Automated Generate Test Sequence from State Transition Diagram Using Ant Colony Optimization

Suchada Ratanakongnate and Phakakarn Makmun

Abstract—This research is created Test Sequence using Unified Modeling Language. State Transition Diagram is behavior diagram, which helps analyzer to see the relationship of each process in software and it can be done before developing real software. The XML document assists in data managements. Thus, various applications can be easily deployed and easy to use. To generate test sequence and find priority of test case by Ant Colony Optimization method (ACO). The function of this method is that the ants will choose the path of the likelihood of a multiple of the amount of pheromone and heuristic values from the path connecting. Starting from start node to the end node, the decrease of amount of pheromones is determined by the rate of evaporation. The advantages of evaporation is able to force the ants avoid selecting local optimal solution and add sub transition search. If the probability is equal to, the ants do not walk in the transition that has been in the past. So to be able to explore other paths which have never been before. Researchers create test sequence this way and focus on the analysis of priorities of test cases so that testers can test the software according to the important of test case.

Index Terms—Test sequence, test case, state transition diagram, ant colony optimization (ACO), extensible markup language (XML).

I. INTRODUCTION

Nowadays, the need of software utilizations is overwhelming. Therefore, software companies try to find a way to evaluate its performance so that customer can trust in the software they have built. However, the lines of code have grown complicated. The benchmarks have to be redesigned to suit with software size variations. Testers have to be precise about the test in order to make sure that the benchmark will work properly and it would not fail on testing some processes.

The research on testing software processes has been continuously studied so as to raise standard on benchmark. The popular methods are Cuckoo Search [1] and Firefly Algorithm [2]. Both researches to establish test sequence and priorities of the test case. The results of the research can be designed to be appropriate. By giving priority to the node over the path. Some path have disappeared, and test cases to test software that is not covered enough. Ant Colony Optimization (ACO) [3], [4] is a way to test software function by imitating the way ants find a route to deliver its food. Analyzing of pheromone value and heuristic value lead to a greater possibility to find better directions to home. This method allows better outcome, proper directions and may lead to a new part which they have never been before. [5]-[8] is a way to analyze the traffic and the traveling salesman can design routes that appropriate the problem by using ACO.

II. RELATED WORK

A. Ant Colony Optimization (ACO)

ACO was founded by Marco Dorigo in 1991, based on an idea that ants usually try to find the shortest ways to deliver its food by measuring pheromone value. The process contains two main steps.

1) Edge Selection: Select a route from the current node to the next node. Analyze that which node is the node from the current node, a node analysis and probability as follow.

$$P_{ij} = \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_k (\tau_{ij}^\alpha \eta_{ij}^\beta)} \quad (1)$$

$P_{ij}$ = The probability of selecting a node $i$ to node $j$.
$\tau_{ij}$ = The amount of pheromone paths between nodes $i$ to node $j$.
$\eta_{ij}$ = The heuristic value of the path between nodes $i$ to node $j$.
$\sum_k (\tau_{ij}^\alpha \eta_{ij}^\beta)$ = Total amount of pheromone and the heuristic path of all the lines connected to the node $i$.
$\alpha$ = The weighted importance of pheromones.
$\beta$ = The weighted importance of the heuristic.

2) Pheromone Update: Once they made a decision on their path, they will adjust pheromone value as follow.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_k \Delta \tau_{ij}^k \quad (2)$$

$\rho$ = The rate of evaporation of pheromones.
$\tau_{ij}$ = The amount of pheromone paths between nodes.
$\Delta \tau_{ij}^k$ = The pheromone from the selected path from node $i$ to node $j$.

B. State Transition Diagram

It is one of many behavior diagrams in UML, used to analyze system requirements, which contains three parts as below.

1) State: is a condition to wait for a particular event.
2) Event: is working to make the system conditions change.
3) Transition: is a path indicates the event.

\[ V(G) = E - N + 2 \]  

\( V(G) \) = Cyclomatic Complexity  
\( E \) = Number of connections between nodes  
\( N \) = Number of nodes

In this study, the researchers use Cyclomatic Complexity to determine the number of paths offered.

III. LITERATURE SURVEY

There are many researches to study and apply Ant Colony Optimization implemented in the software testing or the management of travel and other such research was conducted as follows. [3] Analysis of the flow chart reversed by Ant Colony, research shows that it can create the tests that have been running reverse. And prioritization of testing by giving priority to the pheromone and the heuristic. If the path is formatted as a tree graph. The test will be repeated or loop. [4] Analyzing Activity Diagram of none-loop graph and enhances pheromone value without considering evaporating rate. If it is found that the rate is zero, there is only an option to the destination. In this case, the path could be reused, leading to the overflow. [5] This is search which adapted to find an optimum by Ant Colony method to find the best prediction. First, testers create all possible routes to a destination, which enrich an average time to the finishing line. It is 47% and 56% better used along with Previous Path Replacement (PPR). [9] This research presents Feature Selection Problem. It makes use of non-direction connection and ACO to solve a problem. [10] It is the best to use ACO with Normalized Root Mean Square Error (NRMSE) to measure software reliability. Compared with BPNN, TANN, PSN, MARS, GRNN, MLR, TreeNet, DENFIS, Morlet based WNN and Gaussian based WNN, ACM is the best way to create a benchmark. [11] This research method is used Ant Colony Optimization to build test sequence from UML statechart diagram using amount of pheromones which is a core value in comparison to the next node. The researcher concluded that is the advantages of the UML tools to create test sequence. But this research uses all state coverage criteria which is not the best test coverage. [12] This article describes the behavior of ants, each of the Ant System, Max-Min Ant System, Ant Colony System, which each are suitable for different uses and examples of solutions to traveling salesman. The researcher said that the effectiveness of Ant Colony Optimization is a relatively young metaheuristic which compared to others such as evolutionary computation, tabu search, or simulated annealing. Yet, it has proven to be quite efficient and flexible. [13] This research was presented to create and prioritize the creation of the test sequence using Ant Colony Optimization to create a path that has the appearance of one condition to the other conditions (Decision to Decision – DD graph) of the control flow graph. Without a backward path and return results from the calculation as an example to illustrate the process of creating a path. The research found that can create a path that covers the entire graph.

From study research of the effectiveness of Ant Colony Optimization. Evaluated this method to be applied to the design of the test sequence by using transition coverage and
IV. THE PROPOSED ALGORITHM

This section presents the ways of the ant to make the testing process and the priorities of the test.
1) Generate an XML file from state transition diagram.
2) Input the XML to propose algorithm.
3) Path traversal is created a testing procedure with the following steps.
   • Calculate the complexity of state transition diagram to be used to determine the number of paths.
   • Set initial parameters.
   • Find paths with the ant colony.

Calculating the probability of each transition that connect to the current node (feasible set). To select the next node with a higher probability. If the probability is equal then will find the followings:
The nodes are connected to the current node is whether the end node and has not been selected. Select this node. If not, do the following steps.
The nodes of the feasible set that a node has not been selected. So select this node. If not, do the following steps.
Check the status of all sub-path graph. If that status has not been selected. Select a node connected to the current node with the status of the child is not selected. If not, do the following steps.
Check count the numbers of transition from Feasible Set. Choose a transition that went through more than a number of transition that traverse. If equal, select a state randomly.
• Update Pheromone and heuristic value of each transition. And check start node with equal or end node. If equal, the end stage of this path, reduce complexity and priority of the path. If not, go to the second step.
4) Path prioritization is calculated by multiplying the pheromone and the heuristic value of the transition together. And is divided by the total number of transitions.

V. EXPERIMENTAL DESIGN

1) Create state diagram from examples by using two example systems in Table I. Each example has been working on different things. The more important part is to demonstrate how the proposed system is working with loops and no loops. The Transfer ATM has four loops but Order online has no loop. The transfer ATM has 13 states and 19 possible events. In the other hand, Order online has 23 states and 26 possible events.

2) Save the file as an XML document using Magic Draw. The structure is as follows (Table II).

The structure of XML is the same as Fig. 2 which is described under Fig. 2.
3) Variable used for data analysis in Table III.

4) Create a program to import and analyze data using Visual C# with the ACO.
1. Input: XML File from State Transition Diagram
2. Set parameter : Configuration parameters in Table III.
3. While CC>0 //Define the number of paths.
4. While i <> e //Examines the current state and the end state that are equal or not. If equal, go to step 5. If not equal, go to step 4.1
4.1 Calculate feasible set from state i
4.2 Calculate probability from feasible set

\[ P_j = \sum_{i} \frac{\tau_{ij}^a \cdot \eta_{ij}^b}{\sum_{i} (\tau_{ij}^a \cdot \eta_{ij}^b)} \]  \hspace{1cm} (4)

4.3 If P(ij) <> P(ik) // Compare the probability of feasible set. Select the transition with the probability over.
   If P(ij) > P(ik) Select P(ij)
   If P(ij) < P(ik) Select P(ik)
4.4 If P(ij) =P(ik) //If the probability is equal. Follow these steps:
   4.4.1 If j or k = e // Choosing this transition whether the transition is connected to the end state
   4.4.2 If j and k <> e // If end state. Check the status of state.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td>The weight of the pheromone.</td>
</tr>
<tr>
<td>( \beta )</td>
<td>The weight of the heuristic.</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Evaporation rate</td>
</tr>
<tr>
<td>( \tau )</td>
<td>Pheromone of transition</td>
</tr>
<tr>
<td>( \eta )</td>
<td>Heuristic Value of transition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
<td>xmi:type</td>
<td>Transition Type</td>
</tr>
<tr>
<td></td>
<td>xmi:id</td>
<td>Transition ID</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>Transition Name</td>
</tr>
<tr>
<td></td>
<td>source</td>
<td>Source State</td>
</tr>
<tr>
<td></td>
<td>target</td>
<td>Destination State</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subvertex</td>
<td>xmi:type</td>
<td>State Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No.</th>
<th>System</th>
<th>State</th>
<th>Event</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Transfer ATM</td>
<td>13</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Order online</td>
<td>23</td>
<td>26</td>
<td>0</td>
</tr>
</tbody>
</table>
4.4.2.1 If Vstate = 0 Select this state
4.4.2.2 If Vstate <> 0 //Check parent child transition (Vtrans)
If Vtrans = 0 //Select feasible state of parent state
If Vtrans = 1 //Check count the number of transition from Feasible Set (Cstate)
If Cstate (ij) > Cstate (ik) Select Cstate (ij)
If Cstate (ij) < Cstate (ik) Select Cstate (ik)
If Cstate (ij) = Cstate (ik) Random state
4.5 Update pheromone

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_k \Delta \tau_{ijk}$$ (5)

4.6 Update heuristic value

$$\eta_{ij} = 2 \cdot \eta_{ij}$$ (6)

4.7 Update weight

weight = weight + ($$\tau_{ij} \cdot \eta_{ij}$$) (7)

4.8 Update number of transition

trans = trans + 1 (8)

4.9 set i = j or k (next node) Go to Step 4
5. If i = e Calculate priority and Go to step 6

priority = weight / trans (9)

6. Set CC = CC – 1
If CC > 0 Go to Step 3
If CC = 0 finish and show test sequence

VI. EXPERIMENTAL RESULT

Table IV represents the method outcomes of two systems. The first system has repeated processes, but the second does not. The function of two groups also work under conditional process and unconditional ones. The result shows that test processes differ from the previous case and the ordering processes by priority of test case from Table V. Fig. 3 shows the display screen of application which shows test sequence of test case number 1 of transfer ATM system. And Table VI illustrates the steps of test sequence of path 1 and 2 of Transfer ATM System to compare the alternative transition with loop of path and condition of state.

<table>
<thead>
<tr>
<th>TABLE IV: RESULT OF TWO SYSTEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE V: THE RESULT OF TRANSFER ATM SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC.</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Fig. 3. The screen of application of transfer ATM.

In Fig. 3, Test case no. column shows the number of test cases. Sequence no. column shows the number of test procedures of each test case. State column shows the status
of each test step-by-step sequence relationship continuity from initial state to the final state. Transition column shows the change in condition between the current and previous state.

The change in condition between the current and previous state. Transition column shows the change in condition between the current and previous state.

1) from the node 1, the start node (i).

Table IV. And Table V is the result of each test step-by-step sequence relationship continuity from initial state to the final state. Transition column shows the change in condition between the current and previous state.

The aim of work is to create algorithm which can prioritize the shortest path first amongst the long paths. The reason to give priority to the path with shorter lengths because it will mostly take shorter time. Moreover failure in the early sequence should be test and handling before continuing to test other paths. But sometime the critical path is the path that went through a number of times.

VII. CONCLUSIONS

This research is to study the process of benchmark creation test sequence by state transition diagram. State transition diagram that helps simulate software functionality in aspects of its behavior, that allowing tester to see the relationship between small parts. This method can be done even before software developing processes. First, tested software are converted into XML and test processes are designed by ACO. Start by checking the possible relative to the current state and selecting a next state to the law of probability and to update of pheromones will do when that path is selected. The result shows that the proposed method is proved to be magnificent. It can allow testers to create test case and test sequence. Also the method can be used along with repeat process without any redundancies. There is also the method that can be combined with repeatable processes without making a duplicate of the selected path in the past. However, this method is still needed to be proved in other situations such as compatibility of other diagrams and more.

REFERENCES


Suchada Ratanakongnate got the bachelor degree in computer science from Chulalongkorn University in 1983 and master degree in computer science from National Institute of Development Administration in 1990. Her current position is associate professor of the Department of Computer and Information Science, King Mongkut’s University of Technology North Bangkok. The area of her research is software testing.

Phakakarn Makmun got the bachelor degree from Pranakhon Rajabhat University in 2009 and master degree from the Department of Computer and Information Science of King Mongkut's University of Technology North Bangkok in 2016. Currently she is working as a software developer. The area of her research is software testing.