A Critical Review on Test Case Prioritization and Optimization using Soft Computing Techniques

Priyanka Bansal
Mtech Scholar CSE KIET, Ghaziabad
pbansal64@yahoo.com

Abstract: Test case prioritization involves scheduling test cases in an order that increases the effectiveness in achieving some performance goals. One of the most important performance goals is the rate of fault detection. Test cases should run in an order that increases the possibility of fault detection and also that detects the most severe faults at the earliest in its testing life cycle. Regression Testing is an usual and a very costly activity to be performed, often in a time and resource constrained environment. Thus we use techniques like Test Case Selection and Prioritization, to select and prioritize a subset from the complete test suite, fulfilling some chosen criteria. Present paper gives the approach into existing single objective test cases prioritization and optimization using techniques such as Genetic Algorithms, Ant Colony Optimization.

Key Word: Test Case Prioritization, Regression Testing, Genetic Algorithm (GA), Prioritization Factors, Ant Colony Optimization

1 INTRODUCTION:

Software testing is one of the major and primary techniques for achieving high quality software. Software testing is done to detect presence of faults, which cause software failure. However, software testing is a time consuming and expensive task [1], [2]. It consumes almost 50% of the software system development resources [2], [3], [4]. Testing can be done either manually or automatically by use of testing tools. It is found that automated software testing is better than manual testing. However, very few test data generation tools are commercially available today [5].

Test case prioritization involves scheduling test cases in an order that increases the effectiveness in achieving some performance goals. One of the most important performance goals is the rate of fault detection. Test cases should run in an order that increases the possibility of fault detection and also that detects the most severe faults at the earliest in its testing life cycle. The nature and location of actual faults are generally not known in advance, test case prioritization techniques have to rely on available surrogates for prioritization criteria [6].

Over the lifetime of a large software product, the number of test cases could drastically increase as new versions of software are released. Because the cost of repeatedly retesting all test cases may be too high, software testers tend to remove redundant or trivial test cases to construct a reduced test suite for regression testing at a reasonable cost [7]. After development and release, software undergo regress maintenance phase of ten to fifteen years. Modifications in software may be due to change in customer’s requirements or change in technology or platform. This leads to release of numerous versions or editions of the existing software. Also in case of the version or edition’s test only modified and affected parts are to be tested to impart confidence in the modified software which is the process of regression testing [8]. Test case prioritization is an aspect of regression testing.

GA has been applied in many optimization problems for generating test plans for functionality testing, feasible test cases and in many other areas [9], [10]. GA has also been used in model based test case generation [3], [11]. Various techniques have been proposed for generating test data/test cases automatically using GA in structural testing [2], [5]. GA has also been applied in the regression testing, object oriented unit testing as well as in the black box testing for the automatic generation of the test cases [11], [9], [12].

1.1 Genetic Algorithm

In GA each chromosome consists of genes, a population P= (c1….cm) is formed from a set of chromosomes. A chromosome is a string of binary digits and each digit that makes up a chromosome is called a gene. The GA increases the population of chromosomes by continuously replacing one with another population and it based on fitness function assigned to each chromosome. The strong one go further and the weak chromosome eliminated generation by generation. Crossover and mutation these are two main concepts in genetic algorithm.

Initialize (population)
Evaluate (population)


While (stopping condition not satisfied) {
    Selection (population) 
    Crossover (population) 
    Mutate (population) 
    Evaluate (population) 
}

1.1 Selection
There is a selection pattern to find which are chosen for mating and it is based on fitness and capability of an individual to survive and reproduce in an environment. Selection generates the new one from the old one. Each chromosome is examined in present generation to determine its fitness value.

1.1.2 Crossover or Recombination
After selection, the crossover operation is applied to the selected chromosomes. It involves swapping of genes or sequence of bits in the string between two individuals.

1.1.2 Mutation
Mutation alters chromosomes in small ways to introduce new good traits. It is applied to bring diversity in the population.

1.2 Ant Colony Optimization
Ant colony optimization technique is a set of instructions based on search algorithms of artificial intelligence for optimal solutions; here the iconic member is ANT System, as proposed by Colorni, Dorigo and Maniezzo. Ants are blind and small in size and still are able to find the shortest route to their food source. They make the use of antennas and pheromone liquid to be in touch with each other. 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. Moreover, ACO deals with two important processes, namely: Pheromone deposition and trail pheromone evaporation. Pheromone deposition is the phenomenon of ants adding the pheromone on all paths they follow.

Pheromone trail evaporation means decreasing the amount of pheromone deposited on every path with respect to time. Updating the trail is performed when ants either complete their search or get the shortest path to reach the food source. Each combinatorial problem defines its own updating criteria depending on its own local search and global search respectively.

Artificial ants leave a virtual trail accumulated on the path segment they follow. The path for each ant is selected on the basis of the amount of “pheromone trail” present on the possible paths starting from the current node of the ant. In case of equal or no pheromone on adjacent paths, ants randomly choose the path. Pheromone trail on a path increases the probability of the path being followed. Ant then reaches the next node and again does the path selection process as described above. This process continues till the ant reaches the starting node. This finished tour gives the solution for shortest or best path which can then be analyzed for optimality.

1.2.1 Test Case Selection Using ACO
The proposed test case prioritization technique using Ant Colony Optimization within a time restricted framework [3] is implemented and evaluated. The technique uses the fault detection and execution time information of the regression test suite as an input. In the proposed algorithm, execution time acts as cost of executing the test case. Prioritization is done in order to achieve total fault detection and minimum cost of execution. We abbreviate the technique as ACO_TCSP. The basic block diagram for the ACO_TCSP (Ant Colony Optimization for Test Case Selection & Prioritization) system is shown in Fig.1. The inputs to the system include details of the test suite i.e., the test cases along with the faults covered by them and their execution time. These inputs are generally tabulated and are to be entered by the tester. The User of the ACO_TCSP tool needs only to enter the time constraint details at the run time. The output then produced has path details for each iteration, pheromone details, best path details and the final selected & prioritized test suite.

2 A Survey of Recent Research in the Field

2.1 Sangeeta Sabharwal, Ritu Sibal and Chayanika Sharma[13]
In this paper a GA based approach is proposed for identifying the test path that must be tested first. Test paths or scenarios are derived from activity diagram and state chart diagram respectively. The proposed approach makes use of IF model and GA to find the path to be tested first.

2.2 S. Raju, G. V. Uma[14]
In this paper the regression testing based test suite prioritization technique is illustrated. A new prioritization technique is proposed for requirement
based System level test cases to improve the rate of fault detection of severe faults.

2.3 R.Krishnamoorthi and S.A.Sahaaya Arul Mary2[15]

In this paper author propose a new test case prioritization technique using Genetic Algorithm (GA). The proposed technique prioritizes subsequences of the original test suite so that the new suite, which is run within a time-constrained execution environment, will have a superior rate of fault detection when compared to rates of randomly prioritized test suites. This experiment analyzes the genetic algorithm with regard to effectiveness and time overhead by utilizing structurally-based criterion to prioritize test cases. An Average Percentage of Faults Detected (APFD) metric is used to determine the effectiveness of the new test case orderings.

2.4 Chartchai Doungsa-ard, Keshav Dahal, Alamgir Hossain, and Taratip Suwannasart[16]

In this paper author proposed an approach for generating test data from UML state diagram using genetic algorithm. This approach helps software developers to reduce their effort in generating test data before coding.

2.5 Arvinder Kaur, Shubhra Goyal[17]

In this paper a new Genetic Algorithm to prioritize the regression test suite is introduced that prioritize test cases on the basis of complete code coverage. The genetic algorithm would also automate the process of test case prioritization. The results representing the effectiveness of algorithms are presented with the help of an Average Percentage of Code Covered (APCC) metric.

2.6 Bharti Suri, Shweta Singhal[18]

This paper presents an implementation of an already introduced Ant Colony Optimization Algorithm for Test Case Selection and Prioritization. Graph representation and example runs explained in the paper show how the random nature of ACO helps to explore the possible paths and choose the optimal from them. Results show that ACO leads to solutions that are in close proximity with optimal solutions. In this study a tool ACO_TCSP for the same has been developed and applied on an example. Though in these tests the best solution was not found for all cases still the results obtained are in close proximity to the optimal results. The reduction in test suite size is achieved to be 62.5% in all the 4 test runs.

3 Conclusion & Future Work

Test data generation is one of the key issues in software testing. A properly generated test suite may not only locate the errors in a software system, but also help in reducing the high cost, efforts associated with software testing. Present work surveyed various techniques of software test case optimization. First we summarized traditional and advanced test optimization techniques, and then we identified gaps in existing techniques. Optimization of test cases is multi-objective optimization, NP complete and peculiar nature problem. Soft computing can be used for these type problems,
whose inexact solutions driving is computationally hard tasks such as the solution of “NP-complete problems”. In conclusion, a lot of test cases optimization techniques have been developed for achieving software testing effectiveness and fault coverage. Review of existing literatures has identified that there are several objectives of test case optimization like maximum number of defect detecting capability, minimum test design efforts/cost, minimum execution cost, maximum coverageability of client requirements & codes, maximum mutant killing score and so forth. Therefore optimization of test cases should be treated as multi-objective optimization problem. However most of test cases optimization approaches are single objective. Single objective formulation of test cases optimization problem is not justified and not meeting the objectives of testing. Some objectives are conflicting in nature, coverageability of one objective will suffer other objective while considering all objectives concurrently. So, there is strong need to shift the paradigm from single objective test case optimization to multi-objective test case optimization. Moreover for these techniques, soft computing approaches like Genetic Algorithms, Fuzzy Logic, Artificial Neural Network etc may be well suited for experimentation and validation purpose.

4 References:


