.

Test Case Selection Technique implementation using Ant Colony Optimization to reduce the execution time of Regression Testing

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Technology

in

Computer Science and Applications

Submitted By

Champat Garg

(Roll No. 601303010)

Under the supervision of:

Vineeta Bassi

Assistant Professor

(CSED)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"***Test Case Selection Technique implementation using Ant Colony Optimization to reduce the execution time of Regression Testing***"**, in partial fulfillment of the requirements for the award of degree of Master of Technology in *Computer Science and Applications* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Ms. Vineeta Bassi* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

champat

Signature:

(Champat Garg)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Vineeta Bassi)

Assistant Professor

CSED

Countersigned by

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Thapar University

Patiala

(Dr. S. S. Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

Abstract

Software testing is an investigation conducted to provide stakeholders with information regarding the quality of the product or service under test. Software testing is especially important for guaranteeing software quality in associations. In fact, the quality of test suite plays an important role for the success of software testing. Regression testing, which is a type of software testing, is very expensive which is to be executing in time and resource constrained environment. Because of time and cost constraint, it's not possible to complete extensive regression testing. There are various techniques which are used to solve the problem of time and cost constraints. Test Case Selection is one of the technique which helps in reducing the number of test cases by selecting only those test cases from the test suite which can detect all those faults which were detected by the whole test. There is an extraordinary desire in the field of Swarm Intelligence as it generally comes with a challenging task. Ant Colony Optimization (ACO) is one of challenging techniques for its provision in software testing.ACO is efficient algorithm to implement Test Case Selection technique by selecting test cases with higher probability. Our main objective of thesis is to reduce the execution time of Regression testing with the help of Ant Colony Optimization.

In this research we are modifying the previous equation of probability calculate used in ACO up to now to get better results in case of execution time of Regression testing. We are implementing ACO using both previous and modified equation on some test cases and doing comparison of the results generated by them.

Programming language used for implementing the Ant Colony Optimization algorithm is C# developed by Microsoft.

ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to **Ms. Vineeta Bassi** Assistant Professor, Computer Science and Engineering Department, Thapar University, Patiala for her patient guidance and support throughout this report. I am truly very fortunate to have the opportunity to work with her.

I am also thankful to our **Head of the Department**, **Dr. Deepak Garg**, entire faculty and staff of Computer Science and Engineering Department and then friends who devoted their valuable time and helped me in all possible ways towards successful completion of this work. I thank all those who have contributed directly or indirectly to this work.

Lastly, I would also like to thank my parents for their years of unyielding love and encourage. They have always wanted the best for me and I admire their determination and sacrifice.

Champat Garg

(601303010)

List of Contents Page No.

List of Figures Page No.

List of Tables Page No.

CHAPTER 1

INTRODUCTION

In this chapter we are explaining about Software Development Process, Software Testing and Regression testing. Then we are explaining Test Case Selection technique. Ant colony optimization is used to reduce the execution time of regression testing. Ant colony optimization is explained with the help of example. It is originated from the behavior of ants which secrete pheromone from their body while moving on their path. Then we explained the objective of our work.

1.1 Software Development Process

For the development of a complete software, a whole process in well define structure form is designed. This whole process consists of different phases which are required in development of software. Firstly the requirement phase that is used to gather the requirements from customer. After that the design phase is used to design the software. Design phase is followed by the coding phase in which code is written for the software. After that code is tested. There are various types of testing which are used to test the code. At last maintenance phase comes which is very important for the effectiveness of the software. In this phase considerable modification is done in software for the purpose of add a new functionality or enhance an existed functionality .These modifications can affect the other components of the software which are already tested during testing phase. To prevent the impact of changes to already tested code Regression testing comes into play. Test suite is a set of conditions which are tested in software to check any faults. In regression testing whenever any kind of modification is done in software test suites are run again.

1.2 Software Testing

Today word technology is being used by all the fields or daily activity either is a productive or related to our profession or another activity in our life like business related application or tool, organization, study system, entertainment (video games, television, score board), house using articles(fridge, washing machine) etc. To make our end user happy, user friendly, easy to use and satisfaction of them using software our primary goal is to provide better the quality of software without error instead of quality [1]. Software testing plays an important role to meet our goal considering user satisfaction, thus we need to make sure software testing is being performed in any scenario. For conducting the software testing, time and costs constraints must be considered.

Meaning or purpose of software testing can be considered as the flow to verifying and validating a process or program (include number of processes) of any of the application or product.

Meeting the criteria as per the defined and decided requirements during development working results are expected,

- Same characteristics also can be applicable and implemented.
- Satisfies the needs of stakeholders.

Purpose of software testing is to provide the right and actual resultant information about the features and product quality along with the quality of service. In other words we can consider it as defined flow of verifying and validating a software program or a product or an application [2]. Software testing allows the organization to appreciate the quality of software and understand the risks of software implementation.

1.2.1 Regression Testing

- To test the code there are different types of testing among which regression testing is one of it. After the completion of coding changes can be performed in the code.

- Software undergoes changes in order to fix the defects or enhance the existing functionality, or for adding new functionality.
- When a system is maintained, features that are already tested may be impacted by the changes made to the system [3]. So we need to assure that such changes did not introduce new faults that are the reason why regression testing is performed, by rerunning previously executed tests. The need to retest the test-suite arises for the purpose of maintaining the software after any changes is made.

1.3 Test case Selection

For testing purpose a number of test cases are generated, it is not easy to test all the test cases. Test case selection is a process that helps in reducing the number of test cases. Subset of the test cases is selected from the test suite in such a way that they can achieve the goal of detecting all those faults which were detected by the whole test suite. Once we are done with the selection of test cases, we must execute it to find the bugs or faults. It is very efficient method for reducing the cost in regression testing [4,18]. As regression testing is very costly due to rerunning all the test cases. So the test case selection technique helps to select an appropriate number of test cases from a test suite that will reveal us faults within minimum time after the program is modified [5,19].There are various algorithms through which Test case selection technique can be implemented for e.g. Biogeography Based Optimization(BBO), Ant Colony Optimization(ACO) [6].

1.4 Ant Colony Optimization

Ant Colony Optimization (ACO) includes a set of instructions, based on search algorithms of artificial intelligence for finding solutions to combinatorial optimization problems [7]. It is one of the effective metaheuristic techniques for solving combinatorial optimization problems. It is inspired from the natural ants [17].

Entomologists who study insects have done lot of study on ants. After study they have found that ants can do lot of things through their colony brain. Even though ants are

blind and small in size, they have the ability to find the shortest route for food source. They remain in touch with each other with the help of antennas and pheromone liquid. Pheromone is a chemical which ants keep depositing on the path returning to the nest after getting food. Stigmergy is the process in which through the environment, members of a population indirectly communicate with each other for e.g. during the foraging process ants communicate with each other indirectly by depositing pheromone trails on the ground and therefore influence other ants decision to move towards pheromone deposited path instead of walk randomly [8,20].Pheromone also start to evaporate with time. So in the longest path pheromone will evaporate more because it takes more time by ants to travel. Due to which longest path has low pheromone concentration. Similarly shortest path will have high pheromone concentration. That's why ant start following the shortest path and eventually all ants will lead to the shortest path.

ACO is a population-based search method because it involves multiple agents (i.e. ants) which take participate in searching the solution. That's why it is proved as a very effective technique in solving many combinatorial optimization problems. ACO consists of two processes namely Pheromone deposition and pheromone trail evaporation. Pheromone deposition is the phenomenon of ants adding the pheromone on all paths they follow. Pheromone trail evaporation means diminishing the amount of pheromone deposited on every path with respect to time.

Execution of ACO algorithm can be described in two steps [4]:

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

b) Updating of the pheromone information.

1.4.1 Example of ACO

Here we are explaining an example of ACO using diagrams. In our example there are 3 paths P1, P2 and P3 on which ants can move and P2 is the shortest path. In beginning ants will move randomly and deposit pheromones as shown in Figure 1.1.

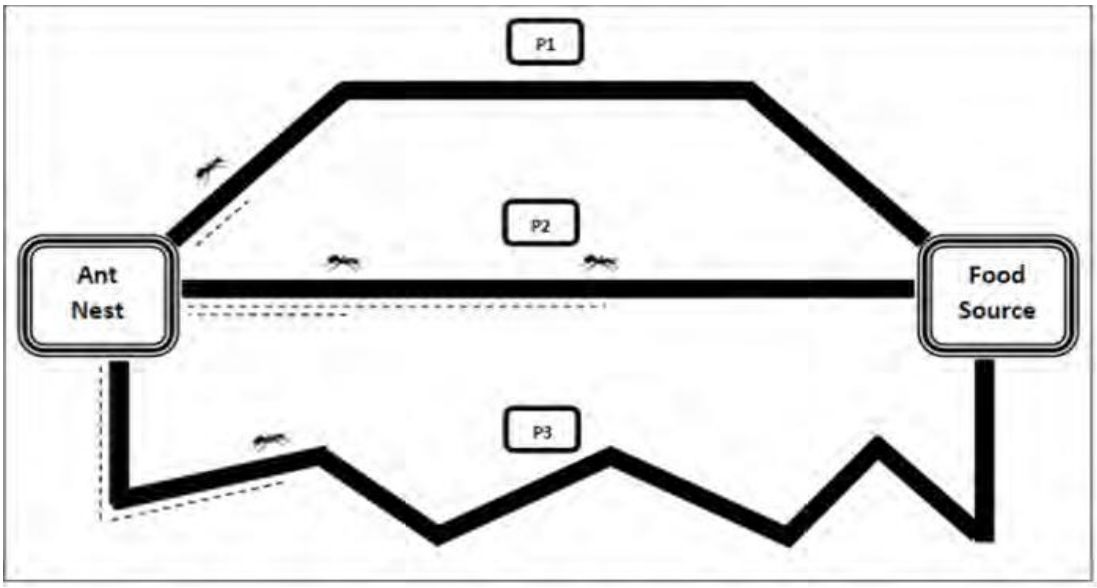

Figure 1.1: Working of ACO, initially 4 ants start from the nest [9].

The process of finding shortest path has begun. Now, some more ants have started to move and more pheromone has deposited on path P2. It is shown in Figure 1.2.

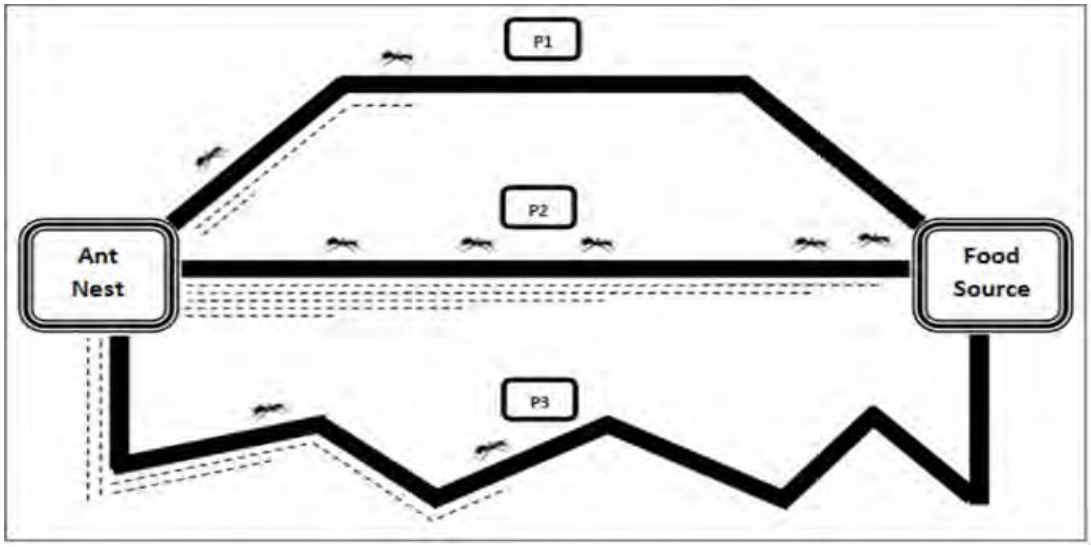

Figure 1.2: Working of ACO, More ants deposit more pheromone on various paths [9].

Some more ants started from nest and followed the pheromone with more concentration, which make more deposition of pheromone on shortest path. This is shown in Figure 1.3.

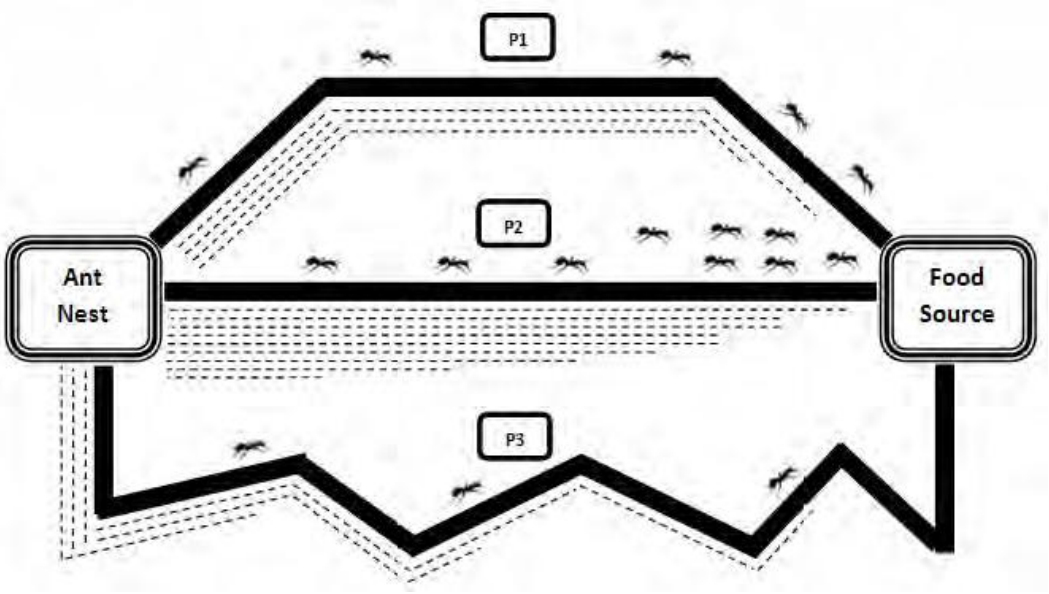

Figure 1.3: Working of ACO, some more ants start from the ant nest, now converging towards the shorter path [9].

Finally all ants will start to move on shortest path P2 which is shown in Figure 1.4.

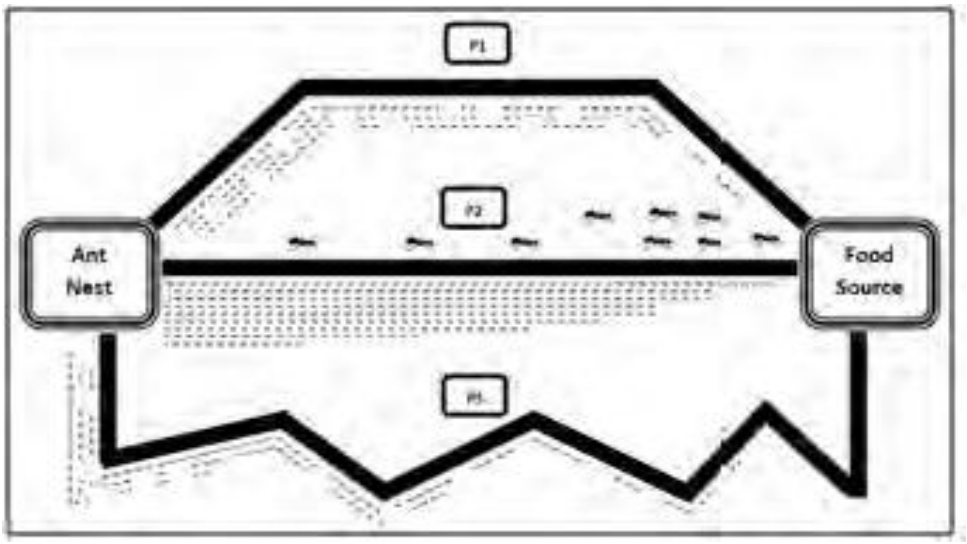

Figure 1.4 Working of ACO, finally all the ants converge to follow the shortest path [9].

Chapter 2

Literature Survey

Suri Bharti, Singhal Shweta *et al* (2011)

This paper [9] explained basic concept of ant colony optimization and told about software testing. Then software testing process is explained by using flowchart for it. Distribution of ACO in various processes of software testing is done in which 57% of ACO is applied on Test Data generation, 19%-19% is applied to path sequence generation and optimization of test data and 5% is applied to survey of evolutionary testing. Percentage of artifacts utilization and percentage types of input taken is also explained in this paper.

Chengying Mao, YuXinxin, Chen Jinfu *et al* (2012).

This paper [10] improves the Ant Colony Optimization to enhance the quality of test data so that it can cover the faults as soon as possible. Some rules have been defined again to meet the objectives of the global rule, the local transfer rule and pheromone update rule. The criterion used is the branch coverage. They included the term ant fitness. Ant fitness is checked while searching for next test case. Modified Ant Colony Optimization achieves better results than the Simulated Annealing and Genetic Algorithm.

Ding Rui, Feng Xian bin, Li Shuping, Dong Hong bin e*t al* (2012)

This paper [11] introduced an algorithm by applying particle swarm optimization into genetic algorithm. Steps they followed for their algorithm are like first initial population is created. Then families are divided by fitness value, after that cross over in family is done. Then mutation in cross over families is done to satisfy terminal condition. Their proposed work provides high efficiency in searching software testing use cases.

Gokalp Osman and Ugur Aybars *et al* (2012).

This paper [12] reformed the original Ant Colony Algorithm by employing crossover mechanism in genetic algorithm. They used four ant colonies rather than one colony. Then pheromone table is generated for each colony. Then crossover is done in these pheromone tables so as to get best pheromone tables with best solution are chosen. The reformed algorithm is tested using Traveling Salesman Problem. The result of the reformed algorithm outperforms the original Ant Colony Algorithm.

LatiuGenianaIoana, Cret Octavian Augustin, Vcariu Lucia *et al* (2012)

This paper [13] used the evolutionary algorithms to generate path test data .The evolutionary algorithms are namely Genetic Algorithm (GA), Simulated Annealing (SA) and Particle Swarm Optimization (PSO).The result derived from those algorithms are compared to find out which algorithms produces better quality of test data. The result shows that Simulated Annealing produces better quality of test data than Particle Swarm Optimization and Genetic Algorithm.

Yi Minjie *et al* (2012)

This Paper [14] proposed an algorithm which is used for path oriented testing .It is a hybrid of two algorithms Ant Colony System Algorithm and Genetic Algorithm. This hybrid algorithm is used to produce path oriented test data. It is implemented to solve the Triangle discrimination problem. After that comparison of the results produced by the hybrid algorithm and genetic algorithm is done .After comparison hybrid algorithm is found more efficient than genetic algorithm.

Gupta Nirmal Kumar and Rohit Mukesh Kumar *et al* (2013)

In their work, [3] build genetic algorithm for structural testing to generate more suitable test cases. They decrease the number of test cases which are unfeasible with the help of genetic algorithm and making the genetic algorithm more efficient by producing those test cases which are feasible. They check infeasibility of test case on basis of dependency of test cases and run time exceptions. They showed that test cases

generated with such strategy are of higher quality as such test cases will not be trapped in unwanted paths.

Gurinder Singh and Dinesh Gupta *et al* (2013)

This Paper [4] has proposed a new test case reduction hybrid technique based on Genetic Algorithms and Ant Colony Optimization. They first implemented ACO to generate various paths then output of these paths is provided to genetic algorithm to do selection, crossover and mutation so as to get better result. In initial iteration this hybrid technique gives better results. It provides positive feedback and it can lead to better solutions in optimum time.

K. Karnavel *et al* (2013)

In this Paper [15] Bee Colony Optimization (BCO) algorithm is used which performs regression testing. The fault coverage is included within the BCO algorithm for covering the fault in minimum execution time. After using this approach in testing the software application, the execution time and cost of testing is found to be reduced.

Daniel Di *et al* (2013)

This Paper [2] discussed an industrial case study on the Coverage-Based Test Case Prioritization with real regression faults. Because of the limitations of time or resource rerunning all the test cases again during regression testing is found to be very time consuming and inefficient. And with increasing the number of test cases its performance degrades. Therefore, strategies for regression testing to reduce the number of test suit choose/select appropriate test cases which are best suitable then make a sequence or synchronize them on the basis of priority. They have provided answers to many research questions such as: How granularity of coverage criteria when used in the analysis for test case prioritization affects the rate of fault detection? etc. Four different prioritization techniques are examined and evaluated. The coverage-based prioritization techniques are namely: Total Coverage, Additional Coverage, Total Coverage of Modified Code, and Additional Coverage of Modified Code. The result

showed that the additional coverage with finer grained coverage criteria outperformed all other techniques.

KevilienuoKire and Neha Malhotra *et al* (2014)

This Paper [1] provides ideas of test case selection based on Ant Colony Optimization which can be helpful for the researchers who are working on software testing. This paper provides the comparison of ACO Algorithm with GA, SA, and PSO algorithms and provided the ACO is giving better results in many complex combinatorial optimization problems especially travelling salesperson problem.

Neha Sethi *et al* (2014)

In this paper [16] authors implemented Ant colony optimization and done a modification in formula to calculate probability so as to get better results. They include execution time instead of using errors covered only. Execution time is inversely proportional to the formula of probability. Then they compared results of execution time and Average percentage fault detection of best path found in traditional ACO and in modified ACO for test case selection and found that modified ACO is providing better results.

Chapter 3

Problem Statement

Software maintenance is very important phase of the Software development process. During Software development, requirements always keep changing due to which modifications are done in the software. Regression testing participates to software maintenance as it tests the software when an updating is made to the software. Because of time and cost constraint, it's not possible to complete extensive regression testing. There are various techniques which are used to solve the problem of time and cost constraints. Test Case Selection is one of the technique which helps in reducing the number of test cases by selecting only those test cases from the test suite which can detect all those faults which were detected by the whole test. Ant Colony Optimization is a very effective algorithm to implement the Test Case Selection technique. Based on the probability ACO select the test cases.

In ACO equation to calculate probability which is used up to now selects that test case which covers maximum faults as compare to other test cases. But problem is that it can be a case that the test which covers maximum errors contains those faults which are already discovered by previously selected test cases. In such cases choosing of such test case can increase the execution time. In this research work we are performing some modification on the equation to overcome this problem.

Objective

The objective of this thesis is to reduce the execution time of regression testing by implementing Test Case Selection technique using ACO.

Chapter 4

Proposed work

In this chapter we are explaining how we solved the problem mentioned in Chapter 3.Block diagram of ACO for test case selection is given. Steps of implementation and both previous and modified equation used are also given. Main modules built for implementation are explained in brief. Flow chart of Ant colony optimization is also given.

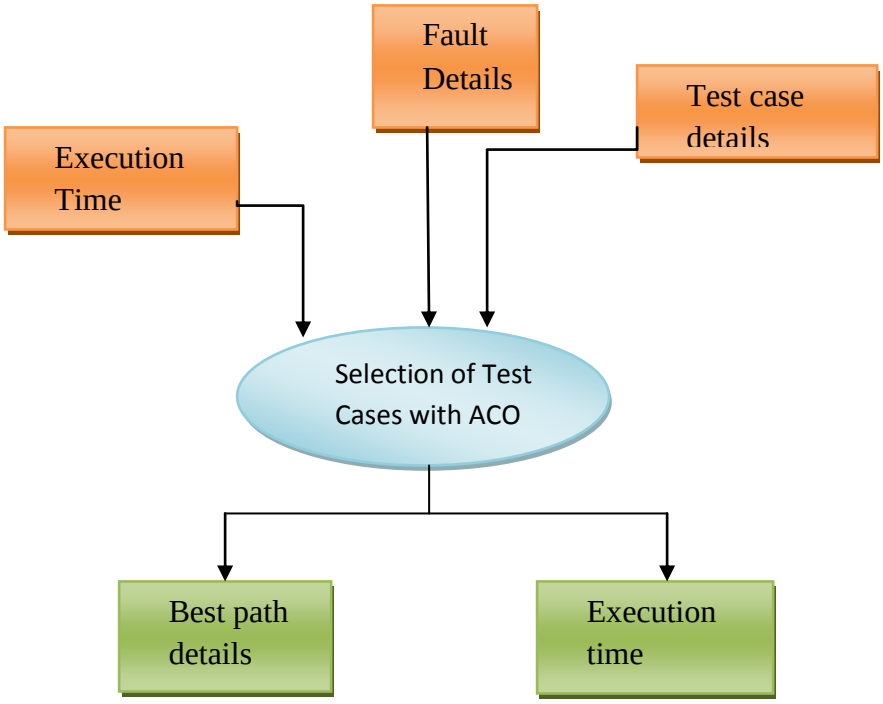

Figure 4.1: Block diagram for ACO System

4.1 Steps for implementation

Ant Colony Optimization (ACO) algorithm will work as follows:

- Input the Test Suite details.

- Initialize all the parameters with input values.
- For each ant randomly select the test case to start the tour.
- For each ant keep selecting the next test case until all faults are found.
- Selection of the next test case depends on :
    - ✓ Maximum Unique Errors found
- Terminate the search/iteration process when all faults are detected.
- For each ant k formula to update the pheromone for each state transition xy covered is as follow:

$$\tau_{xy}^{k} = (1-\rho)\tau_{xy}^{k} + L^{k}/\sigma$$

Where,

τ_{xy}^{k} Is the amount of pheromone deposit on a state transition xy

ρ Is the pheromone decrease rate

L^{k} is the total test cases covered by ant k.

and

σ is the pheromone deposition rate.

- Pheromone Updation
    - ✓ Deposit Pheromone on the each path covered by ants in their solution
    - ✓ Reduce Pheromone on each edges by the rate of decrease factor.(evaporate)
- Determine the final path.
- At each iteration selection of best path depends on :
    - ✓ Minimum Execution Time

4.2 Formulation used

In previous equation ,probability of moving from test case i to test case j is

$$\text{Probablity}_{j} = \frac{\left(\frac{\text{Errorscovered}_{j}}{\text{Executiontime}_{j}}\right)^{\beta} X\ (\text{Phermononetrail}_{i,j})^{\alpha}}{\sum_{k=1}^{n}\left(\left(\frac{\text{Errorscovered}_{k}}{\text{Executiontime}_{k}}\right)^{\beta} X\ (\text{Phermononetrail}_{i,k})^{\alpha}\right)}$$ where

$j, k \epsilon \{\text{Non visited test cases}\}$

Now formula used by us is

$$\text{Probablity}_j = \frac{\left(\frac{\text{UniqueErrorscovered}_j}{\text{Executiontime}_j}\right)^{\beta} X\ (\text{Phermononetrail}_{i,j})^{\alpha}}{\sum_{k=1}^{n}\left(\left(\frac{\text{UniqueErrorscovered}_k}{\text{Executiontime}_k}\right)^{\beta} X\ (\text{Phermononetrail}_{i,k})^{\alpha}\right)}$$ where

j, kϵ {Non visited test cases}

Some of the modules from the implementation are explained below:

1. void ant::addtestcase(int testcase)
    - The module has been defined so as to add testcase to TestcaseTable.
2. void ant::adderrors(int testcase)
    - To add errors covered by selected test cased to errorcovered table.
    - Calculate unique errors covered by each non-visited test case i.e. in this step we find out the errors for each test case which are not discovered yet.
3. int ant:: allerrorfound()
    - To check if all errors are found.
4. int ant::ChooseNextTestcase()
    - To calculate total probability of unvisited test cases.
    - Finding test case with maximum probability.
5. void ant::UpdateResult()
    - To Update the length of tour.
6. int ant::move()
    - the ant move to next town and add town ID to TestcaseTable.
7. void project::UpdateTrial()
    - Decreasing pheromone from each edge by 10%.
    - Increase pheromone trails of edges covered by ant.
8. project::project()
    - initial PheromoneObj,read map infomation from file .
9. void project::GetAnt()
    - randomly put ant into map
10. void project::StartSearch()
    - begin to find best solution
    - Find out the best solution of the step and put it into temporary variable.

11. void main()

- Execution of the implemented code starts from the Main module.
- Other modules are called from this module and the final results are displayed.

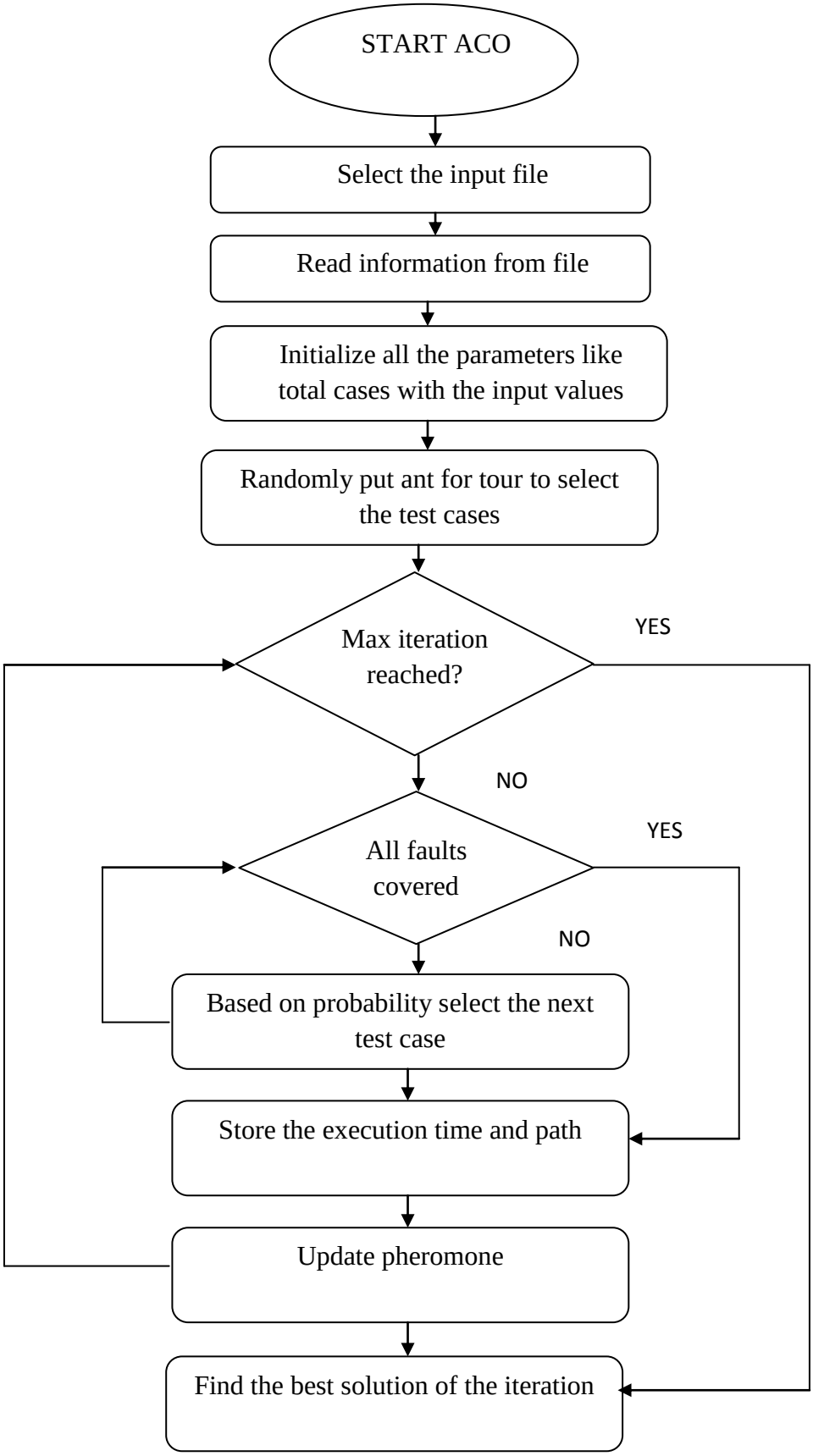

Figure 4.2 Flowchart for the proposed approach

Chapter 5

Testing and results

Ant Colony Optimization (ACO) is a very effective technique in solving combinatorial optimization problems. Based on the probability it will search its solution. In this chapter we are showing the results generated after implementing both previous and modified equation of ACO mentioned in chapter 4 on two input test suite. Each input test suite contain total number of test cases, total faults, execution time of each test case, number of faults covered by each test case. Then we are doing the comparison of results generated by both equations for each input test suite.

5.1 Results

Table 5.1 represent the first input test suite by using which we are generating the results of both equations.

Test cases/faults	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	Execution Time	No of faults
T1	*		*	*	*					*						7	5
T2			*	*	*					*	*		*		*	9	7
T3		*				*	*							*		6	4
T4	*							*	*			*				4	4
T5		*				*		*	*							6	4
T6			*	*	*		*							*		7	5
T7										*	*	*	*		*	7	5
T8			*		*			*							*	5	4
T9		*		*				*	*							4	4
T10		*					*			*	*					5	4

Table 5.1 First input test suite showing faults covered and execution time for each test case

First we are showing results of previous technique. First run of previous technique is shown in Figure 5.1.

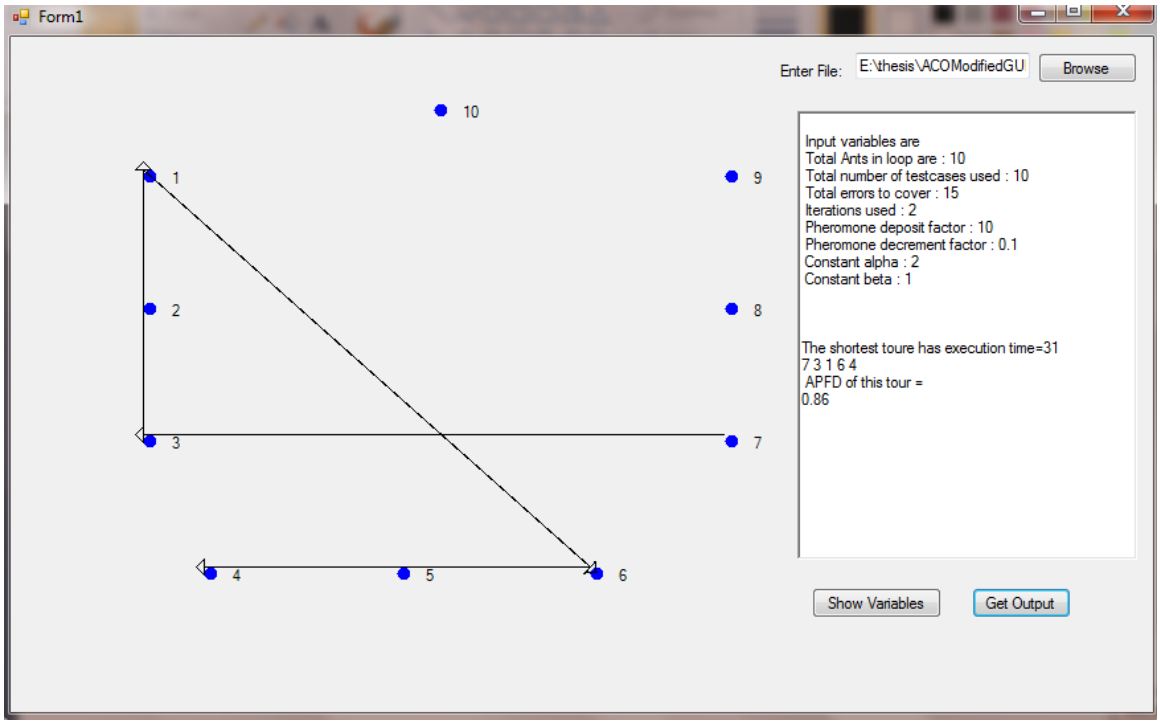

Figure 5.1 Fist run of previous technique for first test suite.

In Figure 5.1 we can see that total errors to be found are 15, number of iterations used are 2, pheromone deposition factor is 10, pheromone decrement factor is 10% and best path come in first run is 7->3->1->6->4 and execution time is 31.

Second run of previous technique is shown in Figure 5.2.

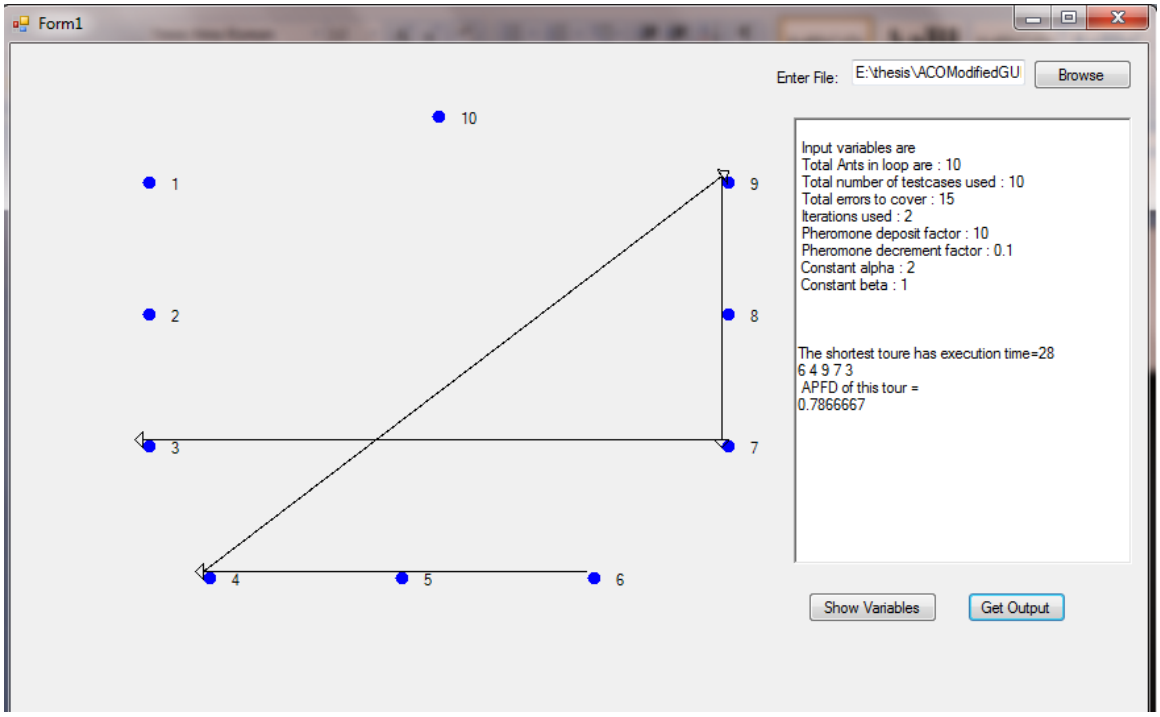

Figure 5.2 Second run of previous technique for first test suite.

After first run of previous technique we are checking result for second time. As we can see from Figure 5.2 that best path chosen during second run are 6->4->9->7->3 and execution time is 28.

Third run of previous technique is shown in Figure 5.3.

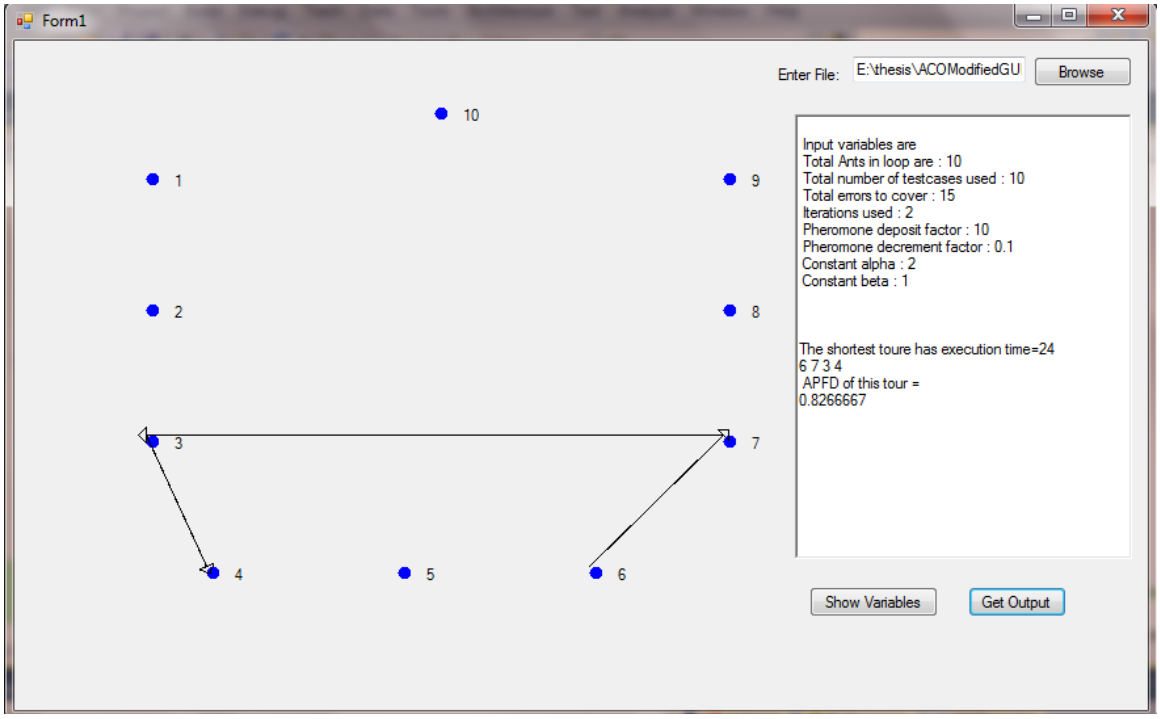

Figure 5.3 Third run of previous technique for first test suite.

After second run of previous technique we are checking result for third time. As we can see from Figure 5.3 that best path chosen during second run are 6->7->3->4 and execution time is 24.

After running the previous technique several times we are showing the results in table 5.2

Execution Time	Selected Test Cases
47	5,4,9,8,10,2,1,6
47	3,1,6,4,9,8,10,2
54	2,1,6,4,9,8,10,7,3
54	9,8,10,2,1,6,4,7,3
33	3,4,9,8,10,2
54	6,4,9,8,10,2,1,7,3
47	4,9,8,10,2,1,7,3
33	2,1,7,3,4
43	8,10,2,1,7,3,4

38	10,2,1,7,3,4
24	4,1,7,3
30	5,4,1,7,3
24	7,3,4,1
43	9,8,10,2,1,7,3

Table 5.2 Previous technique results for first test suite

So, out of these results we are selecting the best case. It means minimum execution time of previous technique is 24 for first test suite.

First run of modified work is shown in Figure 5.4.

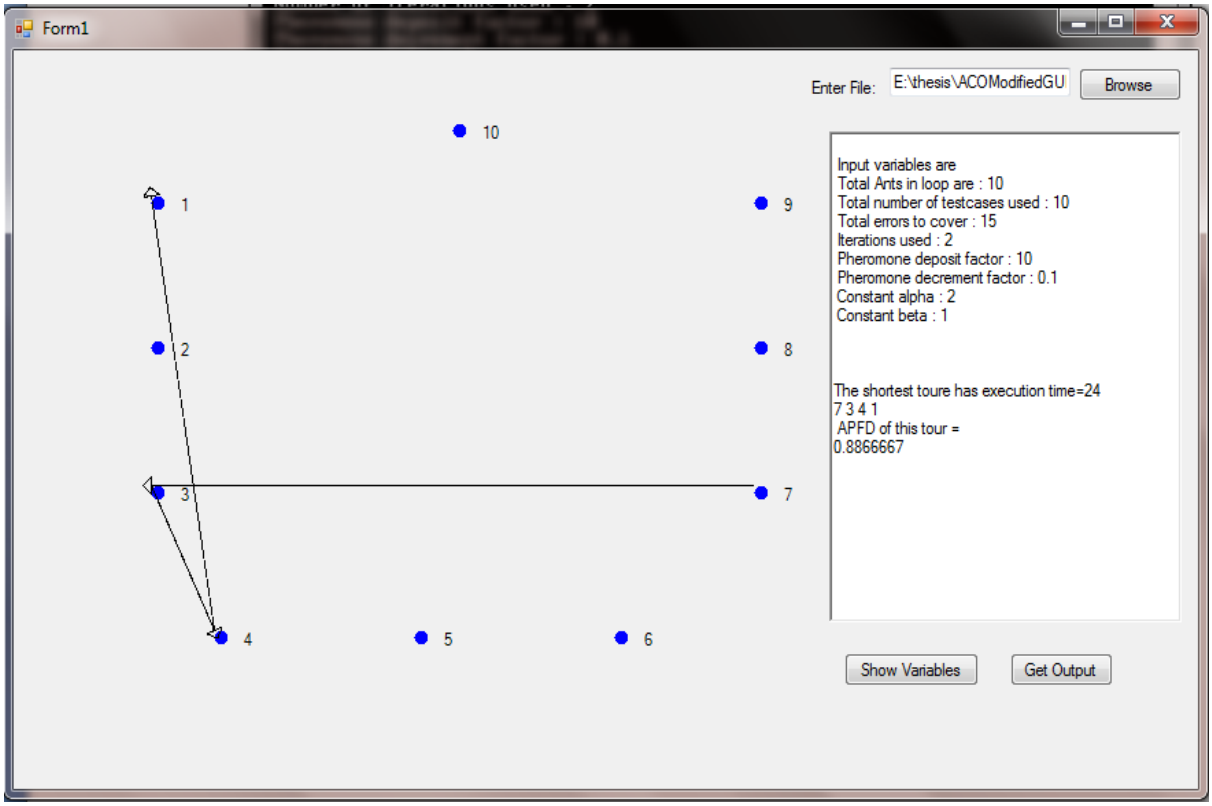

Figure 5.4 First run of modified technique for first test suite.

In Figure 5.4 we can see that for modified work total errors to be found are 15, number of iterations used are 2, pheromone deposition factor is 10, pheromone decrement factor is 10% and best path come in first run is 7->4->3->1 and execution time is 24.

Second run of modified technique is shown in Figure 5.5.

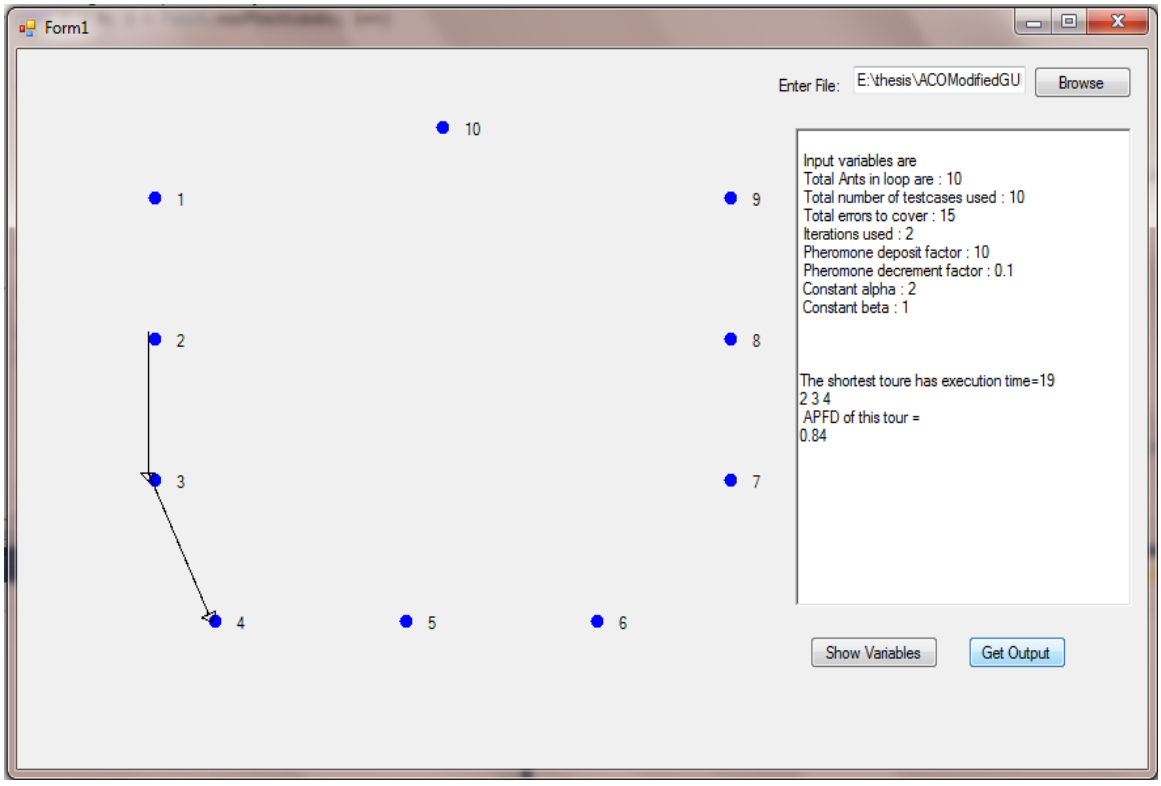

Figure 5.5 Second run of modified technique for first test suite.

After first run of modified technique we are checking result for second time. As we can see from Figure 5.5 that best path chosen during second run are 2->3->4 and execution time is 19.

Third run of modified technique is shown in Figure 5.6.

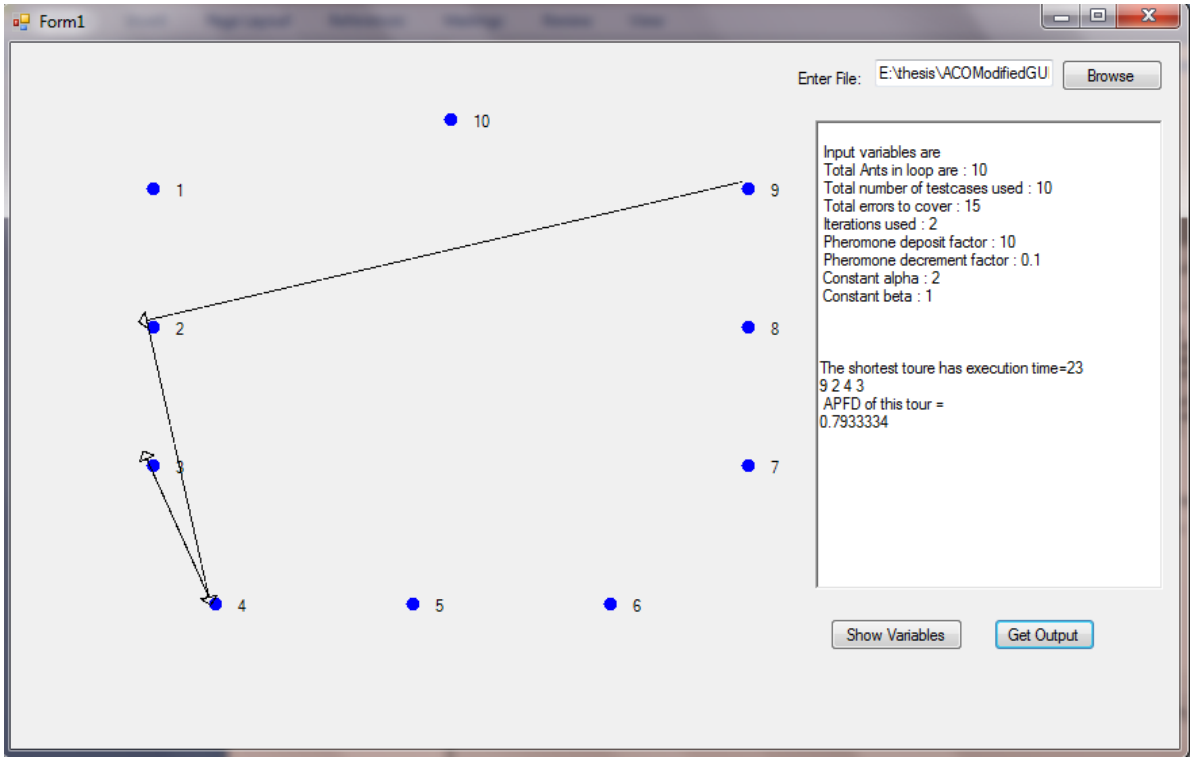

Figure 5.6 Third run of modified technique for first test suite.

After second run of modified technique we are checking result for third time. As we can see from Figure 5.6 that best path chosen during third run is 9->2->4->3 and execution time is 23.

After running the modified technique several times we are showing the results in Table 5.3

Execution Time	Selected Test Cases
28	4,3,9,7,1
24	9,7,1,3
25	5,2,4,3
24	7,1,3,9
31	8,10,4,3,9,7
33	10,4,3,9,7,1
19	2,4,3
24	7,1,3,9
25	5,2,4,3

24	9,7,1,3
31	6,4,3,1,7
29	10,4,3,1,7
24	1,3,9,7
24	3,1,7,4

Table 5.3 Modified technique results for first test suite

So, out of these results we are selecting the best case. It means minimum execution time of modified technique is 19 for first test suite.

Table 5.4 represent the second input test suite for which we are showing the results of both equations.

Test cases/faults	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	ET	FC
T1	*		*	*	*													*			7	5
T2			*	*						*	*		*		*			*			9	7
T3		*				*	*							*		*					6	5
T4	*							*	*			*					*		*		7	6
T5		*				*		*	*												6	4
T6			*		*		*							*			*				7	5
T7										*	*	*	*		*						7	5
T8			*		*			*							*						5	4
T9		*		*				*													4	3
T10		*					*			*	*										5	4
T11		*				*														*	4	3

T12				*													*	*		4	3
T13	*	*			*								*				*		*	8	6

Table 5.4 Second input test suite showing faults covered, and execution time for each test case

First we are showing results of previous technique. First run of previous technique is shown in Figure 5.7.

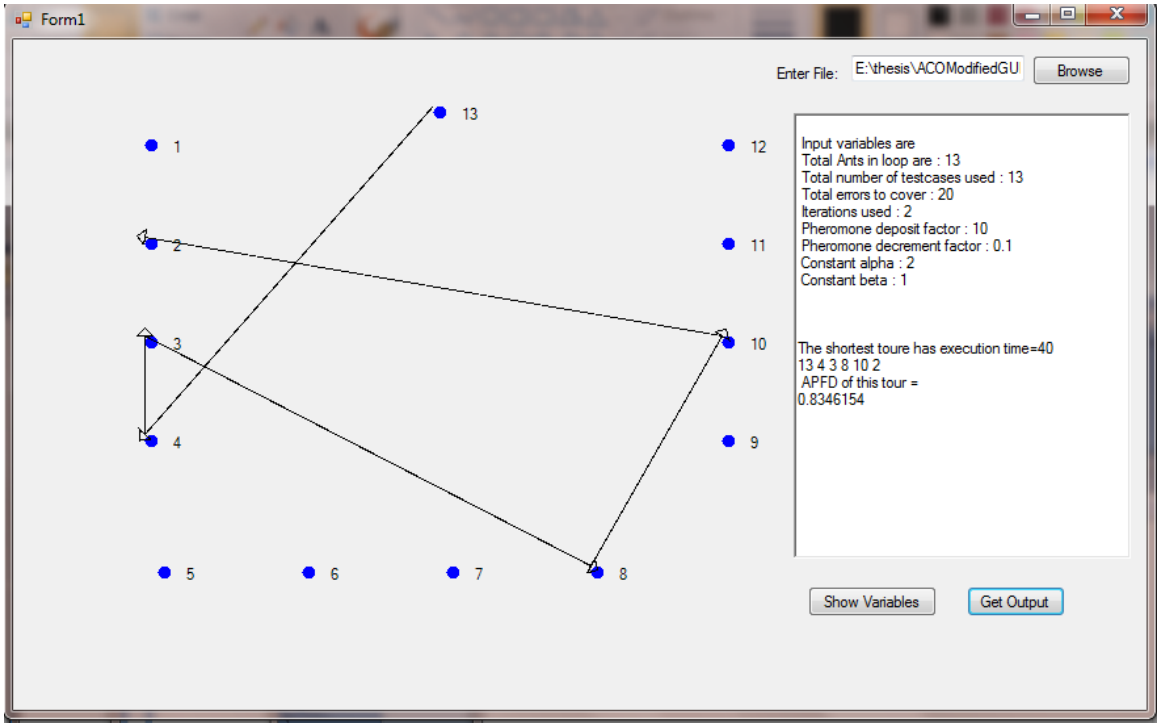

Figure 5.7 First run of previous technique for second test suite.

In Figure 5.7 we can see that total errors to be found are 20, number of iterations used are 2, pheromone deposition factor is 10, pheromone decrement factor is 10% and best path come in first run is 13->4->3->8->10->2 and execution time is 40.

Second run of previous technique is shown in Figure 5.8.

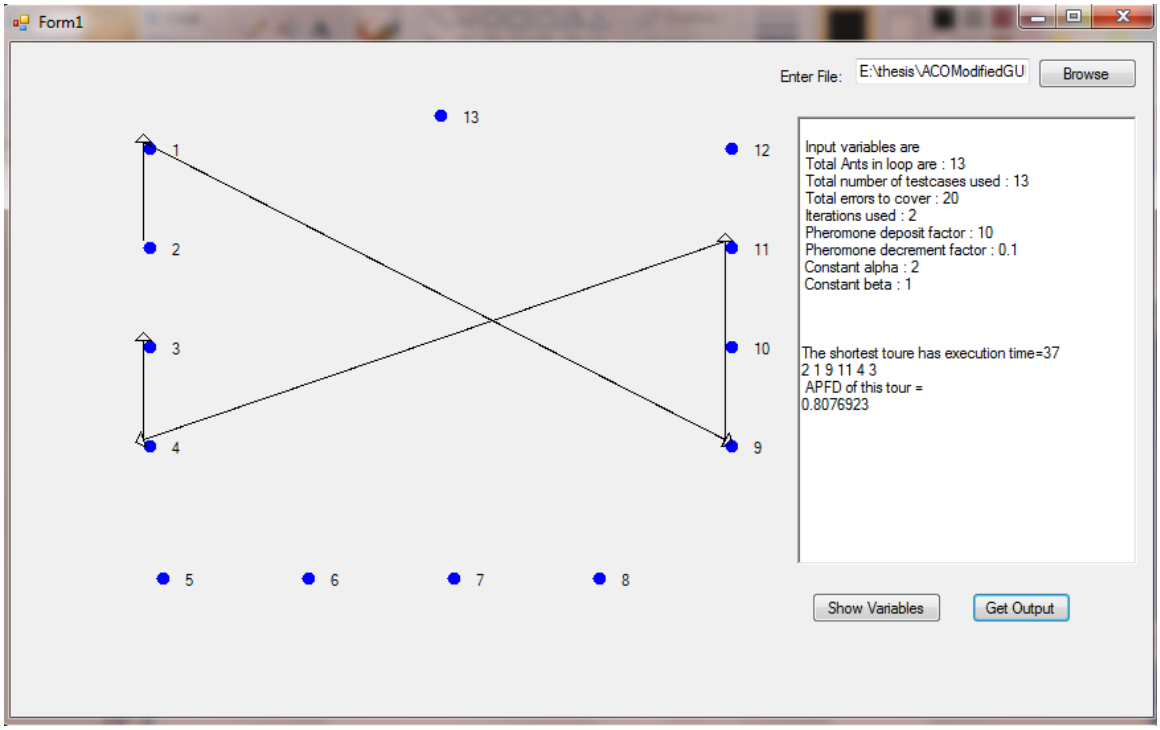

Figure 5.8 Second run of previous technique for second test suite.

After first run of previous technique we are checking result for second time. As we can see from Figure 5.8 that best path chosen during second run is 2->1->9->11->4->3 and execution time is 37.

Third run of previous technique is shown in Figure 5.9.

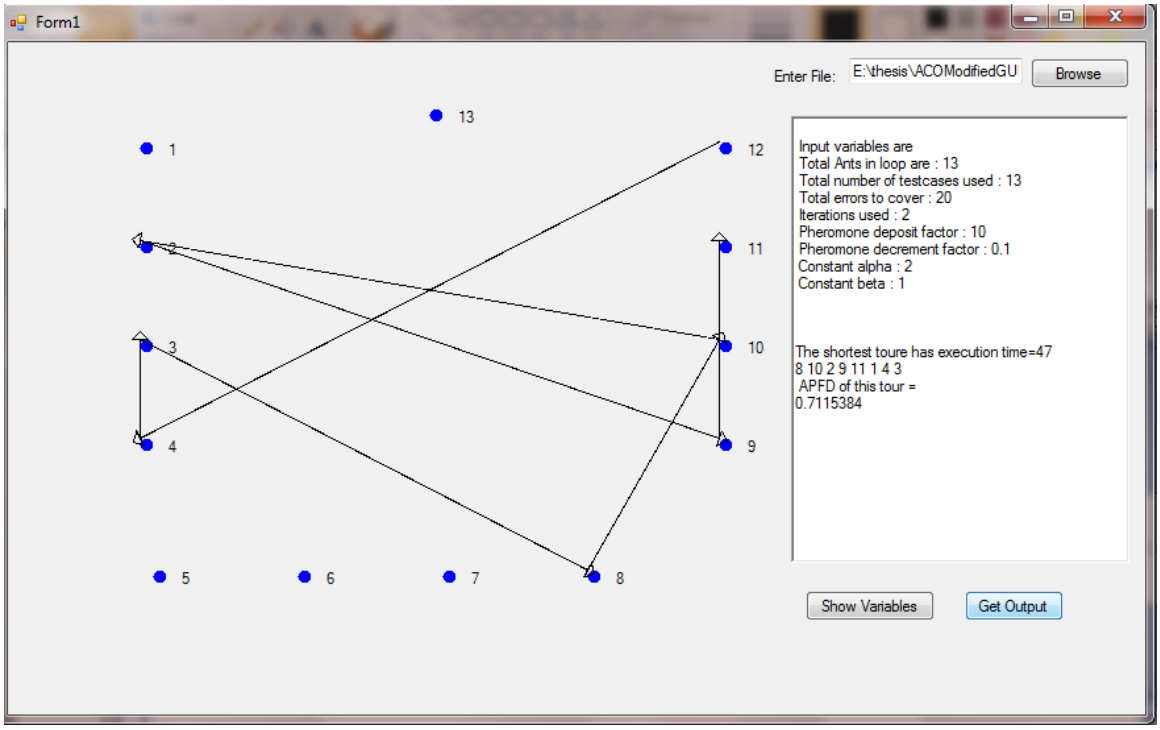

Figure 5.9 Third run of previous technique for second test suite.

After second run of previous technique we are checking result for third time. As we can see from Figure 5.9 that best path chosen during second run is 8->10->2->9->11->1->4->3 and execution time is 47.

After running the previous technique several times we are showing the results in Table 5.5

Execution Time	Selected Test Cases
44	12,4,3,8,10,2,9,11
47	8,10,2,9,11,1,4,3
43	11,1,4,3,8,10,2
47	9,11,1,4,3,8,10,2
54	7,4,3,8,10,2,1,9,11
54	6,4,3,810,2,1,9,11
47	3,8,10,2,1,4,9,11
47	2,1,4,3,8,10,9,11
53	5,4,3,8,10,2,1,9,11

42	10,2,1,9,11,4,3
37	2,1,9,11,4,3
47	4,3,1,9,11,8,10,2
54	6,4,3,1,9,11,8,10,2
47	2,1,9,11,8,10,4,3

Table 5.5 Previous technique results for second test suite

So, out of these results we are selecting the best case. It means minimum execution time of previous technique is 37 for second test suite.

First run of modified technique is shown in Figure 5.10.

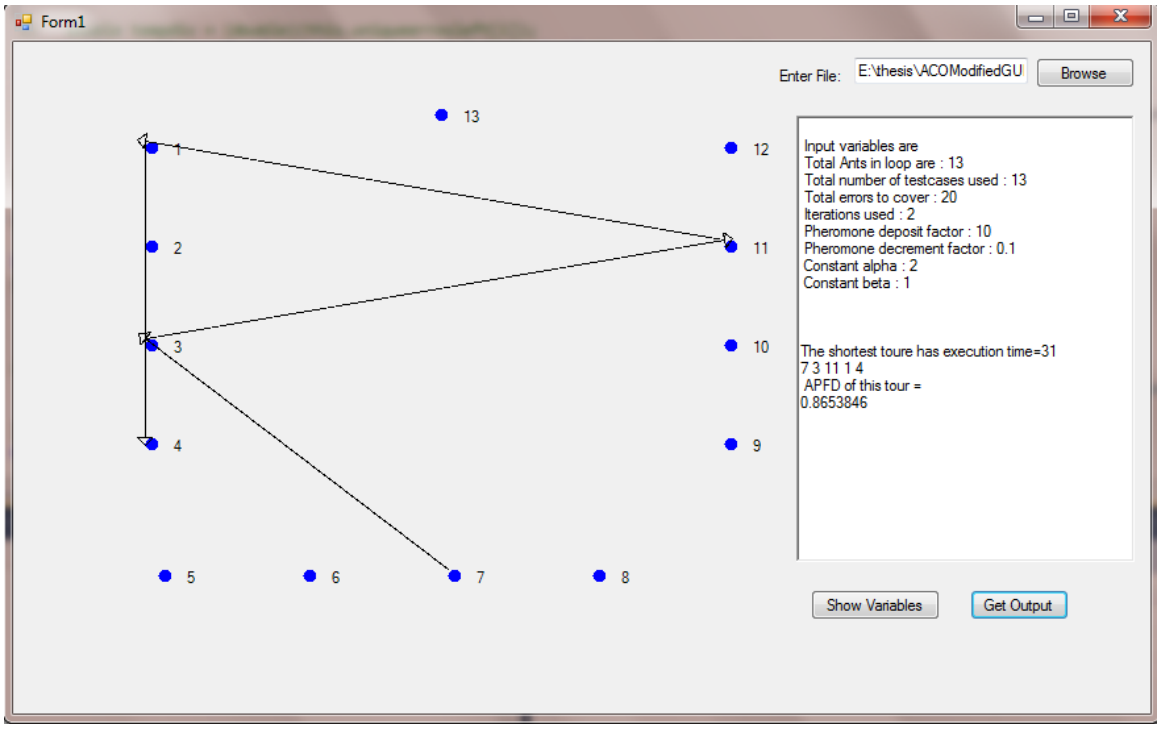

Figure 5.10 First run of modified technique for second test suite.

In Figure 5.10 we can see that total errors to be found are 20, number of iterations used are 2, pheromone deposition factor is 10, pheromone decrement factor is 10% and best path come in first run is 7->3->11->1->4 and execution time is 31.

Second run of modified technique is shown in Figure 5.11

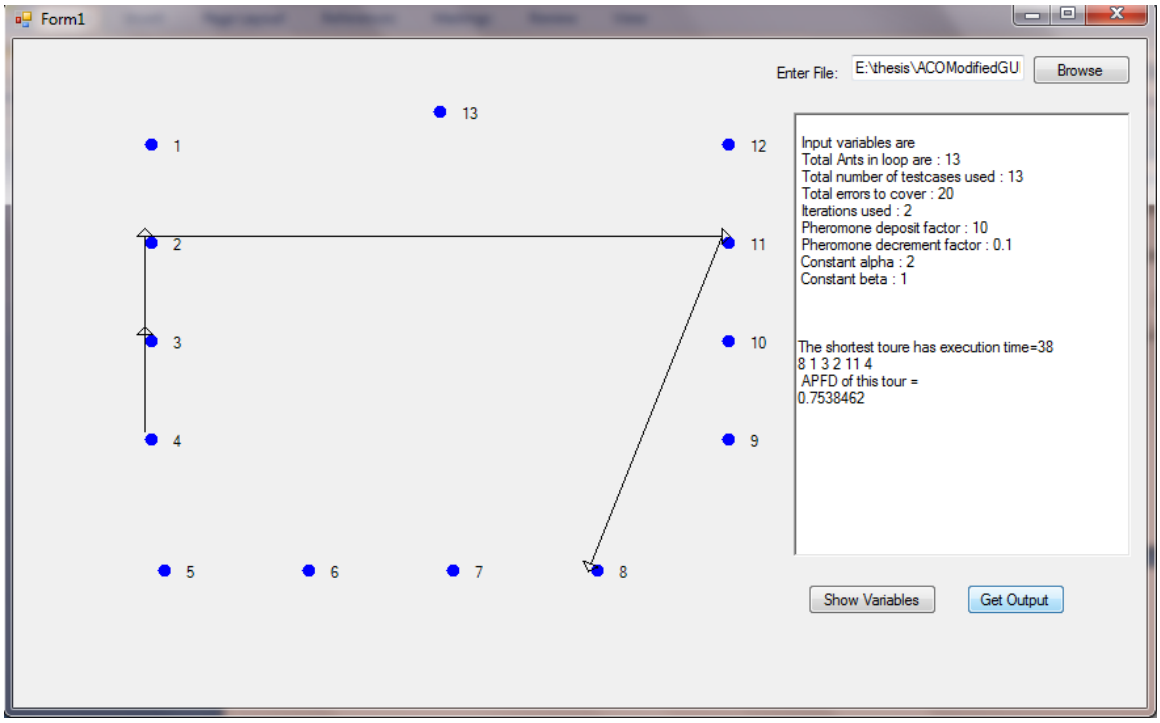

Figure 5.11 Second run of modified technique for second test suite.

After first run of modified technique we are checking result for second time. As we can see from Figure 5.11 that best path chosen during second run is 8->1->3->2->11->4 and execution time is 38.

Third run of modified technique is shown in Figure 5.12.

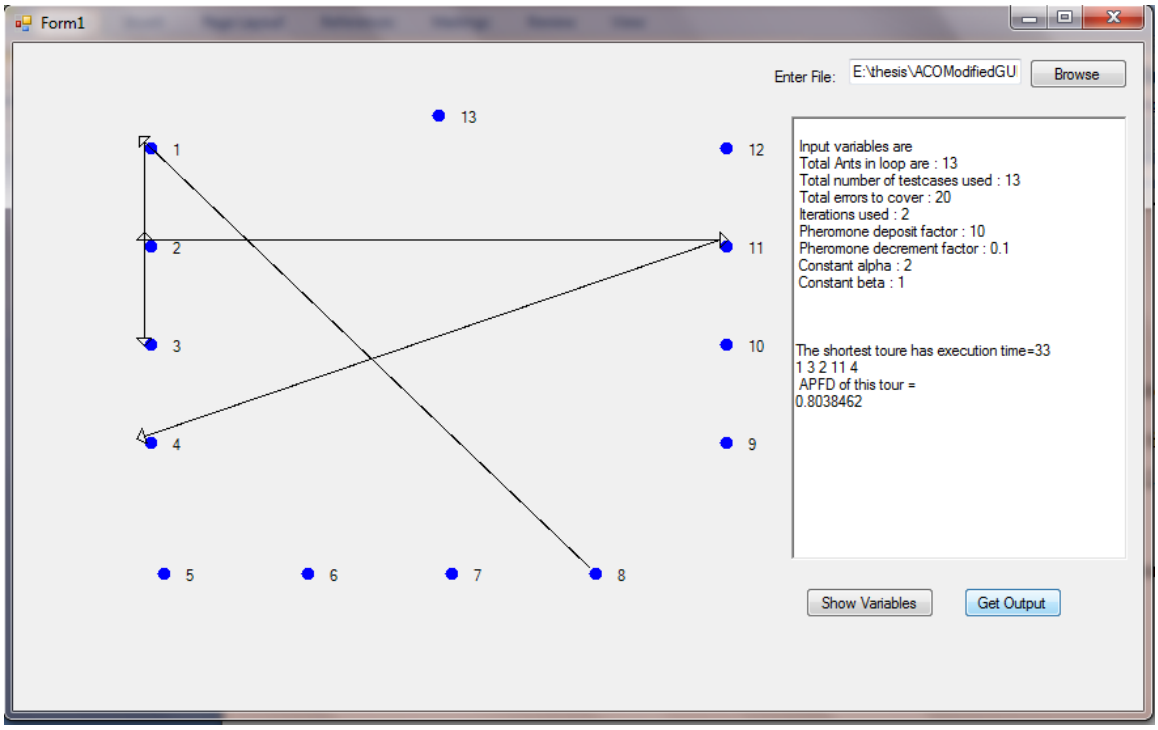

Figure 5.12 Third run of modified technique for second test suite.

After second run of modified technique we are checking result for third time. As we can see from Figure 5.12 that best path chosen during third run is 1->3->2->11->4 and execution time is 33.

After running the modified technique several times we are showing the results in Table 5.6

Execution Time	Selected Test Cases
38	8,1,3,2,11,4
33	2,11,4,1,3
39	5,2,11,4,1,3
38	8,1,3,2,11,4
33	11,4,1,3,2
37	12,3,2,11,4,1
33	1,3,2,11,4
38	10,4,1,3,2,11
40	7,3,2,11,4,1

44	6,9,7,3,2,11,4
33	1,3,2,11,4
33	4,1,3,2,11
33	3,2,11,4,1
38	10,4,1,3,2,11

Table 5.6 Modified technique results for second test suite

So, out of these results we are selecting the best case. It means minimum execution time of modified technique is 31 for second test suite.

4.2 Discussions

Comparison table of previous and modified technique best case for first test suite is shown in Table 5.7.

	Previous Technique	Modified Technique
Execution Time	24	19

Table 5.7 Comparison of previous and modified technique best case for first test suite

Graphical representation of both Table 5.2 and Table 5.3 is shown in Figure 5.13

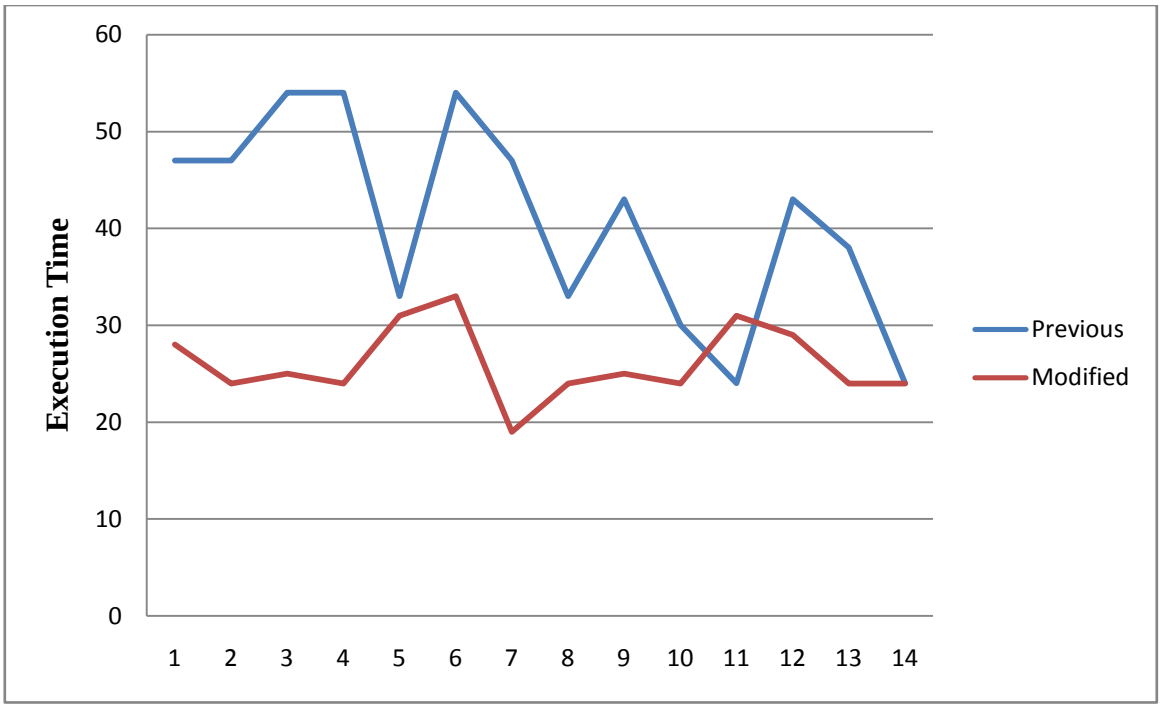

Figure 5.13 Graphical representation of results generated by both techniques for first test suite

From Figure 5.13 we first noticed that results obtained from previous technique have execution time within the range of 24-53 as compare to modified technique which have execution time within the range of 19-33.The longest execution time in previous technique is 53 as compared to modified technique which has 33.

The graphical representation of comparison Table 5.7 is shown in Figure 5.14.

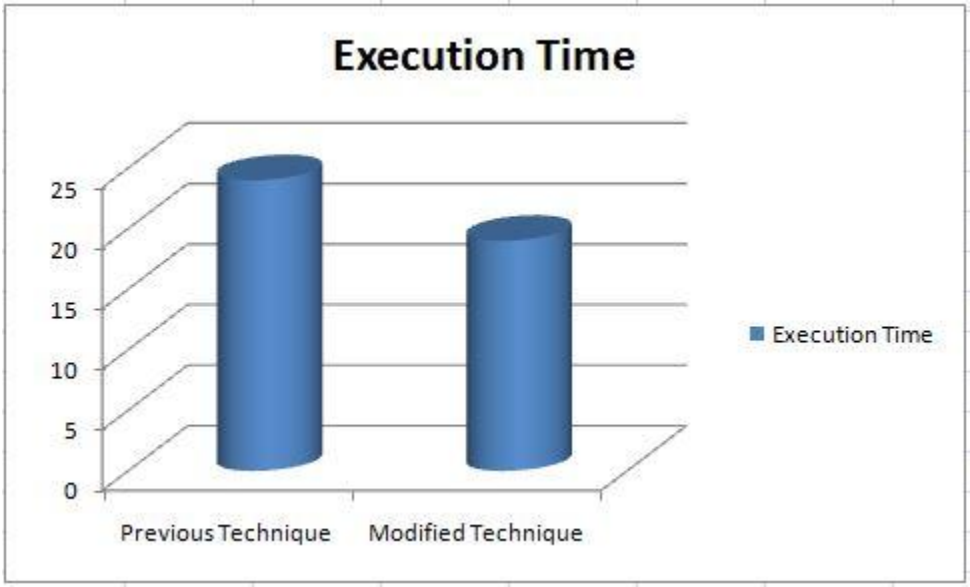

Figure.5.14 Graphical representation of comparison of both techniques best case for first test suite.

Comparison table of previous and modified technique best case for second test suite is shown in Table 5.8.

	Previous Technique	Modified Technique
Execution Time	37	31

Table 5.8 Comparison of previous and modified technique best case for second test suite

Graphical representation of both Table 5.5 and Table 5.6 is shown in Figure 5.15

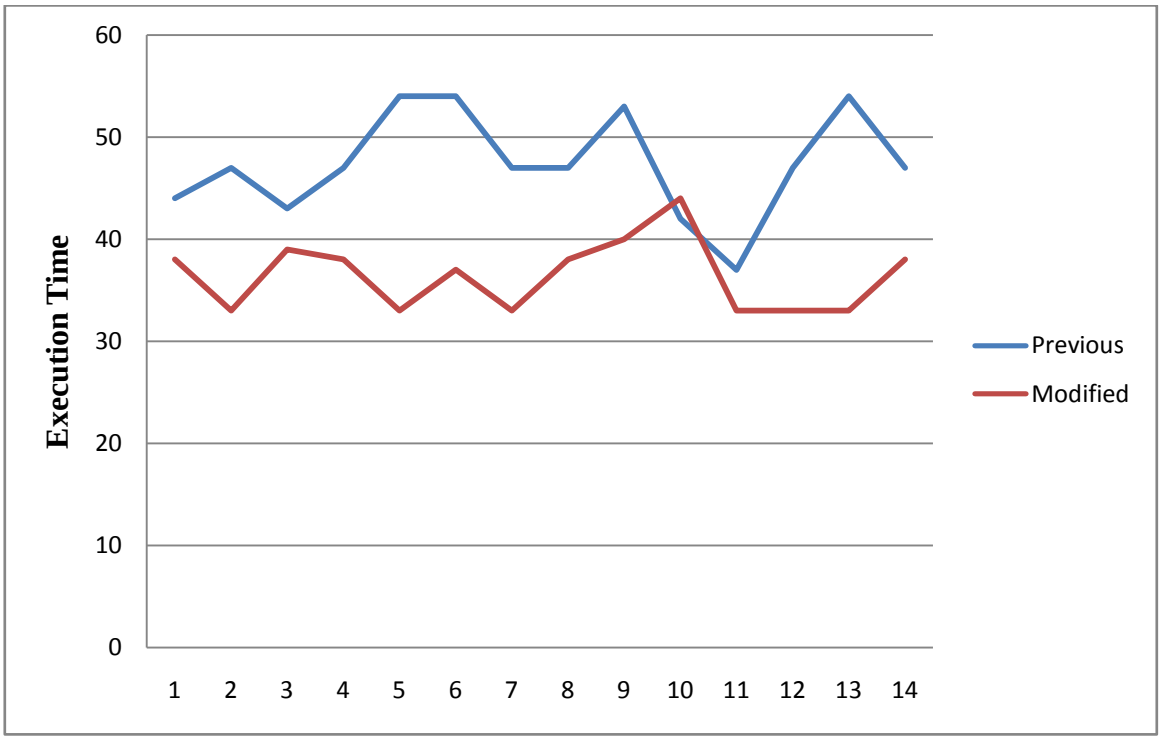

Figure 5.15 Graphical representation of results generated by both techniques for second test suite

From Figure 5.15 we first noticed that results obtained from previous technique have execution time within the range of 37-54 as compare to modified technique which have execution time within the range of 31-44.The longest execution time in previous technique is 54 as compared to modified technique which has 44.

The graphical representation of comparison Table 5.8 is shown in Figure 5.16.

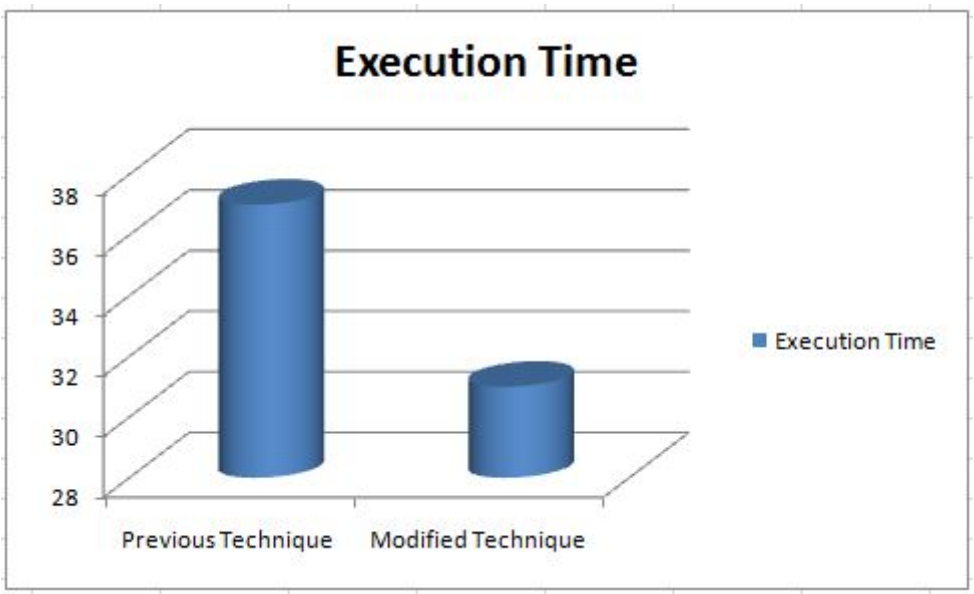

Figure 5.16 Graphical representation of comparison of both techniques best case for second test suite.

Chapter 5
Conclusion

5.1 Conclusion

Regression testing is very important for the software maintenance. To make it effective Test Case Selection technique is used. ACO is an algorithm used for implement test case Selection. ACO select the test cases based on the probability. In this thesis we introduce modified equation to calculate the probability of test case in ACO .Then we implement the both modified and previous equation which is used up to now on two test suite. After comparing the results generated by both equations we find out that modified equation gives better results in terms of minimum execution time.

5.2 Future Scope

In future more work can be done on following:

1) On other applications of Ant colony optimization like Travelling Salesman Problem.
2) Some other techniques of regression testing like test case prioritization can be modify to achieve better results.

References

[1] Kevilienuo Kire and Neha Malhotra “Study of test case selection and prioritization”, International journal of computer applications vol. 85-No. 5, pp.28-30, 2014.

[2] Daniel Di Nardo, N. A. Coverage-Based Test Case Prioritization: An Industrial Case Study. IEEE Sixth International Conference on Software Testing, Verification and Validation, (pp. 302-3011). Luembourg (2013).

[3] Gupta Nirmal Kumar and Rohil Mukesh Kumar "Improving GA based Automated Test Data Generation Technique for Object Oriented Software", IEEE International Advance Computing Conference (IACC), Ghaziabad, pp.249 – 253, 2013.

[4] Gurinder Singh and Dinesh Gupta “An Integrated Approach to test suite selection using ACO and Genetic Algorithm”, International journal of advanced research in computer science and software engineering (iiarcsse), vol 3, pp.1770-177, 2013.

[5] Priyanka Bansal, “A Critical Review on Test Case Prioritization and Optimization using Soft Computing Techniques”, 2nd International Conference on Role of Technology in Nation Building (ICRTNB-2013), pp74-77, 2013.

[6] Wang Jun, Z. Y.,” Test Case Prioritization Technique based on Genetic Algorithm”, International Conference on Internet Computing and Information Services, Hong Kong, pp. 173 – 175, 2011.

[7] Rothermal, Roland H. Untch, Chengyun Chu and Mary Jean Harrold,“Prioritizing Test Case for Regression Testing”, IEEE Transactions on Software Engineering, 2001.

[8] H. He and F. Min, "Accumulated cost based test-cost-sensitive attribute reduction," in Proceedings of the 13th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, ser. LNAI, vol. 6743, pp. 244-247,2011.

[9] Bharti Suri and Shweta Singhal, “Literature Survey of Ant Colony Optimization in Software Testing”, CSI Sixth International Conference on Software Engineering(CONSEG),Indore,pp1-7, 2012.

[10] Chengying Mao, YuXinxin, Chen Jifu, Chen Jinfu "Generating Test Data for Structural Testing Based on Ant Colony Optimization "12th International Conference on Quality Software, Xi'an, Shaanxi, pp. 98 – 101, 2012.

[11] Ding Rui, Feng Xian bin, Li Shuping, Dong Hong bin "Automatic Generation of Software Test Data Based on Hybrid Particle Swarm Genetic Algorithm", IEEE Symposium on Electrical & Electronics Engineering (EEESYM), Mudanjiang, Kuala Lumpur, pp. 670 – 673, 2012

[12] Osman Gokalp and Aybars Ugur, "Improving Performance of ACO Algorithms Using Crossover Mechanism Based on Mean of Pheromone Tables", 2012 International Symposium on Innovations in Intelligent Systems and Applications (INISTA), Trabzon, pp. 1-4, 2012.

[13] Latiu Geniana Ioana, Cret Octavian Augustin, Vcariu Lucia "Automatic Test Data Generation for Software Path Testing using Evolutionary Algorithms", International Conference on Emerging Intelligent Data and Web Technologies, Bucharest, pp.1-8, 2012.

[14] Yi Minjie,"The Research of path-oriented test data generation based on a mixed ant colony system algorithm and genetic algorithm", International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), Shanghai, pp.1-4, 2012.

[15] K. Karnavel, J. "Automated Software Testing for Application Maintenance by using Bee Colony Optimization algorithms (BCO)." Information Communication and Embedded Systems, (pp. 327-330). Chennai (2013).

[16] S. Neha, R. Shaveta and S. Paramjeet, "Ants Optimization for Minimal Test Case Selection and Prioritization as to Reduce the Cost of Regression Testing", International Journal of Computer Applications, vol 100 no 17, pp. 48-54, 2014.

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

[18] A.Pravin and Dr. S.Srinivasan, " An Efficient Algorithm for reducing the test cases which is used for performing regression testing", 2nd International Conference on Computational Techniques and Artificial Intelligence, Dubai (UAE), pp. 194-197,2013.

[19] Z. Pawlak, "Rough sets," International Journal of Computer and Information Sciences, vol. 11, pp. 341-356, 1952.
[20] M.Dorigo and C. Blum, "Ant colony optimization theory: A survey", Theoretical Computer Science, vol. 344, no. 2-3, pp. 243-278, 2005.

VIDEO PRESENTATION

https://youtu.be/HHKM-_OdUg0