CHAPTER 6

SYNTHESIZED OVERVIEW OF TEST CASE OPTIMIZATION TECHNIQUES

6.1 INTRODUCTION

Software testing is an expensive but a required phase of software development life cycle. By the recent studies it is observed that testing consumes more than 50% of the cost of software development and 30% to 40% of overall software development time on few instances. Thus, it is must to reduce the effort and time of software testing by reducing the test suites size. Nowadays, there has been a rapid growth of practices in reducing the number of test cases in test suites. Recently, a large number of software test suite reduction techniques have been implemented. Among the activities of testing, the test case reduction is one of the seriously challenging works and also an essential one, since it has notable influence on the efficiency and effectiveness and of testing process. There is no surprise that many researches in the past years have been dedicated their time on test case optimization. As a result, various techniques of test case reduction have been investigated thoroughly. Currently software systems became more puzzling, for example, with unit modules developed by different vendors, using various techniques in different programming languages and even executed in multiple platforms. Many reduction techniques for test suites are considered by the companies in software testing. Still a big gap exists between theoretical software application systems and real use of test case reduction techniques proposed by research. It is suggested for Software test suite reduction researchers, to essentially reassess the available techniques, recognize the open problems and looking forward for an open view on the future of test case reduction. In the
direction of such motive, this study offers an essential view on popular test suit reduction techniques and by taking a novel drive-reach. This contains collective work gathering and self-standing sector, each focusing on an important analysed topic, in this study a test suit reduction technique. The test suite created and minimized should satisfy the given requirements. Further, the test suite generated is an absolute set of all possible test cases. Some of the test cases in the original test suite may be redundant with respect to the requirement criteria. Those test suites will only be observed and removed when implementing reduction techniques. Thus the test suite optimization techniques we consider in this study includes Ant Colony Optimization, Artificial Bee colony optimization, Particle swarm optimization, Genetic algorithm. The work is an attempt to analyse the algorithms currently available for test suite reduction from a large number of test suites and compare their efficiencies based on test suite reduction and performance.

6.2 OVERVIEW OF TEST CASE OPTIMIZATION TECHNIQUES

Different algorithms based on test suite optimization such as genetic algorithm, artificial bee colony optimizations, Ant colony optimizations and particle swarm optimization have been mainly assessed for test suite reduction from a large test suite. Plenty of research work happened for optimizing test suites or test cases like Ant colony optimization, particle swarm optimization, Artificial Bee colony optimization, Genetic algorithm, Gravitational search algorithm and also few of hybrid approaches by combining the above mentioned algorithms. In the Artificial Bee Colony algorithm, the scavenging behaviour of honey bees has been adapted to the job scheduling mechanism. The bee colony algorithm has been used to create pair wise test sets. Another test suite optimization approach is based on the behaviour of biological bees. The ABC algorithm for automatic generation of structural software tests is suggested to the approximate solution of difficult optimization problems. A hybrid genetic
algorithm is focused on quality improvement and optimization of test cases.

6.2.1 ANT COLONY OPTIMIZATION (ACO)

Swarm intelligence is one of a popular and known technique for problem solving which observes the behaviour from natural biological systems which has been mainly assessed for test suite reduction and dominance from a large test suite. Ant colony optimization (ACO) is introduced a class of optimization algorithms modelled on the actions of an ant colony. ACO is a probabilistic method which is useful in problems that looks for better ways through graphs. Artificial ants are the simulation agents and locate optimal solutions by navigating through a parameter space which represents all possible solutions. Natural ant’s behaviour of guiding each other to where the resources are available for exploring their environment. The simulated ants similarly register their positions and their quality, so that more ants locate better solutions. (Dorigo. M et al, 2006). An alternate of the double bridge experiment, one bridge is longer than the other. In this case, the stochastic changes in the first choice of a bridge are much reduced and a second technique plays an important role. The ants selecting the short bridge by chance are the first to reach the destination. Therefore the ants through shorter bridge are achieving earlier than the long one and this increases the probability that other ants also select it instead of the long one. A new model is developed by the observed behaviour; at a given moment in time m₁, ants used the first bridge and m₂ the second one, then the probability p₁ for an ant to opt the first bridge as follows.

\[ p₁ = \frac{(m₁ + k)^h}{(m₁ + k)^h + (m₂ + k)^h} \]  

(6.1)

An ant is a simple calculation agent in the ant colony optimization. It iteratively
creates a solution for the problem. The in-between solutions are mentioned to as solution states. At the end of each iteration, each ant moves from a state \( x \) to state \( y \) conforming to a more complete in-between solution. The trail level denotes a posteriori symptoms of the attraction of that move. Trails are modernized usually when all ants have achieved their solution, adjusting the level such as increasing or decreasing the trail levels corresponding to moves which were part of good or bad solutions respectively. Figure 6.1 shows the double bridge experiment graphical representation.

![Figure 6.1 Experimental Setup for the double bridge Experiment with equal different lengths](image)

Here parameters \( k \) and \( h \) are to be fit with experimental data. \( p_2 = 1 - p_1 \). The simulations explained a very good fit for \( k = 20 \) and \( h = 2 \). The adaption of food search behaviour suggested was the base motivation for the development of ant colony optimization. (Deneubourg. J.L et al, 1990). In ACO, a number of artificial ants develop solutions to the optimization problem which is taken for assessments and transform information based on the quality of these results through a channel to communicate, which is suggested by real ants. Different ant colony optimization algorithms have been suggested. The original ant colony optimization algorithm is called as Ant System and was suggested in the early decades. Then, a number of other ACO algorithms were coming up which shares the same basic behaviours of ants.
6.2.2 ARTIFICIAL BEE COLONY OPTIMIZATION (ABC)

Artificial Bee Colony (ABC) algorithm is based on swarm and it is a meta-heuristic algorithm. It is basically inspired by the intellectual foraging behaviour of honey bees and formulates that scavenging act of honey bees. The eventual goal of the bees is to discover the location of the food source with high nectar amount (Karaboga. D, 2005). The colony of bees in ABC algorithm has three different types of bees: employed bees, onlookers and scout bees. Employed bees forage in search of new food source and back to hive to perform a dance. The employed bees that discover an abandoned food source becomes a scout and search for a new food source again. Onlookers conclude the food source depending upon the dance of employed bees. A nectar source is selected by each bee with the help of succeeding a nest mate whose food source has already found. The bees dance on the hive, to explore that they identified nectar sources and communicate their nest mates to follow them. Other bees follow the dancing bees to one of the nectar areas. On collecting the nectar they return back to their hive, handover the nectar to a food storing bee. After handing over the food, the bee chooses with one of the available choices,

(a) Leave the food source and be as a non-aligned follower,
(b) Without joining with the nest mates, continue to forage for the New food source or
(c) Volunteer by dancing before the nest mate’s return.

Many food areas are found and communicated by the dancing bees within the hive. The policy, by which the bee decides to follow a particular dancer, is not found yet but it is assumed as the staffing among bees is always a function based on the quality of the food source identified.
6.2.3 PARTICLE SWARM OPTIMIZATION (PSO)

Particle swarm optimization is a swarm intelligence based optimization algorithm that is used to resolve optimization problems. In particle swarm optimization, software agents are called as particles, they are in movement in the whole search space of an optimization problem. The position of a particle indicates a candidate solution to the optimization problem. Each particle looks for better positions in the search space by updating its velocity often according to rules originally observed by model behaviours models of bird flocking (Qinghai Bai, 2010). When searching food, the birds are either spread or go together until they locate the place where they can get food. When birds are searching for food from one location to another, there is always a bird that can identify the food well. That is, the bird well knows of the places where food can be available or in other words having the better food source information. Because they are exchanging the information, the birds will then assemble to the place where they can find food. As far as this PSO algorithm is concerned, resolution swarm is compared with the bird swarm, the birds moving from one location to another is equal to the progress of the solution swarm, good information is equal to the good optimum solution, and the food source is equal to the most optimist solution. Most of the optimist solution can be worked out in PSO algorithm by the coordination between each other individual. This algorithm is used to resolve the complex optimization problems. Since this approach has many advantages such as simplicity and painless implementation, this can be used extensively in the fields such as model classification, function optimization, the signal processing, machine study, neutral network training, vague system control, automatic adaptation control. Nowadays PSO is widely used in all the fields for achieving optimized solutions.
6.2.4 GENETIC ALGORITHM (GA)

Genetic algorithm (GA) is an optimization technique which solicits solutions to different real time problems. GA is a method for solving both conditioned and unconstrained optimization issues based on natural biological evolutionary selection process. GA repetitively updates a population of individual solutions. In all steps, the genetic algorithm chooses individuals randomly from the existing population to be parents and with the help of them produce the children of next generation. Over consecutive generations, the population advances in the direction of an optimal solution. NP-hard problems also can be resolved by applying GA. GA’s search methods were introduced by John Holland and comprehensively studied by Goldberg, De Jong and many other researchers. The survival of fittest technique is where only the best solutions survive and are updated until we get a good result (Bharti Suri et al, 2009). The GA process has various steps as shown in Figure 6.2.

Figure 6.2 GA Architecture
GA gives the best solution in a particular subset of solutions. Genetic algorithm requires a typical genetic characterization and a fitness function to validate the solution domain. Few researchers states that the solutions to the problems have been encoded. The initial population is chosen using fitness-based function and tournament selection like roulette wheel. The next generation of solution population is produced from first generation of solution population using crossover and mutation behaviours. Thus the new population is chosen and further it takes part in producing the next generation solution population. Figure 6.3 explains the functional workflow of Genetic Algorithm.

Figure 6.3 GA functional workflow
This process is continual until a pause or stop condition is attained, that is the result has been found or number of generations reached to the upper limit. This method is based on merging two individual’s information and generates two new children of information units. The new population is created by cutting two strings at the user crossover point and swapping them. The result of this process is the new generation solution population.

Take two strings 10111 00101, 11111 00001 and perform a 2-point crossover on them, the new population strings created after applying crossover are, 1011100001 and 1111100101. The available common techniques used in Multi-objective GA to attain the goals of multi-objective optimization are fitness functions, diversity, elitism, constrain handling, parallel and hybrid multi objective genetic algorithm. The Genetic algorithm has three different types of rules at each step to create the next generation from the current population. They are selection, crossover and mutation. Selection process chooses the individuals called parents that contribute for the population of next generation. Crossover combines two parents to generate children for the subsequent generation.

6.3 CONCLUSION

A benchmark study was conducted on few of the most popular optimization techniques. Software testing is one of the cost consuming activity but mandatory for quality assurance in software development lifecycle. By reducing the number of test cases or test suites the software testing cost can be considerably reduced. This research work analyses the test case optimization techniques in the field including probabilistic, meta-heuristic, Multi-objective optimization. All of the algorithms studied are direct methods and have some common Characteristics, but other aspects of these methods are significantly different. This study provides a synthesized overview of test case optimization techniques.