Automated Software Testing Using Metaheuristic Technique Based on Improved Ant Algorithms for Software Testing

Pooja Sharma
Dept. of CSE
CBS Group of Institutions
Fatehpuri (Jhajjar), India
E-mail: sharma.pooja218@gmail.com

Abstract- Testing can never completely identify all the defects within software [1]. Instead, it furnishes a criticism or comparison that compares the state and behavior of the product against oracles principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts[2], comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria. Testing effectiveness can be achieved by the State Transition Testing (STT) which is commonly used in real time, embedded and web based type of software systems. Aim of the current paper is to present an algorithm by applying an ant colony optimization technique, for generation of optimal and minimal test sequences for behavior specification of software. Present paper approach generates test sequence in order to obtain the complete software coverage. This paper also discusses the comparison between two Meta heuristic techniques (Genetic Algorithm and Ant Colony optimization) for transition based testing.

Keywords- Software Testing, Ant Colony Optimization (ACO), Genetic Algorithm (GA), Improved Ant Algorithms

INTRODUCTION
Software products should be reliable, correct, and scalable. To ensure these qualities, it is necessary to test the software at various conditions and hence software testing is an important component of the software development process [3, 4, 5]. A primary purpose of testing is to detect software errors/problems so that the defects may be discovered and corrected [3] before delivery the software. Due to the time and cost constraints, it is not possible to test the software manually and fix the defects [4]. Thus the use of test automation plays a very important role in the software testing process [5]. Now a day, Artificial Intelligence (AI) techniques are changing the nature of the test automation process [6]. It has been identified that one of the software engineering areas with a more suitable and realistic use of artificial intelligence techniques is software testing [6, 7, 8, 9] and these techniques are known as a metaheuristic approach [10] This paper purposes an algorithm which uses an ACO optimization technique to generate the automatic state transition test sequence, which gives a strong level of software coverage.

An ACO algorithm [11] [12] is a probabilistic technique for solving computational problems which can be used to find "good" paths through the graph. It is depends on the behavior of ants in finding paths from their colony to food [12] [13]. Using ACO to generate test sequences for state-based software testing is already presented in [14] but the main problem is complete software coverage. As the present market is highly competitive, it is a pressing need of software organizations to provide good quality software products to the customer within the estimated budget, and hence a strong level of testing coverage technique is essential. The ACO as said earlier is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. State based testing using an ACO has been nicely represented [14], this approach is well suited for state based testing, not of all transition based coverage, still this approach is quite better for doing research using ACO in the area of software testing.

I. GENETIC ALGORITHM AND SOFTWARE TESTING
It has been identified that one of the Software Engineering areas with a more prolific use of AI techniques is software testing. The focus of these techniques includes the applications of genetic algorithms (GA). GA provides a general method for a searching technique, which uses the concept of natural evolution [15]. It is inspired by natural genetics to provide the solutions to problems. Genetic algorithms are inspired by Darwin's theory of evolution [16]. GA is very important for various reasons and it has been applied to a large number of scientific and engineering problems, such as automatic test data generation, optimization, machine learning, automatic programming, transportation problems, adaptive control, etc. [17][18]. GA evolves as a number of suitable test data sets, each test data for at least each path. The best test data for each parallel path are picked and used for calculating the overall coverage test data [19]. Another work by [20] [21] enhances the coverage of transitions but does not guarantee for full coverage of the test data. Therefore, we can conclude[20][19], that the GA approach is not feasible for the type of control, embedded and web based kind of software systems because the GA based proposed approach does not provide a full coverage of all the transitions. If the software is not fully
covered, there may be uncovered transitions which have defects, which in turn can cause problems in the system. Graphical representation of the above analysis [19] [20] is shown in Figure 1.

![Transition coverage graph for telephone system](image)

**Figure 1.** Transition coverage graph for telephone system

It is clear that around 43% of the transitions are uncovered. These uncovered transitions may reveal a lot of uncovered defects in the large state transition systems. Therefore, this approach using GA algorithm is not suitable for real time and control systems.

In this type of requirement, an Ant’s behavior is very useful. The next section discusses about how ACO can generate full coverage of transitions in a state transition based software testing.

II. AN ANT COLONY OPTIMIAZTION

First, I suggest the approach to generate the automatic test cases and provide a solution to cover all the transition at least once. The purpose of the ACO algorithm is to cover the optimal transition at least once in the UML [22] State Transition Diagram of the software under test. It provides the optimal test sequence of transition in the state transition diagram. Selection of transitions depends upon the probability of the transition. Higher probability values means that the chances to select the transition are also high. The probability value of transition depends upon: the feasibility of transition ($F_{ij}$), which shows direct connection between the vertices; pheromone value ($\tau_{ij}$), which helps other ants make decisions in the future, and heuristic information ($\eta_{ij}$) of the transition, which indicates the visibility of a transition for an ant at the current vertex. In some cases if there are equal probabilities of feasible transition, then by the following three steps the algorithm selects the feasible transition [13].

a) An ant will select a self-transition if it exists at a current vertex or else the ant will approach to rule 2. An ant will select the next state according to the value of visited status parameter ($V_s$). If the current vertex $V_i$ is direct connected to the vertex say $V_j$ and is not visited yet by the ant, then the ant will select $V_j$ as the next state which means that the transition ($V_i$ -> $V_j$) traversed. This concept fulfills the criteria of all state coverage at least once.

b) After all the above consideration, if the selection is not possible then the ant will select any feasible transition randomly.

In this algorithm the ant has the ability to collect the knowledge of all feasible transitions from its current state. An approach for the feasibility check of the transitions from the current state is used. This approach is defined in the feasibility set of transition ($F_s$). The ant also has four other information about transition viz. Pheromone level on transition ($\tau_{ij}$), Heuristic information for the transitions ($\eta_{ij}$), visited states with the help of visited status ($V_s$) and the probability parameter $P$. After the selection of a particular transition the ant will update the pheromone level as well as the heuristic value. Pheromone level is increased according to last pheromone level and heuristic information but heuristic information, is updated only on the basis of the previous heuristic information. An ant $p$ at a vertex ‘i’ (here vertex means state of state transition diagram) and another vertex ‘j’ which is directly connected to “i” means that there is a transition between the vertices ‘i’ and ‘j’ i.e. ($i$,$j$). In the graph this transition is associated with five tuple $F_s$($p$), $\tau_{ij}$($p$), $\eta_{ij}$($p$), $V_s$($p$) and $P$($p$) where ($p$) shows that values of tuple associated with ant $p$. All description about these attribute is given below:

a) Feasible transition set: $F = \{F_s(p)\}$ represents the direct connection with the current vertex ‘i’ to the neighboring vertices “j”. Direct connection shows that the vertices are adjacent to the current vertex ‘i’, i.e. a direct edge exists in between the current vertexes ‘i’ and the chosen vertex ‘j’.
   - $F_1=1$ means that transition between the vertex ‘i’ and ‘j’ is feasible.
   - $F_1=0$ means the transition between the vertex ‘i’ and ‘j’ is not feasible.

b) Pheromone trace set: $\tau = \tau_{ij}$($p$) represents the pheromone level on the feasible transition ($i$,$j$) from current vertex ‘i’ to next vertex ‘j’. The pheromone level is updated after the particular transition. This pheromone helps other ants to make decisions in the future.

c) Heuristic set: $\eta = \eta_{ij}$($p$) indicates the visibility of a transition, for an ant at current vertex ‘i’, to vertex ‘j’.

d) Visited status set: $V_s$ shows information about all the states which are already traversed by the ant $p$. For any state ‘i’:
   - $V_s(i) = 0$ shows that vertex ‘i’ is not visited yet by the ant $p$.
   - Whereas $V_s(i) = 1$ indicates that state ‘i’ is already visited by the ant $p$. 

http://www.ijritcc.org
e) Probability set: Selection of transition depends on the probabilistic value of transition. Since it is inspired by the ant behavior, probability value of the transition depends on the feasibility of transition \( F_0(p) \), pheromone value \( \tau_0(p) \) and heuristic information \( \eta_0(p) \) of transition for ant \( p \). There are \( \alpha \) and \( \beta \), two more parameters which are used to calculate the probability of a transition. These parameters \( \alpha \) and \( \beta \) control the desirability versus visibility factors. \( \alpha \) and \( \beta \) are associated with the pheromone and heuristic value of the transitions respectively. The proposed ant colony algorithm helps to not only get knowledge of the present state but also all the feasible transitions from the current state to the next state and the historical knowledge of the already traversed transitions and states by the ant.

### III. ANALYSIS

In proposed algorithm comparison can be done in three areas:

1. Uses of this approach in the real world,
2. Advantages of the previous work,
3. Advantages of the existing approach in the field of software testing.

ACO approach has been already used in such kind of problem [14], but the main limitation of the work for any real software [14], states are not only important including states, transitions are also equal importance, but in [14] only states are captured, not much worried about transitions. This paper tries to compare the previous work [23] with the proposed modified algorithm. I apply STTACO tool [23], GA [20] and the proposed approach in one of the real time case studies.

### IV. IMPROVED ANT ALGORITHMS

Existing ant colony optimization (ACO) for software testing cases generation is a very popular domain in software testing engineering. However, the traditional ACO has flaws, as early search pheromone is relatively scarce, search efficiency is low, search model is too simple, positive feedback mechanism is easy to produce the phenomenon of stagnation and precocity. This paper introduces improved ACO for software testing cases generation: improved local pheromone update strategy for ant colony optimization, improved pheromone volatilization coefficient for ant colony optimization (IPVACO), and improved the global path pheromone update strategy for ant colony optimization (IGPACO).

At last, we put forward a comprehensive improved ant colony optimization (ACIACO), which is based on all the above three methods. The proposed technique will be compared with random algorithm (RND) and genetic algorithm (GA) in terms of both efficiency and coverage. The results indicate that the improved method can effectively improve the search efficiency, restrain precocity, promote case coverage, and reduce the number of iterations.

Ant colony optimization has a good optimization effect on path optimization, especially traveling salesman problem (TSP) [24-26]. In this paper, I construct ant colony path model to realize the software test case generation. Using the path model presented in this paper, because of a lack of closed loop feedback, the simple ACO (SACO) will tend to move randomly. We improve ant colony optimization to realize generation of higher coverage TC and prove that the improved algorithm is effective for test case generation. Now I will describe the four improved ACO algorithms: ILPACO, IPVACO, IGPACO, and ACIACO.

#### A) Improve Local Pheromone Update Strategy for Ant Colony Optimization (ILPACO)

SACO is an approach, which uses a small constant to update local pheromone. Let \( k \) be an integer, \( 1 \leq k \leq 2N-1 \). I denote by \( V_k \) path node of ant colony and by \( (V_k, V_{k+1}) \) pheromones from node \( V_k \) to node \( V_{k+1} \). I define local volatilization coefficient, \( \alpha \), and define the small constant to update local pheromone, \( \tau_0 \). Based on these, the update process of \( (V_k, V_{k+1}) \) is defined as follows:

\[
\tau \left( V_k, V_{k+1} \right) \leftarrow (1 - \alpha) \tau \left( V_k, V_{k+1} \right) + \alpha \tau_0.
\]

I present an improved local pheromone update strategy for ant colony optimization (ILPACO). Using the method of program instrumentation, I get the coverage to update local pheromone instead of the small constant \( \tau_0 \) [27]. Because the value of the coverage will be smaller than one, it is necessary for me to select a proportionality coefficient \( k' \) to multiply by the coverage to fit with a wider range of applications. So local pheromone update variable \( \tau' \) is defined as \( \tau' = k' \cdot \text{coverage} \). Based on these, the update process of \( (V_k, V_{k+1}) \) is defined as follows:

\[
\tau \left( v_k, v_{k+1} \right) \leftarrow (1 - \alpha) \tau \left( v_k, v_{k+1} \right) + \alpha \tau'.
\]

<table>
<thead>
<tr>
<th>( N )</th>
<th>( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.003</td>
</tr>
<tr>
<td>16</td>
<td>1.054</td>
</tr>
<tr>
<td>80</td>
<td>1.065</td>
</tr>
</tbody>
</table>

The coverage presented can be any kind of coverage. In this paper, I select three commonly used kinds of coverage: statement coverage (SC), branch coverage (BC), and modified condition/decision coverage (MC/DC) [28].

#### B) Improved Pheromone Volatilization Coefficient for Ant Colony Optimization (IPVACO)

I describe IPVACO algorithm. For smaller volatilization coefficient \( \alpha \), the pheromone volatilization is slower; the optimal path will be restrained. In contrast, for the larger volatile coefficient, but not too large, pheromone evaporates...
quickly and the effect of the optimal path is strengthened. Larger $\alpha$ makes the previous experience to be ignored easily and tend to search more by recent experience. In the Saco algorithm, $\alpha$ is fixed.

In this paper, I put forward the adaptive $\alpha$, beginning with a small value tendency to search, and the late tendency to develop with larger values. At the same time, this improved method still uses coverage to update local pheromone.

Let (0) be initial volatile coefficient and let $\lambda$ be a constant value. I denote by $t$ iteration and ($t$) volatile coefficient of the number $t$ iteration. Consider the following:

$$\alpha \ (t + 1) = (t) .$$

I need to determine right range of $\lambda$. Based on the formula (3), it is easy to deduce formula ($t = \lambda t - 1$), and total volatile pheromone of one route is $\alpha(1) \cdot (1 - \lambda t)/(1 - \lambda)$.

The value of desired pheromone of one route is $t \cdot e \cdot N/4$ and $e$ is the value of desired pheromone in one generation. Because most of ants’ coverage ranges from 0.25 to 0.35, I can select a value range from 0.25 to 0.35. And $N$ is the number of ants that update pheromone. In this paper, I select 1, 16, 32, and 80 as $N$. The initial pheromone value of each route is taken as 0.1. Let (1) = 0.05. In Figure 1, line 1 shows the total desired amount of pheromone in one route when $e$ is equal to 0.25. line 2 shows the total desired amount of pheromone in one route when $e$ is equal to 0.35, and line 3, the exponential curve, depicts the amount of pheromone volatilization with different value of $\lambda$. Because the pheromone volatilization should be less than the total amount of the path pheromone. Because I select the value of $\lambda$ in a range, I am not sure whether the value satisfies the condition that the pheromone volatilization should be less than the total amount of the path pheromone. So it is necessary for me to verify if the value is appropriate.

C) Improve the Global Path Pheromone Update Strategy for Ant Colony Optimization (IGPACO).

I describe IGPACO algorithm, which is based on ILPACO algorithm. I use the coverage to update local pheromone, at the same time, and only the best ant can update the amount of Pheromone in each iteration, rather than all the ants [29]. Performance of ant colony optimization, as a swarm intelligence algorithm, depends on the individual ants and overall synergy effect [30]. According to pheromone concentration distribution, ants choose routes. That makes the convergence speed slow. Besides, there are certain possibilities that ant colony fall into local optimal solution prematurely. I put forward the approach that the optimal ant update pheromone can reduce the branch paths effectively, whose coverage is low, make ants avoid unnecessary paths, and improve the convergence speed. Consider the following:

$$\tau \ (v_k, v_{k+1}) \left\{ \begin{array}{ll}
(1 - \alpha) \tau \ (v_k, v_{k+1}) + \alpha \tau'' & \text{if Ant} \ I \text{ is Best Ant} \ (1 \leq I \leq N) \\
(1 - \alpha) \tau \ (v_k, v_{k+1}) & \text{else}.
\end{array} \right.$$  

D) Comprehensive Improved Ant Colony Optimization (ACIACO).

Comprehensive approach combines together methods represented above comprehensively to improve simple ant colony algorithm. The methods represented conclude ILPACO, IPVACO, and IGPACO. ILPACO, as it shows in (2), can strengthen the ant colony path with high coverage. And the higher coverage the path has, the greater the path’s pheromone is, which makes routes have a good gradation with each other [31]. IPVACO, as it shows in (3), improves volatile coefficient increases gradually with the increase of number. Beginning with a small value tends to search and the late tendency to develop with larger values. IGPACO selects the best ant update path pheromone in each iteration to search for a more optimal path and improve the convergence speed.

V. EXPERIMENTAL ANALYSIS

I present the result of the four proposed approaches. C++ language is used for experiment programming. Classic triangle classification (CTCP) and the collision avoidance system (TCAS) are selected as two under test programs. Because CTCP and TCAS have characteristics of many branches and determines, they are suited as under test programs for software test case generation. I introduce the experiment process of CTCP. And, in I introduce the experiment process of TCAS. In two experiments, I compare data by seven approaches, including RND, GA, Saco, ILPACO, IPVACO, IGPACO, and ACIACO. I use the statement coverage, branch coverage, and modified condition/decision coverage (MC/DC), three kinds of coverage, as quality standard of test cases. In order to avoid the contingency, I select average coverage and average minimum generation of all test inputs in 100 runs as the experimental data [32]. I conclude and analyze the efficiency of all improved approaches. I can conclude the performance of seven different algorithms. RND and Saco are random processes and the coverage is lowest. GA is effective to promote coverage of test cases and decreases number of iterations. And the performance of GA is similar to that of IGPACO. ILPACO uses coverage as ant colony pheromone feedback to strengthen the advantage path and coverage of test cases is promoted. Based on ILPACO, IPVACO uses adaptive volatile coefficient and makes volatile speed slow first and then faster. The coverage of test cases generated on IPVACO is better than ILPACO. IGPACO uses best ant to update pheromone. IGPACO has a good astringency. And the minimum number of iterations where branch coverage achieves 100% is much smaller than IPVACO and IGPACO.
ACIACO, can effectively improve the search efficiency, restrain precocity, achieve the highest coverage, and minimize the number of iterations. The proposed technique will be compared with RND and GA in terms of both efficiency and coverage. The results indicate that the improved method can improve the search efficiency, restrain precocity, promote case coverage, and reduce the number of iterations effectively.

VI. CONCLUSIONS

I have improved ACO, establish ant colony search path, and put forward improved approaches: ILPACO, IPVACO, IG PACO, and ACIACO. The proposed technique is compared with RND and GA in terms of both efficiency and coverage. Through the comparison and analysis of CTCP and TCAS, the improved method can effectively improve the search efficiency, restrain precocity, promote test cases coverage, and reduce the number of iterations. The future work will consider following fields. (1) Construct high efficiency ant colony search path. Because ant colony path model is simple, the algorithm astringency is reduced. Establishing efficient ant colony search path can significantly improve test case coverage and reduce the number of iterations greatly. (2) Improve ant colony algorithm further sophisticatedly. For MC/DC such correlation extremely high coverage, I need to think of a more sophisticated algorithm; algorithm should take into account correlation problem of the structures and variables in under test program. (3) Based on ACO, put forward the comprehensive algorithm with other intelligent optimization algorithms. Because of some disadvantages in ant colony optimization, I can use genetic algorithm, particle swarm optimization, artificial bee colony, or another heuristic algorithm to combine with ant colony optimization effectively, mutually complementing each other. The comprehensive algorithm will improve the quality of the software test case generated effectively.

REFERENCES


