Ant Colony Optimization Based Optimization of Software Test Suite

Akanksha Gaur
Department of Computer Science and Engineering
Integral University, Lucknow, India

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

Abstract: Ant Colony Optimization (ACO) is naturalistic approach that has been a hot topic of research from the past decade. It is a probabilistic technique which has shown impressive results in solving computational problems, by reducing the problem size in order to find an optimal solution. The advantages of ACO technique can be put to use in innumerable problems, one of which is the discipline of software testing. This paper presents a comprehensive study of thirty such studies, which focus on applying ACO in various software testing problems. As per our studies, till date only one literature survey exists in this context. Hence, this paper crucially analyzes the common parameters of all the studies.

Keywords: Software Testing, Ant Colony Optimization, Test Suite minimization, Analysis.

I. INTRODUCTION

Software testing is the practice of executing a program with the aim of revealing errors so that the same can be resolved. It's one of the fundamental techniques which estimate the quality of the software. Software testing ensures that the software is made in accordance to the business aspects and customer requirements. In spite of all this, software testing does not ensure that the product is perfect and error free. It can only ensure correct working under specific conditions. Complete testing the product is practically impossible. Numerous surveys can be observed in the field of software testing. The first ever methodical review in software testing was published in 2004 on "the testing technique experiments" [1]. Yoo and Harman have presented a thorough survey on "Regression Test Suite Minimization, Selection and Prioritization" [2]. Similarly, a literature survey specializing on Regression Test Selection process was presented by Ding, Kau, Li and Yang [3]. Engström and Runeson [4] also published a questionnaire based survey in 2010.

ACO mimics the behaviour of real life ants in finding the optimal path from their nest to the food source; this movement of ants is aided by pheromones and this technique is called "stigmergy". Dorigo, the father of ACO, in 1990's proposed the theory of virtual ants finding solutions to problems using graphs [5]. B. Suri and S. Singhal [6] presented an in depth literature review on "Ant Colony Optimization in Software Testing".

By now Dorigo's technique has been practically applied in resolving combinatorial optimization problems milieu networking; such as knapsack problem, vehicle routing, distributed networks, data mining, telecommunication network, travelling salesman problems, [5, 7, 8, 9] etc. 2003 was the year when ACO was applied to software testing concepts for the first time ever [10, 11] and since then research in this field has been intensifying [12].

This paper emphasises on the applications, effects and advantages of ACO in software testing and how it can be made more accurate.

II. SOFTWARE TESTING

The activity of checking whether the actual results match with the expected results is software testing, it also ensures that the code is bug free. This usually involves executing the programs to evaluate its functionality, correctness and business value. Testing is a necessary evil, as it's expensive in terms of money and time but cannot be avoided as unresolved errors will definitely lead to potential monetary losses later. Although there are more 150 types of testing, software testing can be broadly classified into two categories, namely, White Box Testing or Non Functional Testing and Black Box Testing or Functional Testing.

White box testing helps discover errors such as incorrect spellings or missing requirements in contrary to black box testing which focuses on actual requirements. One more important phase of testing is Regression testing, it ensures that the amendments and corrections made in the code do not lead to more errors, which is kind of a maintenance activity. This is the most expensive kind of testing as it involves re-testing of all the test cases that have already been executed. In spite of being quite fruitful, regression takes a lot of effort, manpower and time, thus this area of testing needs to be automated for providing cost effective solutions. For the same purpose we use techniques of test suite minimization, selection and prioritization, which in turn make the testing process intelligent.

Test suite minimization is an iterative process which can be studied in the following diagram, Fig 1.

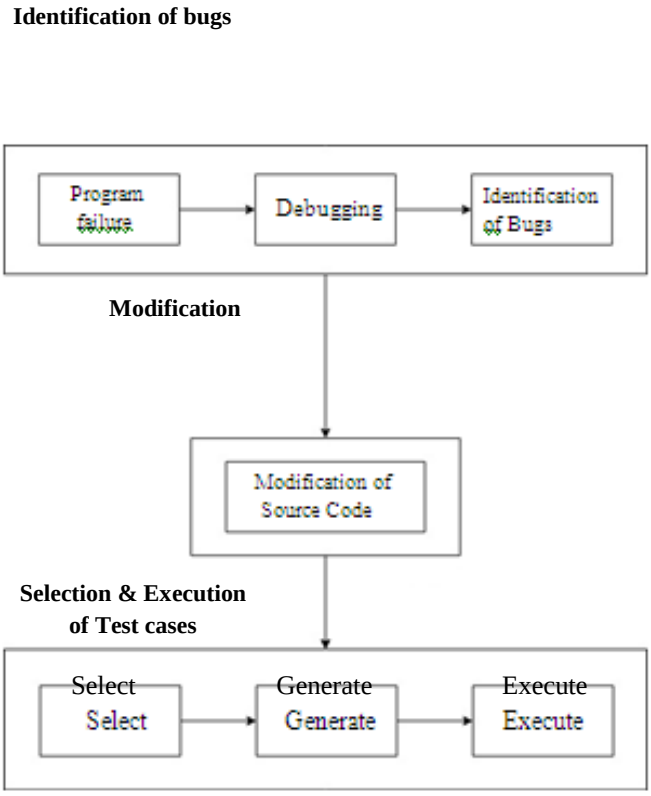

Fig.1 Test Suite Minimization process

III. ANT COLONY OPTIMIZATION

As discussed earlier, ACO is rooted on the hypothesis of "stigmergy" – indirect communication between members of a population through interaction with the environment [5]. The population of ants and amount of pheromones laid are the two basic parameters of this approach.

Detailed explanation of the Ant Colony:

1. At time = 0, n number of ants are present at the nest N.
2. As the ants move from N to N+1, they deposit a substance named "pheromone" on the trail.
1. Ant n_1, ..., n_{n+1}, move on an arbitrary path laying down pheromones making a pheromone trail which other ants can smell, Fig. 2. The number of ants on the first move depends on the number of available paths.
2. Ant n_2, n_3, n_4, etc, follow this pheromone trail which eventually leads them to the food source, Fig. 3.
3. Though this pheromone trail does not remain forever because the pheromone keeps evaporating at a specific rate.
4. The ant choosing the shorter path will also be faster in returning to the nest, because of the stronger pheromone smell.
5. Hence, after a specific period of time, all the ants converge to follow the shortest path, Fig. 4.

The following diagrams represent the working of Ant Colony, with three paths, where,

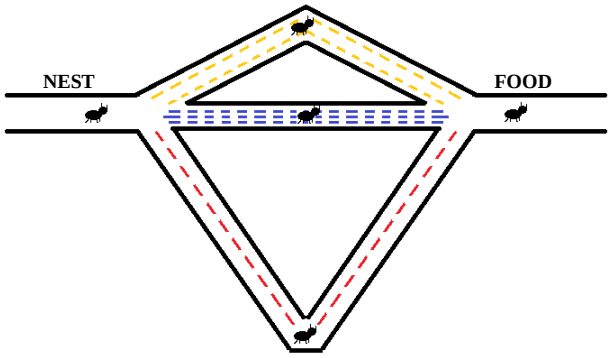

Fig.2 Working of ACO, for ant n_1, n_2, n_3

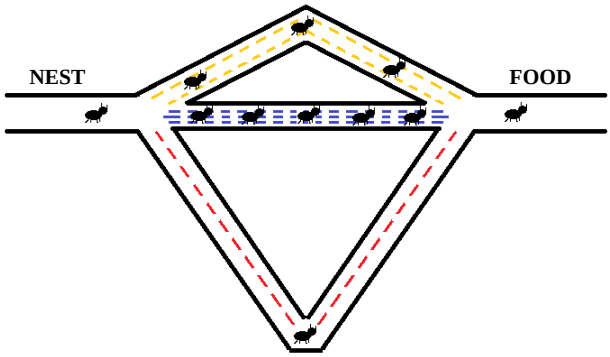

Fig.3 Working of ACO, following ants deposit more pheromones

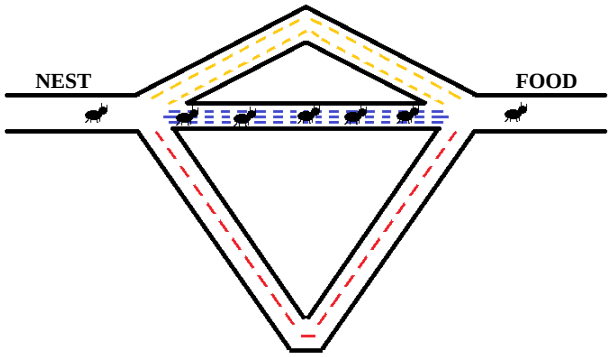

Fig.4 Working of ACO, all ants converge to the follow the shortest path

- Red path = Least pheromone level
- Orange path = Moderate pheromone level
- Blue path =Highest pheromone level

IV. NEED FOR SURVEY ON USE OF ACO IN SOFTWARE TESTING

During studies we reported that ACO has been applied to various networking problems but its application to software testing is fairly new. ACO has shown optimal results in solving NP hard problems. The selection and prioritization of test suites requires creation of minimal subsets of a test, which is a NP complete problem, as this set can be obtained in polynomial time. As we know, all NP complete problems are NP hard; ACO can be used to solve it. However, standalone ACO does take longer time to converge and uses randomization up to some extent. We need a modified ACO algorithm which not only has all the benefits of original ACO but has added functionalities also.

V. HISTORY OF ACO

The first implementation of ACO technique was accomplished in the 1996 by Dorigo [5]. In his paper, he defined ACO as a "new general-purpose heuristic algorithm which can be used to solve different combinatorial optimization problems". The paper describes the following properties of ACO: Versatility, Robustness and Population based approach. It also proposes and implements an algorithm of ACO for solving the travelling salesman problem. Later in 2003, Doerner and Gutjahr discussed the extraction of suitable test paths from Markov Usage Model description of expected use of software system [10]. Their approach discussed the trade off between testing costs and failure risks that would be caused by untested state transitions. Further in 2003, McMinn and Holcombe in their paper showcased the effects of presence of states in test objects in the search of test data using evolutionary testing [11].

VI. RELATED WORK

Since, testing is a necessity; specialists have introduced copious techniques of test suite minimization in order to reduce time and efforts. Many taxonomies of ACO's application in software testing have been discussed as:

1. Analyzing Test Case Selection & Prioritization using ACO

Bharti Suri, Shweta Singhal (December 2015 in CSIT)

The authors discuss the evolved regression testing suite selection using BCO, GA and empirical comparison with ACO. The paper presents a new improved modified technique based on Bee Colony Optimization and Genetic algorithm that makes use of permutations/combinations to generate a new set of test cases [31].

2. A safe, efficient regression test selection technique

Do H, Mirarab S, Tahvildari L, Rothermel G (April 2008 in ACM Transactions on Software Engineering and Methodology (TOSEM)

The authors make an empirical study of the effect of time constraints on the cost-benefits of regression testing. [32].

3. Software Test Suite Minimization using Genetic Algorithm

Vineeta Mishra, Mohd Haroon, Sish Ahmad (2017, IEEE)

In this paper, the authors enlist the automated approach to select test cases for OOP by using genetic algorithm [33].

4. Ant System: Optimization by a Colony of Cooperating Agents

Marco Dorigo (IEEE transactions on systems, man, and cybernetics-part b cybernetics, February 1996)

An analogy with the way ant colonies function has suggested the definition of a new computational paradigm, which we call Ant System. The authors propose it as a viable new approach to stochastic combinatorial optimization. The main characteristics of this model are positive feedback, distributed computation, and the use of a constructive greedy heuristic [5].

5. Evolved regression test suite selection using BCO and GA and empirical comparison with ACO

Bharti Suri, Shweta Singhal (CSIT 3(2-4):143-154, December 2015)

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. A lot of techniques have already been developed and proved to be very effective in reduction of test suite. The paper presents a new improved modified technique based on Bee Colony Optimization and Genetic Algorithm that makes use of permutations/combinations to generate a new set of test cases. The proposed modified technique has been empirically validated on 17 sample programs for its near 90% correctness and an average 80 % execution time reduction capability. Also, the developed technique has been compared with the existing technique for test case selection using ACO. The comparison proves the superiority of the developed technique against the existing one in majority of the programs with some exceptions. In addition to this, the results have been analyzed based on the language of the programs under test and the type of desired result. The comparison between the tool MHBG_TCS (developed for the technique proposed in this paper) and the tool ACO_TCSP (existing tool) yielded superiority of the new technique in general. All the results prove the validity of our technique and inspire us to work further on this technique [34].

6. Ant colony optimization algorithm: advantages, applications and challenges

Kavita Tewani (Computer Modelling & New Technologies 2017 21(2) 69-70)

Ant Colony optimization is a technique for optimization that was introduced in early 1990's. ACO algorithm models the behavior of real ant colonies in establishing the shortest path between food sources and nests and this technique is applied on number of combinatorial optimization problem, communication networks and robotics. This paper introduces the advantages of using the ACO algorithms with the help of some problem examples and the challenges faced for solving the problems. Initially, the paper discusses about the biological inspiration and behavior of ant colony and then relates with the real life problems [35].

7. Implementing Ant Colony Optimization for Test Case Selection and Prioritization

Bharti Suri, Shweta Singhal (IJCSE, December 2015)

In this paper, the authors give a detailed study of their tool MHBG_TCS. It showcases how the randomized motion of ants helps them explore all the possible paths and choose the optimal one. The results of this study show promising findings as the solutions are in close proximity with optimal solutions [12].

Table I. Comparative analysis of Field of Application, Legacy, and Tool used, Input, Approach

Reference	Field of Application	Legacy	Tool used	Input	Approach
[10]	Test Sequence Generation	Call centre application	Not Declared	Markov Usage Model graph	Markov usage model based description
[11]	Test Sequence Generation	Self-built	Not Declared	Source Code	Structural testing based
[12]	Test Suite Minimization and Prioritization	ACO_TCSP	Not Declared	Source Code	Path coverage and fault detection
[13]	Test Data Generation	Coffee & Cocoa Vendor Machine	Not Declared	UML State Chart Diagram	State based software testing
[14]	Test Data Generation	Coffee & Cocoa Vendor Machine and "Being Taught"	DAS-Dynamic Ant Simulator	UML State Chart Diagram	State based software testing
[15]	Test Data Generation	Triangle, JAVA, Next Data	Not Declared	Source Code	Mutation Testing
[16]	Optimized Test Sequence Generation	Coffee & Cocoa Vendor Machine	Search Applet 4%	UML State Chart Diagram	Branch & Statement coverage
[17]	Test Data Generation	Self-built	Not Declared	Pair-wise Test Input Data	Pair-wise covering Test Data
[18]	Test Suite Optimization	1.Factorial Program 2.Greeting Message 3.Marks Processing 4.Discount Calculation 5.Credit Card Validation 6.Calculator	Not Declared	Source Code	Path Coverage-Time spent on initial process will be a total waste and lead to a substantial time overhead
[19]	Test Data Generation	Cohen (+) generated variable strength interaction test suites inputs	Not Declared	Input with Factors	Fault based
[20]	Test Data Generation	Self-built	Not Declared	Input variable with Factors	Fault based coverage

[21]	Test Data Generation	Not mentioned	Not Declared	Control Flow Graph	Control Flow Graph based
[22]	Optimal Path Generation	Self-built	PPTACO	Control Flow Graph	Directed Graph approach Path Coverage based
[23]	Test Data Generation	Telephone Problem	Not Declared	State based model of the system	State, Model and Graph based
[24]	Test Sequence Generation	Enrolment System	SITACO Tool	UML State transition diagram	State Transition Coverage based
[25]	Path Generation	Not mentioned	Extended PPTACO	Pseudo code and Control Flow Graph	Control Flow Graph and Path Coverage based
[26]	Test Data Generation	Not mentioned	Not Declared	Control Flow Graph	Control Flow Graph and Path Coverage based
[27]	Test Data Generation	Triangle Problem	Not Declared	Input domain and variables	Path coverage and Domain based
[28]	Test Data Generation	Not Mentioned	Not Declared	Pair-wise combination of test cases	Pair-wise combination of test cases
[29]	Test Data Generation	Self-built	PCTDACO Tool	Domain of variables and code	Data flow testing approach
[30]	Survey of Evolutionary Computation Software Engineering	Not Mentioned	Not Declared	Not Mentioned	1.Test data generation based 2. Transitional statements based test data generation approach 3. State based software testing approach to test data generation using ACO
[31]	Test Suite Prioritization	Self-built	Not Declared	Source Code	Fault coverage based test case prioritization approach
[32]	Test Case Selection	Self-built	Not Declared	Control Flow Graph	Control Flow Graph and Path Coverage based
[33]	Test Suite Minimization	Self-built	Not Declared	Pseudo Code	Fault coverage based
[34]	Test Suite Selection	Self-built	MHBG_TCS and ACO_TCSP	1. Test cases 2. Faults 3. Execution time	Fault coverage based

[35]	Survey	Not Mentioned	Not Declared	Not Mentioned	Not applicable

VII. SUMMARY AND DISCUSSION

Taking the base paper into consideration, Suri et al. developed a tool MHBG_TCS [34] in C++ language.

This tool takes test cases, respective faults and execution time as inputs and gives the output as minimum number of test cases. They also developed an ACO based tool named ACO_TCSP which uses similar inputs but the output, i.e. selected test case path, was rather different. They relied their technique on the assumption that if the selected test case covers all possible faults further subsets of the test suite are not made and the time constraint selected as the final halting measure of their algorithm. The only problem with this is that the tool is language specific. Also, no clear effectiveness of one too over another is discussed.

VIII. CONCLUSION & FUTURE WORK

The objective of this study is judge the effectiveness of ACO when applied to software testing. After going through the work of prominent researchers in the field; we find that different researchers used different approaches for applying ACO to test suite minimization; these approaches reach from using probability, to time complexity to the merger of other naturalistic approaches such as genetic algorithm and bee colony optimization. Some of the discussed approaches in this paper have high accuracy. In the paper we compare several approaches for ACO in the table, Table 1. We concluded that ACO has high efficiency and accuracy when applied to software testing.

The study shows that ACO techniques are quite effective in software test suite selection and prioritization. As a future advancement we would develop a modified ACO algorithm that will help in reducing the time of convergence. Also, we would like to develop the tool as a multilingual one, so that its language independent.

IX. REFERENCES

[1] N. Juristo, A.M. Moreno, S. Vegas, “Reviewing 25 years of testing technique experiments”, Empirical Software Engineering Journal Vol. 1, No. 2, pp. 7-44, 2004.

[2] S. Yoo. and M. Harman, “Regression testing minimization, selection and prioritization: a survey.”, Software Testing Verification and Reliability. 2010, doi: 10.1002/stvr.430.

[3] W. Ding, J. Kou, K. Li, Z. Yang, “An Optimization Method of Test Suite in Regression Test Model,” In the Proceedings of the 2009 WRI World Congress on Software Engineering (WCSE), IEEE Computer Society Washington, DC, USA, Vol. 52, Issue 1, pp. 14-30, 2010.

[4] E.Engström, P.Runeson, "A Qualitative Survey of Regression Testing Practices," Lecture Notes on Computer Science (LNCS), Springer Verlag, pp. 3-16, 2010.

[5] M.Dorigo, V.Maniezzo, and A.Colorni, “Ant System: Optimization by a colony of cooperating agents,” IEEE Trannsactions on Systems, Man and Cybernetics, vol. B(26), 1996, pp. 29-41.

[6] B. Suri and Shweta Singhal, “Literature Survey of Ant Colony Optimization in Software Testing”

[7] H.Li, and C.Peng Lam, “Software Test Data Generation Using Ant Colony Optimization,” In Transactions on Engineering, Computing and Technology, 2005.

[8] 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.

[9] P.Zhao, P.Zhao, and X.Zhang, “New Ant Colony Optimization for the Knapsack Problem,” in the Proceedings of the 7th International Conference on Computer-Aided Industrial Design and Conceptual Design, Nov 2006, pp 1-3.

[10] K. Doerner, W.J. Gutjahr, “Extracting Test Sequences from a Markov Software Usage Model by ACO,” In the Proceedings of GECCO 2003, LNCS Springer-Verlag Berlin Heidelberg, Vol. 2724, 2003, pp. 2465-2476.

[11] F P. McMinn, M. Holcombe, M., “The State Problem for Evolutionary Testing,” In the Proceedings of GECCO 2003, LNCS, Springer Verlag, Vol. 2724, 2003, pp. 2488-2500.

[12] B. Suri and S. Singhal, “Implementing Ant Colony Optimization” for Test Case Selection and Prioritization”, International Journal on Computer Science and Engineering

[13] H. LI , C. Peng LAM, “An Ant Colony Optimization Approach to Test Sequence Generation for Statebased Software Testing,” In the Proceedings of the Fifth International Conference on Quality Software, September 2005, pp.255-264.

[14] K. Ayari, S. Bouktif, G. Antoniol,” Automatic Mutation Test Input Data Generation via Ant Colony,” In the Proceedings of the 9th annual conference on Genetic and evolutionary computation [GECCO], London, England, July 2007.

[15] D.J. Mala and V. Mohan, “IntelligenTester – Software Test Sequence Optimization Using Graph Based Intelligent Search Agent,” In the Proceedings of International Conference on Computational Intelligence and Multimedia Applications [ICCIMA], 2007, pp. 22-27.

[16] K. Li, Z. Yang, “Generating Method of Pair-wise Covering Test Data Based on ACO,” In the Proceedings of International Workshop on Educational Technology & International Workshop on Geoscience and Remote Sensing, 2008, pp. 776-779.

[17] D.J. Mala, M. Kamalapriya, R. Shobhana, V.Mohan, “A Non-Pheromone based Intelligent Swarm Optimization

Technique in Software Test Suite Optimization," In the Proceedings of IAMA, 2009.
[18] X. Chen, Q. Gu, A. Li, D. Chen, " Variable Strength Interaction Testing with an Ant Colony System Approach," In the Proceedings of Software Engineering Conference, APSEC '09, Penang, Asia-Pacific, Dec 2009, pp. 160-167.
[19] X. Chen, Q. Gu, A. Li, D. Chen, "Building Prioritized Pairwise Interaction Test Suites with Ant Colony Optimization," In the Proceedings of Ninth International Conference on Quality Software [QSIC], 2009, pp. 347-352.
[20] P. R. Srivastava, and V. K. Rai ,"An ant colony optimization approach to test sequence generation for control flow based software testing," In the Proceedings of 3rd International Conference on Information Systems, Technology and Management (ICISTM'09), Springer Berlin Heidelberg, March 2009, pp. 345-346.
[21] P. R. Srivastava, K. Baby, and G Raghurama, "An Approach of Optimal Path Generation using Ant Colony Optimization," In the Proceedings of TENCON 2009, IEEE Press, 2009, pp. 1-6.
[22] P.R. Srivastava, N. Jose, S. Barade, D. Ghosh, "Optimized Test Sequence Generation from Usage Models using Ant Colony Optimization," International Journal of Software Engineering & Applications (IJSEA), Vol. 2, No. 2, April 2010, pp. 14-28.
[23] P.R. Srivastava, K. Baby, "Automated Software Testing Using Metaheuristic Technique Based on an Ant Colony Optimization,"
International Symposium on Electronic System Design (ISED), Bhubaneshwar, India, Dec 2010, pp. 235-240.
[24] P.R. Srivastava, "Structured Testing Using Ant Colony Optimization," In the Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia (IITM'10), ACM Press, Dec 2010, pp. 203-207.
[25] P.R. Srivastava, V. Ramachandran, M. Kumar, G. Alukder, V. Tiwari, P. Sharma, "Generation of test data using Meta heuristic approach," In the Proceedings of IEEE TENCON, Nov 2008.
[26] K. Li, Z. Zhang, and W. Liu, "Automatic Test Data Generation Based On Ant Colony Optimization," In the Proceedings of Fifth International Conference on Natural Computation (ICNC), IEEE Press, 2009, pp. 216-219.
[27] W. Ding, J. Kou, K. Li, Z. Yang, "An Optimization Method of Test Suite in Regression Test Model," In the Proceedings of the 2009 WRI World Congress on Software Engineering (WCSE), IEEE Computer Society Washington, DC, USA, Vol. 4, 2009, pp. 180-183.
[28] A.S. Ghiduk, "A New Software Data-Flow Testing Approach via Ant Colony Algorithms," Universal Journal of Computer Science and Engineering Technology, Vol. 1, No. 1, Oct 2010, pp. 64-72.
[29] M. Chis, "A Survey of the Evolutionary Computation Techniques for Software Engineering," 1st Chapter In Chis, M. (Ed.), Evolutionary Computation and Optimization Algorithms in Software Engineering: Applications and Techniques, 2010, pp. 1-12.
[30] Y. Singh, A. Kaur, and B. Suri, "Test Case Prioritization Using Ant Colony optimization," Association in Computing Machinery ,Newsletter ACM SIGSOFT Software Engineering Notes, New York, USA, 2010, pp 1-7.
[31] Bharti Suri, Shweta Singhal (December 2015 in CSIT), "Analyzing Test Case Selection & Prioritization using ACO"
[32] Do H, Mirarab S, Tahvildari L, Rothermel G (April 2008 in ACM Transactions on Software Engineering and Methodology (TOSEM), "A safe, efficient regression test selection technique"
[33] Vineeta Mishra, Mohd Haroon, Sish Ahmad (2017, IEEE),"Software test Suite Minimization using Genetic Algorithm"
[34] Bharti Suri, Shweta Singhal (CSIT 3(2-4):143-154, December 2015), "Evolved regression test suite selection using BCO and GA and empirical comparison with ACO"
[35] Kavita Tewani (Computer Modelling & New Technologies 2017 21(2) 69-70), "Ant colony optimization algorithm: advantages, applications and challenges "