.

The progression of algorithm is shown below:

Step 1: Generate irregular answers for making initial populace.

Step 2: Evaluate fitness of every arrangement in the populace.

Step 3: Repeat the accompanying method until certain foundation is met.

a. In light of the wellness esteem, select any two arrangements with higher wellness esteem.

b. Keeping in mind the crossover operator, new off springs are generated. In the event that no hybrid was performed, the new posterity created is a precise of guardians.

c. Mutation, mutate the new posterity created.

Step 4: If the condition is fulfilled then stop the system and return the arrangement which is best in current populace

Step 5: Now do a reversal to step 2.

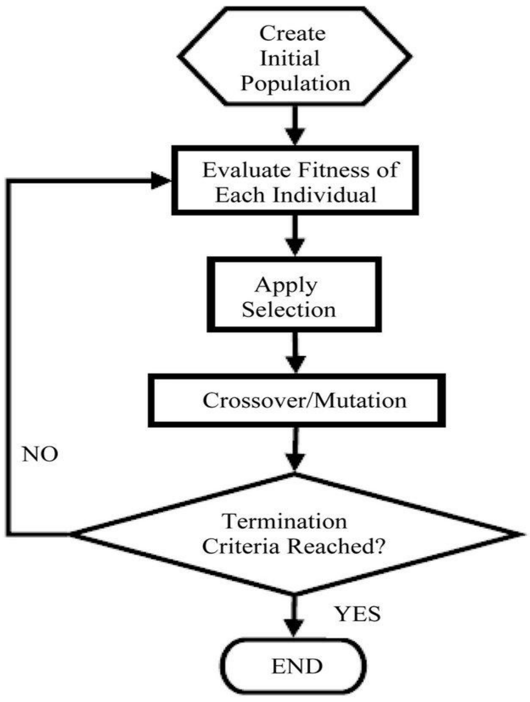

Create Initial Population
Evaluate Fitness of Each Individual
Apply Selection
Crossover/Mutation
NO
Termination Criteria Reached?
YES
END

Fig1. Flowchart of Genetic Algorithm [5]

III. ANT COLONY OPTIMIZATION

Ant colony optimization, a meta heuristic calculation, was presented by Macro Dorigo. It finds an ideal arrangement over a discrete pursuit space and follows the behaviour of ants.

The ants work in gatherings and investigate the way in quest for their sustenance and locate the most limited way to reach there. At first, ants arbitrarily move around their encompassing looking for nourishment. While moving in the encompassing they lay some kind of substance on the ground which is known as Pheromone. At the point when a subterranean insect finds the nourishment it retreats to its home by filling the ground again with that pheromone trails. Because of these pheromone trails different ants know which way to take after.

The way which contains the higher measure of pheromone trails on the ground is picked by the ants. Bigger the centralization of pheromone trail on the way more is the likelihood of picking that way by a subterranean insect.

The progression of algorithm is:

```
procedure ACO metaheuristics
        ScheduleActivities
                ManageAntActivity()
                EvaporatePheromone()
                DaemonActions() {optional}
                local search, elitism
        end ScheduleActivities
end ACO metaheuristics
```

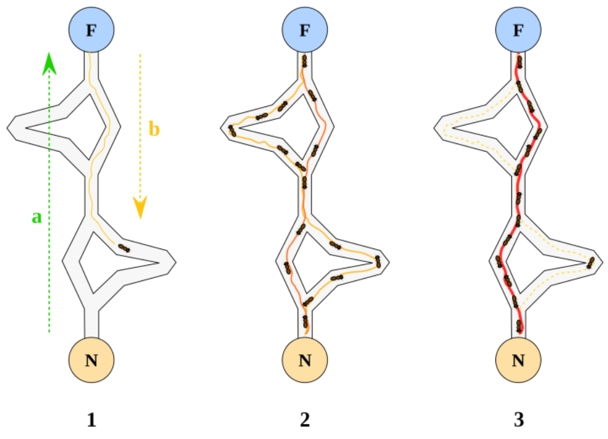


Fig2. Ant Colony Optimization Algorithm [6]

IV. RELATED WORK

A few calculations in view of hereditary calculation [15] and swarm knowledge ie.ant colony optimization and honey bee optimization have been proposed for experiment determination and prioritization from a substantial test suite.

A developmental methodology is produced to element test information era by Anastasis and Andreas [16]. Harman et al proposed a way to deal with lessen the info area utilizing search based method [4]

Kevilienuo Kire and Neha Malhotra[1] produced a testing framework by utilizing manufactured Intelligent procedures. Software Testing is a basic issue in programming improvement and support for expanding the quality and unwavering quality of the programming.

Regression testing[10] is usually performed by running some, or all, of the test cases created to test modifications in previous versions of the software. Many techniques have been reported on how to select regression tests so that the number of test cases does not grow too large as the software evolves. Our proposed hybrid technique combines modification, minimization and prioritization-based selection using a list of source code changes and the execution traces from test cases run on previous versions.

V. PROPOSED METHODOLOGY

In this paper, we proposed another way to deal with diminish the expense of regression testing by test suite decrease. The paper also gives the comparison with Hybrid approach and individual Genetic and Ant colony optimization algorithm based on the number of test cases.

The proposed procedure depends on ideas of combining GA and ACO into a hybrid approach. The procedure chooses the arrangement of experiment from the accessible test suite that will cover all the issues identified before in least execution time. Here ants are utilized as operators who investigate the base arrangement of experiments. Half of the ants will at first begin scrounging with arbitrarily chose test cases. Presently ants will include new experiments her investigated way if including of an experiment expands its fault detection limit.

The crossover operation is utilized to trade the data. The new arrangement of experiment created after crossover is utilized by new ants to search. The procedure is rehashed till any of the ants has found an arrangement of experiments that covers all faults discovery.

The essential for the proposed calculation is a test suite "T" of "n" experiments. The outcome is subset "S", which comprises of m test cases (m<=n), such that the experiments are chosen on the premise of most fault coverage capacity in least execution time.

The progressions for running the proposed algorithm are:

Step 1: Initially the 'n/2' ants begin searching. Not all the ants begin searching.

Step 2: Each ant that have begun search will pick the test cases arbitrarily.

Step 3: The ants will choose test cases on the premise that including those test cases will expand the fault detection limit. This can be checked by "OR" operation of Boolean polynomial math. The quantity of 1's available in the outcome is the quantity of faults covered.

Step 4: The scavenged ants will come back and genetically trade data by crossover technique.

Step 5: Crossover will be performed between the ants whose aggregate execution time is least and the ants with the following least execution time. We expect that the ants that find an arrangement of test cases in least time will deliver another arrangement of test case that will be executed in least time and supportive in future to foresee better also, ideal result.

Step 6: If the came about test cases aggregate execution time is not exactly the most extreme execution time accessible subset of test cases, the new ants will scavenge utilizing those subset of test cases as its underlying way.

Step 7: If the outcomes delivered after crossover does not create new test cases or test cases which are pointless, then they won't be considered. On the off chance that both test sets created after crossover won't deliver new set, no new ants will scrounge.

Step 8: Now again the ants will pick the test cases. Repeat from step 3 to 6 till any of the ants has investigated an arrangement of test cases that can cover all the test faults.

Step 9: As soon as the minimum arrangement of test cases are created, ants go back to their home so alternate ants can take after this test way.

Step 10: As the quantity of emphases expand the framework will give the ideal result.

VI. RESULT ANALYSIS

We have considered a case study of pen with the following requirements:

Table 2. Table contains requirements that our required for the case study

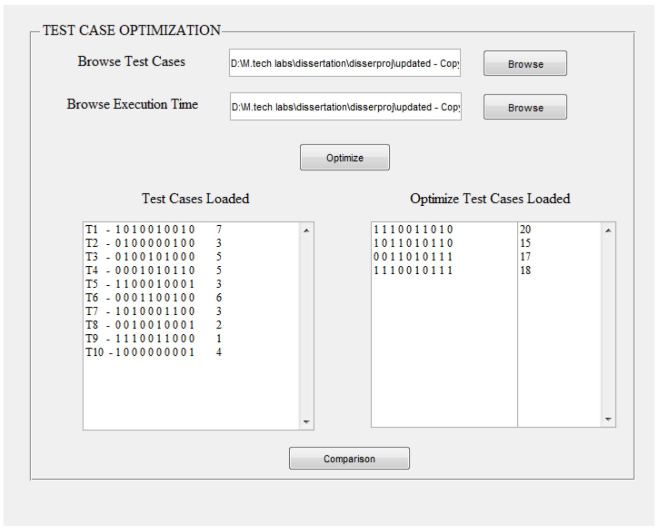
S.No.	Requirement
R1	All parts of the pen are fitting properly and no loose fitting.
R2	Size and shape should be confirmable for writing.
R3	The grip on the pen is superior.
R4	Pen is writing smoothly with continuous and not breaking while writing.
R5	Pen is writing on the page properly.
R6	The dimension of the pen as per mentioned in the requirement.
R7	Pen is usable for similar refills of different brands.
R8	Ink on the paper is belongs with the similar color as what we see in the refill.
R9	How much long can able to write with the single refill of pen.
R10	Ink in not being leak from refill in normal conditions.

Consider, a test suite covering an aggregate of 10 test cases. The relapse test suite "T" as given in Table 1, contains 10 test cases {T1, T2, T3, T4, T5, T6, T7, T8, T9, T10} and their respective execution time. The bit 1's in the test cases represent that this test case satisfies the corresponding requirements. For example in test case id T1: 1010010010 satisfies requirements {R1,R3,R6,R9}.

Table 2. Table contains Test Id, Test Cases and Execution Time of test cases used in experiment

Test Id	Test Cases	Execution Time
T1	1 0 1 0 0 1 0 0 1 0	7
T2	0 1 0 0 0 0 0 1 0 0	3
T3	0 1 0 0 1 0 1 0 0 0	5
T4	0 0 0 1 0 1 0 1 1 0	5
T5	1 1 0 0 0 1 0 0 0 1	3
T6	0 0 0 1 1 0 0 1 0 0	6
T7	1 0 1 0 0 0 1 1 0 0	3
T8	0 0 1 0 0 1 0 0 0 1	2
T9	1 1 1 0 0 1 1 0 0 0	1
T10	1 0 0 0 0 0 0 0 0 1	4

The experiment is done in Matlab. First the procedure takes a file which contains the test cases and then takes another file which contains execution time as shown in Fig 3. After selecting the files it optimizes the test cases and result in shown in different text box. The result contains the Optimized test cases and their execution time.


TEST CASE OPTIMIZATION

Browse Test Cases: D:\M.tech labs\dissertation\disserproj\updated - Cop

Browse Execution Time: D:\M.tech labs\dissertation\disserproj\updated - Cop

Test Cases Loaded

```
T1  - 1010010010   7
T2  - 0100000100   3
T3  - 0100101000   5
T4  - 0001010110   5
T5  - 1100010001   3
T6  - 0001100100   6
T7  - 1010001100   3
T8  - 0010010001   2
T9  - 1110011000   1
T10 - 1000000001   4
```

Optimize Test Cases Loaded

```
1110011010   20
1011010110   15
0011010111   17
1110010111   18
```

Fig3. Test cases and execution time selected and Optimized Test cases with their execution time

Fig4 shows the graph between the quantity of test cases used and quantity of optimized test cases.

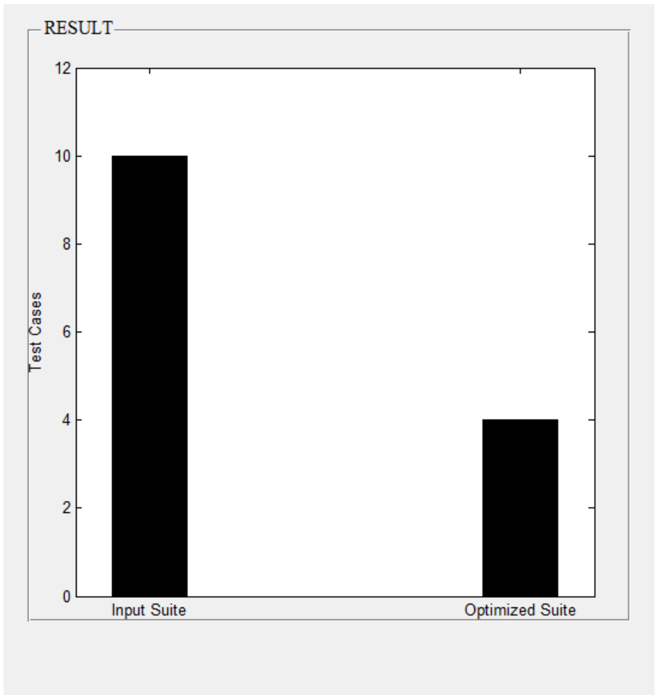


Fig4. Graph between the number of input suite and optimized suite

Now the Hybrid approach which is an integration of Genetic algorithm and Ant Colony optimization algorithm is compared with individual GA and ACO. Fig5 conclude the graph between the hybrid approach and GA and ACO.

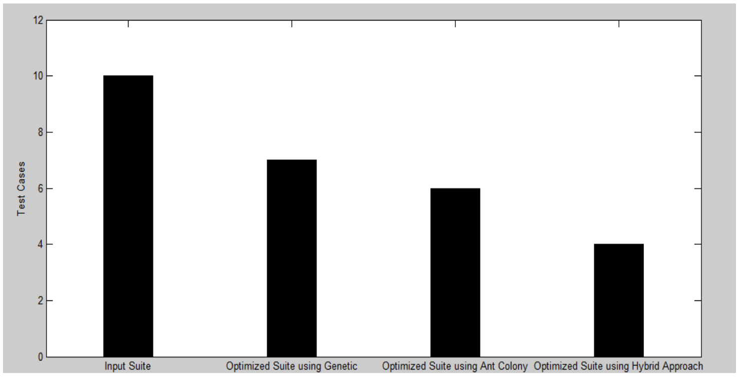


Fig5. Comparison between Hybrid approach and GA and ACO

The result in fig5, helps to conclude that Hybrid Approach is better the GA and ACO as the number of optimized test cases are better and which covers maximum fault detection capacity and least execution time.

VII. CONCLUSION AND FUTURE SCOPE

The proposed experiment generates an approach from an expansive test suite optimization by integrating Genetic algorithm and Ants Colony optimization algorithm. This approach has been tried for a few cases. One of these set of test cases has been appeared in this paper. The strategy created utilizing this methodology distinguishes and decreases the test information. The methodology gives better results in the underlying emphasis of the entire procedure. It gives positive input and subsequently it can prompt better arrangements in ideal time.

The paper also provides comparison between the produced Hybrid approach and Genetic and Ant colony algorithm. The comparison is based on the quantity of the test cases. Hybrid approach gives better result than both these algorithms with maximum fault detection capacity and least execution time.

Issues of future exploration incorporate automation of the strategy and applying it on vast and complex programming. We likewise plan to contrast it with particle swarm optimization and genetic algorithm.

ACKNOWLEDGEMENT

I might want to thank my guide Ms. Mamta Arora for giving me encourage and to inspiring me.

REFERENCES

[1] Kevilienuo Kire and Neha Malhotra, "Software Testing using Intelligent Technique", International Journal of Computer Applications (0975 – 8887), Vol. 90– No.19, March 2014, pp. 22-25

[2] G.Duggal, B.Suri,"Understanding Regression Testing Techniques", COIT, 2008, India.

[3] 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.

[4] K.s. Tang, Sam Kwong, “Genetic algorithms: Concepts and applications”, IEEE transactions on industrial electronics, Vol. 43, No. 5, Oct 1996, pp. 519-534

[5] http://file.scirp.org/Html/7-9601193%5Ca 9b12e 87-908f-4343-8b9d-f41cb52fad6f.jpg

[6] https://www.google.co.in/search?q=ant+colony+optimization&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjm54Wh_d7MAhULp48KHc9mD5cQ_AUICCgC&biw=1366&bih=643#imgrc=zsvOqsppTU2GRM%3A

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

[8] Abhishek Singhal, Swati Chandna, Abhay Bansal “Optimization of Test Cases Using Genetic Algorithm”, International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 3, March 2012, pp. 367-369

[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] 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.

[11] Amit Kumar Sharma,” Optimized Test Case Generation Using Genetic Algorithm”, International Journal of Computing and Business Research (IJCBR), Volume 4 Issue 3 September 2013

[12] D. Goldberg, “Genetic Algorithms in Search Optimization and Machine Learning”, New York,Addision Wesely, 1989.

[13] 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.

[14] 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.

[15] J.Holland, “Adaption in Natural and Artificial Systems”, Ann Arbor, MI: University ofMichigan Press,1975.

[16] 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

[17] Huaizhong Li and C. Peng Lam,” Software Test Data Generation using Ant Colony Optimization”, International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:1, No:1, 2007, pp.137-140.