A Synthesized Overview of Test Case Optimization Techniques

1SRIVIDHYA. J, 2DR. K. ALAGARSAMY
1 Research Scholar, Department of Computer Science, Karpagam University.
jsrividhya.ku2011@gmail.com
2 Department of Computer Science, Madurai Kamaraj University.

ABSTRACT
The key component to assess the software performance is how well it performs and it becomes an important activity in Software Engineering field these days. Testing is broadly used in industry for quality assurance. Indeed, with the complexity, variety of software growing continuously, confirming that it behaves according to the expected levels of quality and reliability becomes more crucial, and more difficult and expensive. One of the essential and critical tasks in software engineering process is downsizing the effort of software testing and tries to make it as pragmatic as possible and reduce the cost & time of development as well. This paper presents a synthesized overview of the most popular techniques for optimization of software test cases. The techniques presented encompasses: (a) ACO - (Ant Colony Optimization), (b) Artificial Bee Colony Optimization (ABC), (c) PSO (Particle Swarm Optimization), (d) Genetic Algorithm. Each technique is furnished by worldwide researchers on the technique, and precisely covers the basic concepts of the technique, the current update, considerations of the open research problems, and an outlook of the future development in the approach. As a whole, the paper aims at giving a preparatory, state of art brief overview of research in test case optimization techniques, while guarantying extensiveness and assertiveness.

Keywords: software testing , Test case reduction techniques, Particle Swarm Optimization, Genetic Algorithm , Artificial Bee colony optimization, Synthesized Overview, Ant Colony Optimization, Test Suite Optimization, Software testing cost reduction.

1. INTRODUCTION
Software testing is essential for all software development. It is an intrinsic part of software engineering. However, testing is an expensive part. It is often accounted for more than 52% of total development costs. Thus, it is crucial to reduce the cost and increase the productiveness of software testing by reducing the test suites. Nowadays, there has been a speedy growth of practices in reducing the number of test suites. Currently, a large number of software test reduction techniques have been developed. Among many testing activities, the test case reduction is one of the most studiously challenging tasks and also of the most essential ones, since it can have a notable influence on the effectiveness and efficiency of whole testing process. There is no wonder that many research efforts in the past years have been spent on test case optimization. As a result, different techniques of test case reduction have been interrogated intensively. Currently software systems have become more and more puzzling, for example, with unit modules developed by different vendors, using various techniques in multiple programming languages and even running on different platforms. Although reduction techniques for test suites are taken up by the companies in software testing practice, there
still exists a big hiatus between practical software application systems and real usability of test case reduction techniques proposed by research. For Software test suite reduction researchers, it is recommended to essentially reassess the already available techniques, understand the open problems and looking forward a broad view on the future of test case reduction.

In the direction of such intention, this paper offers a pivotal overview on a number of popular test suite reduction techniques and by taking an innovative drive-reach that we call a synthesized overview. This contains collusive work gathering self-standing sections, each focusing on a decisive analyzed topic, in our case a test suite reduction technique. The test suite thus generated and reduced should satisfy the testing requirement criteria as defined by the tester. Further, the test suite generated in literature is a complete set of all possible test cases. Some of the test cases in the generated test suite may be redundant with respect to the testing criteria. Those test suits will only be analyzed and removed when applying reduction techniques. Figure-1 explains the basic idea of attaining a minimized (optimized) test suite which would help to reduce the number of test cases in testing and thus result in the reduced time and cost in testing efforts. Thus the test suite optimization techniques we consider in this paper includes

1) ACO - (Ant Colony Optimization)
2) ABC (Artificial Bee Colony Optimization)
3) PSO (Particle Swarm Optimization)
4) Genetic Algorithm

The work is an attempt to analyze the algorithms currently available for reduction of test suites from a large number of test suites and compare their test suite reduction and performance efficiencies.

Figure-1: Basic steps to attain Optimized Test Suites

The paper is organized as follows. Section II befits to related works and research in this specific area. Section III explains about the Ant Colony Optimization. Section IV explains the artificial bee colony optimization. Section V examines the Particle Swarm Optimization. Section VI discusses how the Genetic algorithm works. Section VII summarizes as the Conclusion.

2. Related Work
Various algorithms based on genetic algorithm [1,2] and Artificial bee colony optimizations and Ant colony optimizations [3, 4] and particle swarm optimization [37] have been mainly analyzed for test suite reduction and predominance from a large test suite. A lot of research work happened for optimizing test suites or test cases. Karaboga [6, 7, 8] introduced the theory of Artificial Bee Colony algorithm. The honey bees scavenging
practice has been mock-up to the job scheduling mechanism by Chong et al [9]. A Pretended bee colony algorithm has been used to generate pair wise test sets by McCaffrey et al [10]. A new pheromone based test suite optimization approach has been offered by Jayamala et al [11], which is based by the behavior of biological bees. Dahiya et al [12] presented an ABC algorithm based approach for automatic generation of structural software tests. Marco Dorigo et al [13] suggested a promising approach to the approximate solution of difficult optimization problems by Ant colony optimization technique. Mala et al has introduced a hybrid genetic algorithm based approach for quality improvement and optimization of test cases [14]. The fruit of fault detection of test set when it minimized has been examined by Eric et al [15]. Sthamer[16] and Pargas et al [17] applied GA for automatic test data generation in his proposal. Jones et al proposed a strategy by using GA to automate branch and fault based testing [18] and also presented a view on automatic structural testing using GA [19]. Using Search based technique, Harman et al suggested a technique to reduce the input domain [20]. Anastasis and Andreas introduced an extensive approach to create dynamic test data [21]. Bharti Suri et al, proposed an hybrid technique based on BCO and GA [22].

3. ANT COLONY OPTIMIZATION (ACO)
Swarm intelligence is a known approach to problem solving which extracts inspiration from nature biological systems. Ant colony optimization (ACO), introduced by Dorigo in his doctoral dissertation, is a class of optimization algorithms modeled on the actions of an ant colony. ACO is a probabilistic technique useful in problems that deal with finding better paths through graphs. Artificial 'ants'—simulation agents—locate optimal solutions by moving through a parameter space representing all possible solutions. Natural ants lay down pheromones directing each other to resources while exploring their environment. The simulated 'ants' similarly record their positions and the quality of their solutions, so that in later simulation iterations more ants locate better solutions. [3, 13].

Deneubourg et al. [24] comprehensively examined the pheromone laying and action manners of ants. In an observation known as the “double bridge experiment”, the nest of a colony of ants was connected to a food source by two bridges of equal lengths [see Figure 2(a)]. In such an arrangement, ants start to travel around the environs of the nest and finally reach the food source. Along their path between food source and nest ants deposit pheromone. Initially, each ant incidentally chooses any one of the bridge among the two. Nevertheless, due to casual alternation, after some time one of the two bridges has higher focus of pheromone than the other one and, therefore, attracts more ants. This brings a further amount of pheromone on that bridge making it more attractive with the result that after some time the whole colony traverse toward the use of the same bridge. This colony level behavior, based on autocatalysis, that is, on the exploitation of positive feedback, can be used by ants to find the shortest path between a food source and their nest [13].

Goss et al. [25] considered an alternative of the double bridge experiment in which one bridge is considerably longer than the other [see Figure 2(b)]. In this case, the stochastic fluctuations in the initial choice of a bridge are much reduced and a second mechanism plays an important role: the ants choosing by chance the short bridge are the first to reach the nest. The short bridge receives, therefore, pheromone earlier than the long one and this fact increases the probability that further ants select it rather than the long one. Goss et al. [25] developed a model of the observed behavior: assuming that at a given moment in
time m1 ants have used the first bridge and m2
the second one, the probability p1 for an ant to
choose the first bridge is:

\[
p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h}
\]

where parameters k and h are to be fitted to
the experimental data. Obviously, p2 = 1 - p1.
Monte Carlo simulations showed a very good
fit for k = 20 and h = 2 [26].

The mock-up suggested by Deneubourg and
co-workers [24] for describing the food-search
behavior of ants was the base origin of
motivation for the development of ant colony
optimization. In ACO, a number of artificial
ants build solutions to the optimization
problem which is taken for examination and
exchange information on the quality of these
solutions through a communication channel
that is evocative of the one adopted by real
ants.

![Figure-2: Experimental Setup for the
double bridge Experiment with equal
different lengths.](image)

Different ant colony optimization algorithms
have been proposed. The original ant colony
optimization algorithm is known as Ant
System [27, 28, and 29] and was proposed in
the early decades. Since then, a number of
other ACO algorithms were introduced which
shares the same basic idea.

4. ARTIFICIAL BEE COLONY
OPTIMIZATION (ABC)

Artificial Bee Colony (ABC) algorithm is a
swarm based meta-heuristic algorithm
introduced by Karabogain 2005 [3, 30],
inspired by the intelligent foraging behavior of
honey bees and fabricate that scavenging
action of honey bees. The ultimate goal of the
bees’ is to identify the location of the food
source positions with high nectar amount [31].

The colony of bees in ABC algorithm consists
of three groups of bees: employed bees,
onlookers and scouts [32]. Bharti et al [22],
Employed bees forage in search of their food
source and return to hive and perform a dance
on this area. The employed bee who find
abandoned food source becomes a scout and
find a new food source again. Onlookers
decided their food source depending upon the
dances of employed bees. A nectar source is
chosen by each bee by succeeding a nest mate
whose food source has already discovered.
The bees dance on the hive, to inform that
they discovered of nectar sources and
persuade their nest mates to follow them. To
get nectar, other bees follow the dancing bees
to one of the nectar areas. On collecting the
nectar they come back to their hive, handover
the nectar to a food storer bee.

After renounce the food, the bee opts for one
of the choices with a certain probability (a)
Leave the food source and act as a nonaligned
follower, (b) Without joining the nest mates,
continue to forage at the food source or (c)
volunteer the nest mates by dancing before the
return to the food source. Various food areas
are identified and announced by bee dancers
within the dance area. The policy, by which
the bee decides to follow a specific dancer, is
not revealed yet but it is taken as “the
recruitment among bees is always a function
of the quality of the food source”. To be noted
that not all bees start foraging simultaneously.
By [30], the general scheme of the ABC
algorithm is as follows:

Initialization Phase

REPEAT

Employed Bees Phase
Onlooker Bees Phase

Scout Bees Phase

Memorize the best solution achieved so far

UNTIL (Cycle=Maximum Cycle Number or a Maximum CPU time).

5. PARTICLE SWARM OPTIMIZATION (PSO)
Particle swarm optimization is an optimization algorithm based on swarm intelligence that are used to solve optimization problems. In particle swarm optimization, simple software agents, called particles, move in the search space of an optimization problem. The position of a particle represents a candidate solution to the optimization problem at hand. Each particle searches for better positions in the search space by changing its velocity according to rules originally inspired by behavioral models of bird flocking.[36]

By Qinghai Bai [37], Particle swarm optimization was introduced by Kennedy and Eberhart (1995). While searching for food[37], the birds are either scattered or go together before they locate the place where they can find the food. While the birds are searching for food from one place to another, there is always a bird that can smell the food very well, that is, the bird is perceptible of the place where the food can be found, having the better food resource information. Because they are transmitting the information, especially the good information at any time while searching the food from one place to another, conducted by the good information, the birds will eventually flock to the place where food can be found. As far as particle swarm optimization algorithm is concerned, solution swarm is compared to the bird swarm, the birds’ moving from one place to another is equal to the development of the solution swarm, good information is equal to the most optimist solution, and the food resource is equal to the most optimist solution during the whole course. The most optimist solution can be worked out in particle swarm optimization algorithm by the cooperation of each individual. The particle without quality and volume serves as each individual, and the simple behavioral pattern is regulated for each particle to show the complexity of the whole particle swarm. This algorithm can be used to work out the complex optimist problems.

Due to its many advantages including its simplicity and easy implementation, the algorithm can be used widely in the fields such as function optimization, the model classification, machine study, neutral network training, the signal procession, vague system control, automatic adaptation control and etc(Zheng Jianchao,Jie Jing,Cui Zhihua,2004,(In Chinese)) [37].

5. GENETIC ALGORITHM (GA)
A genetic algorithm (GA) is an optimization technique [22], solicited to different real time problems. By [34] GA is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. GA repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. GA can also be applied in NP-hard [33] problems. GA’s search procedures were introduced by John Holland and extensively studied by Goldberg, De Jong and many other researchers. It uses a “survival of the fittest” technique, where the best solutions survive and are varied until we get a good product [22]. GA provides the best
solution in a specific subset of solutions. GA could also be applied on the NP-hard [33]

A typical genetic algorithm requires: [34] a genetic representation of the solution domain, a fitness function to evaluate the solution domain.

The GA process consists of various steps as shown in figure 3.

**Figure-3: GA Architecture**

Bharti Suri et al [22], state that Encoding is done for the solution to the problem. Using fitness-based function like roulette wheel selection and tournament selection, initial population is chosen. The second generation population of solutions is generated from first generation using genetic operators like crossover and mutation. New population will be chosen and further take part in generating the next generation. The process is repeated until a termination condition is reached (i.e. the result has been found or, fixed number of generations reached). This is the method of merging the information units of two individuals that will produce two more new children (information units). Here cutting of the two strings at the user crossover point and swapping the two. The outcome of this process is the new population. Take two strings and perform a 2-point crossover on them.

1111 1 00001

The new population or strings generated after applying crossover are: 1011100001 and 1111100101.

The Genetic algorithm uses three main types of rules at each step to create the next generation from the current population as below:

**Selection:** Selection rules select the individuals, called parents that contribute to the population at the next generation.

**Crossover:** Crossover rules combine two parents to form children for the next generation.

**Mutation:** Mutation rules apply random changes to individual parents to form children.

**Basic Algorithm**

The basic algorithm GA as follows [36]:

0 START: Create random population of \( n \) chromosomes

1 FITNESS: Evaluate fitness \( f(x) \) of each chromosome in the population

2 NEW POPULATION

1 REPRODUCTION/SELECTION: Based on \( f(x) \)

2 CROSS OVER: Cross-over chromosomes

3 MUTATION: Mutate chromosomes

3 REPLACE: Replace old with new population: the new generation

4 TEST: Test problem criterium

5 LOOP: Continue step 1 – 4 until criterium is satisfied

Figure 4 explains the functional workflow of Genetic Algorithm and Table-1 provides the
comparison of GA and traditional optimization techniques.

**Multi-objective Genetic Algorithm:**
By Abdullah Konak et al [38], GA is a population based approach and is well suited for solving multi-objective optimization problems. In a single run a generic single-objective GA can be easily modified to find a set of multiple non-dominated solutions. The capability of GA to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems with non-convex, discontinuous, and multi-model solutions spaces. Many engineering problems have multi-objectives, including engineering system design and reliability optimization.

The available common techniques used in Multi-objective GA to attain the goals of multi-objective optimization are as below:

1. Fitness functions
2. Diversity
3. Elitism
4. Constrain Handling
5. Parallel and Hybrid Multi-Objective GA

**Table 1:** Comparison of Genetic Algorithm with Traditional Optimization Techniques

<table>
<thead>
<tr>
<th>NO</th>
<th><strong>Genetic Algorithm</strong></th>
<th><strong>Traditional Optimization Technique</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Works with coding of solution set</td>
<td>Works directly with the solution</td>
</tr>
<tr>
<td>2</td>
<td>Searching on population of solutions</td>
<td>Searching on single solution</td>
</tr>
<tr>
<td>3</td>
<td>Evaluation based on fitness function</td>
<td>Evaluation based on the derivatives</td>
</tr>
</tbody>
</table>

**Figure-4:** GA functional workflow
6. 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 paper analyzes 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.

7. REFERENCES


[36] http://www.scholarpedia.org/article/Particle_swarm_optimization

www.jrret.com