Software Test Suite Minimization using Ant Colony Optimization

Akanksha Gaur, Mohammad Arif

Abstract: ***In the fast pacing technological era, the key to a successful software industry is quick delivery of high quality software to the clients. This high quality is achieved by performing software testing on the product. The high quality product ensures stakeholder's satisfaction which in turn spreads good word about the software industry making it a success. In this paper, we will focus on the problems faced during regression testing and how the same can be handled. Regression testing is a critical activity done during the software maintenance phase of the software development cycle. However, it has countless underlying issues like effective test case generation and prioritization, etc which need to be dealt with. These issues demand effort, time and cost of the testing. Different techniques and methodologies have been proposed for taking care of these issues. Use of Ant Colony Optimization (ACO) for test suite minimization has been an area of interest for many researchers. This paper presents an implementation of ACO for test suite minimization, showcasing how arbitrary nature of ACO helps choose an optimal solution to the problem.***

Index Terms: Analysis, Ant Colony Optimization, Software testing, Test suite minimization

I. INTRODUCTION

Ant colonies are remarkably similar to cities. No one choreographs the action, not even the queen ant, but ant behaviour is controlled by swarm logic- put 10,000 dumb ants together, and they become smart." - Alex Cukan. Software testing is an investigation to provide information a bout the quality of the software product or service being test ed to stakeholders. Software testing presents an objective, autonomous software scrutiny to enable the company to appreciate and understand the software implementation risks. Testing involves running a program to reveal errors in order to solve the same. Testing, however, can't guarantee perfect operation of the software under all circumstances; it only verifies that the software will run correctly in specific conditions (as asked for by the client). Testing is a creative process but in some cases previously written tests can be used to save time and cost. Such an approach is suitable for regression testing. "Regression testing is re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change." [2] Its a necessary component to any software development lifecycle as it performs maintenance and quality assurance. Yoo and Harman [3] submitted a rigorous survey on Minimization, Selection and Prioritization of Regression testing. Engstrőm and Rumerson also published a questionnaire based on the Regression Testing Practices in 2010 by [4]. Ant Colony

Optimization is an extremely popular methodology to solve NP hard problems. In regression testing, which is a highly time and cost constrained technique, we need to determine which and how many tests will be run to test the software and this problem is hard, NP hard. Thus, ACO is used to resolve this problem giving an optimal solution. In spite of being such tiny creatures, ants have tremendous brain power. All ants are blind and navigate by the use of a special chemical substance called pheromone. When ants locate food they make a trail for returning to their homes as well. Other ants follow the maximum pheromone trail leading to food and then back home via an optimal path. Also, the ant on the shortest path is the earliest to return, depositing more pheromone on that path (when going back and forth to the nest). This increases the probability that other ants will follow this path [5]. The ants that follow this path will lay more pheromones on that path again. Therefore, even more ants tend to follow this path and continue the process, in the end all the ants converge to the shortest or the best path. Derived from this behaviour of ants, Marco Dorigo proposed Ant Colony Optimization algorithm in 1990's [6].

II. ANT COLONY OPTIMIZATION

"ACO is a population-based search method, based on stigmergy - a consensus social network mechanism of indirect coordination, through the environment, between agents or actions" [7]. As discussed earlier, real world ants use their antennas and deposit pheromone while searching for food. Similar to this, artificial ants leave a virtual pheromone trail. For implementing ACO, the problem is represented in a graph, having possible stops as nodes and possible paths as edges. Now the ants travel through these paths depositing pheromones. In case of equal or no pheromone, random path selection is chosen else the path with maximum "pheromone" is chosen as it's considered the shortest. This autocatalytic behaviour makes the process positive feedback oriented. Once pheromones are laid, it evaporates after a certain interval of time thus helping in avoiding local optimal solution convergence.

This phenomenon has been applied to various fields such as that of MANET[10-12], travelling salesman problem[6], [13], [14], [15], and software testing [16], [17], [18], clustering[19-22], IoT[23], Big Data[24-25], machine learning[26], fuzzy[27-28], e-learning[29], robotics [30].

Revised Manuscript Received on July 02, 2019.

Akanksha Gaur, Computer Science and Engineering, Integral University, Lucknow, India.

Mohammad Arif, Computer Science and Engineering, Integral University, Lucknow, India.

III. TIME CONSTRAINT TEST SUITE PRIORITIZATION AS NP COMPLETE PROBLEM

Regression testing can be considered as a maintenance activity. In this, the functioning of modified software is put under test but testing the whole software with the complete test suite we have is a complete waste of time as well as adds onto the expenses. So, to tackle this problem we use a selection technique that chooses a subset of test suite that was used to test the software before changes were made, and then use that subset to test the modified software. The practice of sequential scheduling of test cases, such that all the performance goals are met, is referred to as prioritization and finding this minimal test suite sub-set is NP-complete, since it can be reduced in polynomial time to the minimal hitting set problem [8]. As all NP complete problems are NP hard, thus ACO can be used to solve it.

IV. IMPLEMENTING ACO FOR TEST SUITE MINIMIZATION

For performing test suite minimization, using ant colony algorithm, we have developed an editor by mapping it with external jre that will help in compilation.

This editor will incorporate the following techniques:

- 0/1 knapsack problem
- Ant Colony Optimisation
- Naive Bayes

All the above mentioned techniques serve different purposes and have helped achieve an accuracy of up to 95%.

Firstly, the external jre mapped with editor will check for errors at compile time and try to auto complete the statements by matching the keywords and classes from the libraries, aided by ants. The ants perform auto completion in the following manner:

- The errors in the code act as food for the ants.
- The ants move towards this erroneous code in order to correct it by removing maximum errors. Subsequently other ants follow and errors are tackled quickly.
- These errors are removed based on the classes, interfaces and libraries related grammatical errors.

Secondly, erroneous statements are assigned using 0/1 knapsack problem.

Finally, using ant colony algorithm the paths to erroneous statements are predicted and travelled by the ants such that all possible errors in the periphery are solved automatically.

V. OVERVIEW OF THE ALGORITHM

Initialization of code:

1. Set error points: N= {a1, a2, a3,…}
2. Edges E= {(a1, a2), (a3, a4), ….} represent the relation between lines
3. F= {Fxy (P)} represents the direct relation between the current line ‘x’ and neighbour line ‘y’
4. R = rxy (A) represents the edge (x,y) pheromone level for (A)
5. Vs = {V(x)} represent the information about the node ‘x’ is already visited or not.
6. Weight of each path that represents the optimal weight of each edge.

Once code is imported, a list of errors is created. Edges are marked for the whole code that represents the relation between the points. Fxy(P) represents the relation between the current point and neighbouring point. Rxy (A) defines the pheromone level on the edge and. Vs marks the visited nodes. At last the optimal weight is calculated. These edges are travelled by the ants to find errors. Each edge shows the complexity tackled to reach the error, and each point of error represents deposited pheromone.

Start:

1. r ← 1
2. Edge n← z
3. α ← 1 and β ← 1
4. cc ← complexity of the path
5. while (cc>0)
6. i←start,w←0,net_weight[cc]←0no_of_transition ←0
7. if Vs[i]=0
8. Vs[i]←w
9. Select the path if it is optimal
10. Move to the next node
11. End if
12. End do
13. Print the all visited path according to sequence of selection
14. Stop

First node is labelled r. Two nodes are connected and the edge is named n. α denotes time complexity and β denotes time complexity. cc is calculated on the basis of α and β. While cc is greater than 0, a loop for nodes is started, where the first node is initialized as 0. The count of errors is stored in w and the total number of errors in the path covered is considered as the net weight that is incremented accordingly. The path with maximum errors is chosen and resolved which subsequently solves neighbouring errors as well.

Ants are trained using our sample data sets. All the possible errors are made into list, all the code is checked based on his list. Any mismatch will be marked as an error. These errors are classified based on the error samples in the datasets, classifier categorizes errors. The error code that has the probability of solving maximum number of peripheral errors is solved first. 0/1 Knapsack problem is applied to check for the optimal path that solves most number of errors in least amount of time. Hence, all the erroneous code is traced at least once. The parameters for grammatical errors are initialized, these are braces ({}), semi colon (;) and keywords (language dependent). The complexity of the path will be initialized as 1 for ant (1). The pheromone is initialized 1 and it keeps on increasing as the ant moves from one error to another. The Naïve Bayes classifier trains the ants based on the trained dataset (containing all possible syntactical and semantic errors of a language), which is different for all languages. As the ant

(1) moves forward, all the ants follow and update pheromones. Check if the errors are removed or not, if errors are still left the steps are executed again but if all the errors are resolved 0/1 Knapsack problem is applied to choose the optimal path.

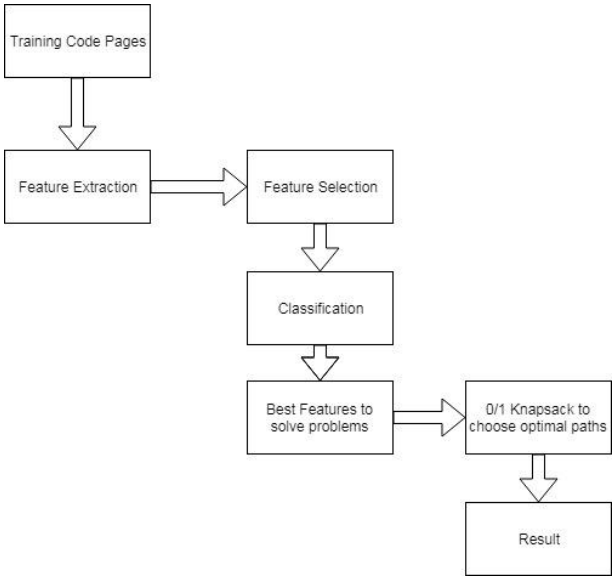

Fig. 1: Block Diagram of the tool ANTK

Figure 2 shows that, ants are trained using our sample data sets. All the possible errors are made into list, all the code is checked based on his list. Any mismatch will be marked as an error. These errors are classified based on the error samples in the datasets, classifier categorizes errors. The error code that has the probability of solving maximum number of peripheral errors is solved first. 0/1 Knapsack problem is applied to check for the optimal path that solves most number of errors in least amount of time. Hence, all the erroneous code is traced at least once. The parameters for grammatical errors are initialized, these are braces ({}), semi colon (;) and keywords (language dependent). The complexity of the path will be initialized as 1 for ant (1). The pheromone is initialized 1 and it keeps on increasing as the ant moves from one error to another. The Naïve Bayes classifier trains the ants based on the trained dataset (containing all possible syntactical and semantic errors of a language), which is different for all languages. As the ant (1) moves forward, all the ants follow and update pheromones. Check if the errors are removed or not, if errors are still left the steps are executed again but if all the errors are resolved 0/1 Knapsack problem is applied to choose the optimal path.

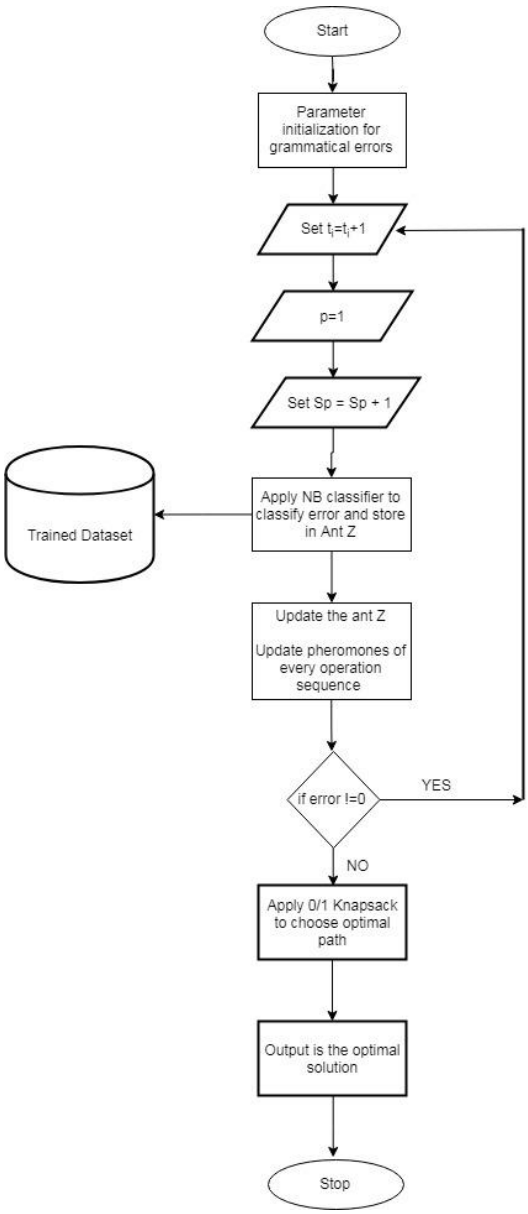

Fig. 2: Algorithm flowchart

VI. RESULTS AND DISCUSSION

Here in the graph, fig 3, the orange bars denote the execution time for a test case when tested by MHBG_TCS, a tool developed using bee colony optimization and ant colony optimization by Bharti Suri and Shweta Singhal, whereas the yellow bars denote the execution time taken by our tool ANT_K that is built based on modified ACO algorithm as we discussed in the previous chapters. The tool gives highly accurate results. Even when compared with one of the most sophisticated tools MHBG_TCS. As seen in the graph, ANT_K performs well in multiple test cases such as P1, P2, P4, P6 and P10. For the newly developed tool to provide such good results is rather impressive.

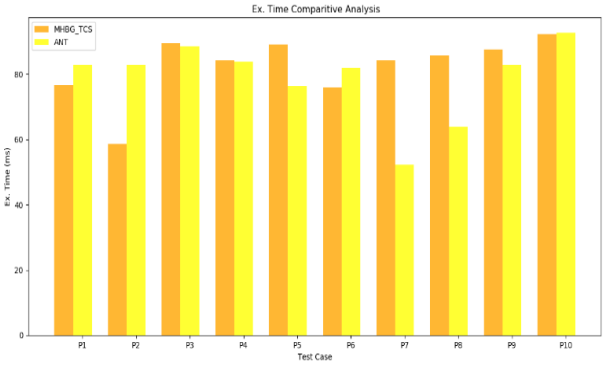

Fig. 3: Comparison of ANTK with MHBG_TCS with respect to Execution Time

TABLE I: Comparison in terms of Execution Time

Test Case	MHBG_TCS (Execution time in ms)	ANTK(Execution time in ms)
P1	76.67	82.78
P2	58.57	82.78
P3	89.47	88.6
P4	84.21	83.78
P5	89.09	76.23
P6	76	81.29
P7	84.21	52.28
P8	85.71	63.98
P9	87.5	82.78
P10	92.3	92.85

Average of MHBG = 823.81/10
= 82.381

Average of ANT = 787.15/10
= 78.715

Accuracy = (Average of ANT / Average of MHBG) * 100
= (78.715/ 82.381) * 100
= 95.54 %

Comparison with MHBG_TCS [9] in terms of different languages:

In the following graph, we have compared our tool with MHBG_TCS while testing programs on C++ and java.

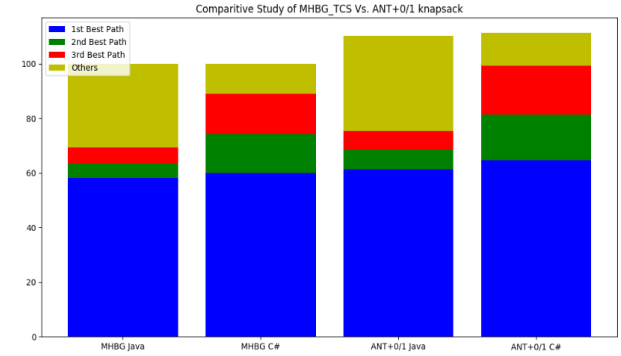

Fig. 4: Comparison of ANTK with MHBG_TCS with respect to Execution Time

As show in the above graph, figure 4, as the time constraint is increased 100% correctness is achieved, which is suggestive of the fact that if more time is given to the approach a 100% optimally minimized test suite can be achieved. The results of the chosen test suite are categorized as the following (based on maximum faults covered and minimum execution time):

- The Best Path (Optimal Path)
- Second Best Path
- Third Best Path
- Others

Inferring from the graph, efficiency of both tools is similar with slight to no variations, in both C++ and java.

VII. CONCLUSION

We need to evaluate the use of ACO in software testing as it is a hot topic and it's going under a lot of research.

Till now only pheromone has been the considered as the algorithm's parameter with randomizes motion and increase time of convergence. But the new algorithm blends in the benefits of 0/1 Knapsack Problem and Naïve Bayes and improves the efficiency of the tool. The tool gives highly accurate results. Even when compared with one of the most sophisticated tools MHBG_TCS, it performs almost equally or even better in some test cases. This builds our trust in the technique being proposed. Similar results have been achieved for all types of programming languages.

REFERENCES

[1]G.M. Kapfhammer, *"Software Testing,"* Chapter in book, Department of Computer Science, Allegheny College, June 2003.
[2]Basu, Anirban (2015). *Software Quality Assurance, Testing and Metrics*. PHI Learning, ISBN 978-81-203-5068-7.
[3]S. Yoo. and M. Harman, "Regression testing minimization, selection and prioritization: a survey." , *Software Testing Verification and Reliability*, 2010. doi: 10.1002/stvr.430.
[4]E. Engström, P. Runeson, "A Qualitative Survey of Regression Testing Practices," *Lecture Notes on Computer Science (LNCS), Springer Verlag*, 2010, pp. 3-16.
[5]Caro, G. Di and Dorigo, M., "AntNet: Distributed stigmergetic control for communications networks" *Journal of Artificial Intelligence Research*, vol. 9, 1998, pp. 317-365.
[6]M. Dorigo, V. Maniezzo, and Colorni, A. "Ant System: Optimization by a colony of cooperating agents" *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3(26), 1996, pp. 29-41.
[7] L. Marsh, C. Onof, "Stigmergic epistemology, stigemergic cognition", *Cognitive Systems Research*, vol. 9, 2008, pp. 136.
[8]A. Ramarajan, S. Usha "Diversity Based Genetic Algorithm for Efficient Test Case Selection", *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5(4), 2016.
[9]B. Suri, S. Singhal "Evolved regression test suite selection using BCO and GA and empirical comparison with ACO", *CSIT*, vol. 3(2–4), 2015, pp. 143–154.
[10]M. Arif, K. I. Rahmani "Adaptive ARA (AARA) for MANETs", *IEEE Xplore in the Proceedings of 3rd Nirma University International Conference on Engineering [NUiCONE 2012]*, 2012. ISBN 978-1-4673-1720-7.
[11] M. Arif, T. Rani. "ACO based Routing for MANETs", *International Journal of Wireless & Mobile Networks (IJWMN).* vol. 4(2),2012, pp. 163-174. ISSN: 0975 - 3834[Online]; 0975 - 4679 [Print].
[12]M. Arif, T. Rani. "Enhanced Ant Colony based Routing in MANETs". *Proceedings of 5th IEEE International Conference on Advanced Computing & Communication Technologies [ICACCT-2011]*, 2011, pp. 48-54. ISBN 81-87885-03-3.
[13]H. Li and C. Peng Lam, "Software Test Data Generation Using Ant Colony Optimization," *Transactions on Engineering, Computing and Technology*, 2005.
[14]R.S. Parpinelli, H.S. Lopes, and A.A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, 2002, pp. 321–332.
[15]P. Zhao, P. Zhao, and X. Zhang, "New Ant Colony Optimization for the Knapsack Problem," *Proceedings of the 7th International Conference on Computer-Aided Industrial Design and Conceptual Design*, 2006, pp 1-3.

[16]K. Doerner, W.J. Gutjahr, "Extracting Test Sequences from a Markov Software Usage Model by ACO," *Proceedings of GECCO 2003, LNCS Springer-Verlag Berlin Heidelberg*, vol. 2724, 2003, pp. 2465-2476.
[17]F.P. McMinn, M. Holcombe, "The State Problem for Evolutionary Testing," *Proceedings of GECCO 2003, LNCS, Springer Verlag*, vol. 2724, 2003, pp. 2488-2500.
[18]B. Suri and S. Singhal, "Implementing Ant Colony Optimization" for Test Case Selection and Prioritization", *International Journal on Computer Science and Engineering*, vol. 3(5), 2011, pp. 1924-1932.
[19] L. M Goyal, M. Mittal and J. K.. Sethi, "Fuzzy Model Generation using Subtractive and Fuzzy C–Means Clustering", *CSI Transaction on ICT, Springer*, 2016, pp 129-133
[20]A. Saxena, M. Mittal and L.M. Goyal, "Comparative Analysis of Clustering Methods", *International Journal of Computer Applications*, vol.118(21), 2015, pp. 30-35.
[21]M. Mittal, L.M. Goyal, D. J. Hemanth and J. K. Sethi, "Clustering Approaches for High-Dimensional Databases: A Review", *WIREs Data Mining KnowlDiscov, John Wiley & Sons*, 2019, pp. 1-14. DOI: 10.1002/widm.1300
[22]M. Mittal, L.M. Goyal, J. K. Sethi and D.J. Hemanth, "Monitoring the Impact of Economic Crisis on Crime in India Using Machine Learning", *Computational Economics, Springer*, 2018, pp. 1-19.
[23]Rajesh Singh, Anita Gehlot, Mamta Mittal, Rohit Samkaria and Sushabhan Choudhury, "Application of iCloud and Wireless Sensor Network in Environmental Parameter Analysis*", International Journal of Sensors, Wireless Communications and Control*, vol 7(3), 2018, pp. 170-177.
[24]Singh A., Mittal M., Kapoor N, "Data Processing Framework Using Apache and Spark Technologies in Big Data*" Big Data Processing Using Spark in Cloud. Studies in Big Data*, vol 43, 2018, pp 107-122.
[25] Mamta Mittal, D. Jude Hemanth, Valentina Emilia Balas and Ragavendra Kumar, "BigData for Parallel Computing" *Advances in Parallel Computing Series ,IOS Press*, 2018.
[26] Mamta Mittal, Lalit Mohan Goyal, Jasleen Kaur Sethi and D Jude Hemanth, "Monitoring the Impact of Economic Crisis on Crime in India Using Machine Learning", *Computational Economics, Springer* , 2018, pp. 1-19.
[27] Goyal L. M., M. Mittal and Sethi, J. K., "Fuzzy Model Generation using Subtractive and Fuzzy C–Means Clustering", *CSI Transaction on ICT, Springer*, 2016, pp 129-133
[28] Nidhi Beniwal, Mittal M., Goyal L.M. & Monika, "Enhance Ad-hoc On-Demand Distance Vector Routing Protocol", *Indian Journal of Computer Science and Engineering*, vol 8 (3), 2017, pp. 463-469.
[29] Aakash Saini, Mamta Mittal, Shweta Singh, "An FPGA Based Efficient Surveillance System: A Split Processing Approach", *International Journal of Imaging and Robotics*, vol. 18(3), 2018.
[30]Shivani Chauhan, Mamta Mittal, Aakash Saini, "Microstrip Patch Antenna: A Design to Study the Parametric Trade-off", *International Journal of Research*, vol. 4(14), 2017, pp 2646- 2654. ISSN 2348-6848

AUTHORS PROFILE

Akanksha Gaur is a M.Tech scholar at the department of Computer Science and Engineering at Integral University Lucknow.

Mohammad Arif is an assistant professor at the department of Computer Science and Engineering at Integral University Lucknow. His works have been published in various reputed journals.