Applying Ant Colony Optimization and Genetic Algorithm into Software Testing

1Ikvinderpal Singh, 2Sapandeep Kaur
1Assistant Professor, Trai Shatabdi GGS Khalsa College, Amritsar, Punjab, India
2Assistant Professor, S.R. Govt. College, Amritsar, Punjab, India

Abstract
This paper proposed software testing system by using Evolutionary techniques. In today’s scenario, Software Testing is a critical issue in software development and maintenance for increasing the quality and reliability of the software. In this paper, the Ant Colony Optimization (ACO) is used to find shortest path with help of distance value, the output of ACO is given as input to the Genetic Algorithm (GA) algorithm for the best optimal shortest path solution and contribute a lot in considerably reducing the testing cost, efforts and time of testing.

Keywords
Software testing; Ant Colony Optimization; Genetic Algorithm

I. Introduction
Software testing is an activity aimed at evaluating difference between the actual requirements and the requirements that are met in a software product or project. The outcome of the software testing phase is generally errors that will be in the software. Testing conforms that the system or software works fulfilling the required needs. Software testing is a process in which the runtime quality and quantity of a software is tested to maximum limits. The process of testing any software system is an enormous task which is time consuming and costly [1,2]. The three stages of software testing are:

a. Generate test cases
b. Execute test cases
c. Evaluate results

The process of testing any software system is a time consuming process and pays much more on cost factor. Lot of time and effort is required in this process. Software testing when gets combined with the artificial intelligence helps in making the work more efficient and error free. Combination of Evolutionary techniques and software testing provide different advantages.

II. Genetic Algorithm
The genetic algorithm is one of the evolutionary approach to computing, which has the ability to determine find best solutions to optimization problems [3]. GAs have been widely studied, experimented and applied in many fields in the engineering worlds. Genetic algorithms are based on the evolution via selection, crossover, mutation and recombination. GAs are useful and work efficiently when the search space is large, complex and poorly understood, when domain knowledge is scarce or expert knowledge is difficult to encode.

*Corresponding author: Ikvinderpal Singh, Assistant Professor, Trai Shatabdi GGS Khalsa College, Amritsar, Punjab, India. E-mail: ips_sikand@yahoo.com; Tel: 8968800616
Received July 12, 2017; Accepted August 24, 2017; Published September 20, 2017

Copyright: © 2017 Ikvinderpal Singh. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
### III. Ant Colony Optimization

Ant colony optimization technique is a set of instructions based on search algorithms of Evolutionary Techniques for optimal solutions [4,5]. ACO inspired from the behavior of live ants, are capable of synchronization with searching solutions for local problem by maintaining array list to maintaining previous information gathered by each ant. This tour of the ants gives the solution for best shortest path which can then be analyzed for optimality.

### IV. Proposed method

The working of proposed technique is based on concepts of ACO and GA given below:

- Select the set of test case from the available test suite that will cover all the faults detected earlier in minimum execution time. Here ants are work as agents who explore the minimum set of test cases. The n/2 of ants will initially start foraging with randomly selected test cases. Now the ants will add new test cases on her explored path if adding of a test case increases its fault detection capacity. The crossover operation is used to exchange the information. The new set of test case produced after crossover is used by new ants to forage. The process is repeated till any of the ants has discovered a set of test cases that covers all faults detection [6, 7, 8].

ACO is proposed to minimize the cost of testing by test cases reduction and the output is given to the GA. Proposed algorithm as follow:

**Algorithm**

1. Initially “n/2” ants are selected for foraging.
2. Repeat the step 1 until termination condition satisfied.

**Start -> Input Cases to ACO -> Best Path find using ACO -> New Test Cases explored by GA -> Maximum Fault Detection -> Stop**

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

The ACO will give the best shortest path with help of distance value; the output is given to the GA algorithm. The set of path are obtained during the ACO process are input to GA and genetic algorithm undergoes the selection, crossover and mutation process and it give the result. The result contain only single path which is optimal among the shortest path [9,10].

GA is suited to the situation where solution space is huge and time taken to search is very high. Major advantage of GA is that it has ability to solve problems with no previous knowledge. For this reason we combined ACO and GA to find shortest path for generation of test cases.

### V. Conclusion

The conclusion shows how software testing using GA and ACO becomes efficient even with increasing number of test cases. Researchers may use this technique to reduce time and effort required for selection of test cases from a large test suite. The approach provides better results in the initial iteration of the whole process. It provides positive feedback and hence it can lead to best shortest path in optimum time. The proposed approach will result in minimizing execution cost, minimizing test design efforts/cost and maximum coverage ability of codes.
References


