Extended ACO Algorithm for Path Prioritization

Saurabh Srivastava
Computer science & Engineering
Lovely professional university

Himanshi Raperia
Computer science & Engineering
Lovely professional university

Jastej Badwal
Computer science & Engineering
Lovely professional university

ABSTRACT

Software testing is one major part of software development life cycle (SDLC) and thus tester need to have good testing algorithms in order to test the software correctly and efficiently. Ant colony optimization technique is a meta-heuristic technique which was first proposed by Dr. Marco Dorigo in his PhD thesis in 1992. He proposed a technique which was completely based on the behaviour of ants while taking their food to their colony. In this paper we put forward an extended approach of ant colony method which can be helpful in providing a better path sequence from shortest to longest path, based on the probability calculated using the extended formula. With the help of results we prove that the proposed formula fulfil the requirements.

Keywords

Software, software testing, ant colony optimization algorithm, path sequence, Cyclomatic complexity, probability.

1. INTRODUCTION

Software testing is one of the most important parts of software life cycle and thus it mostly depends upon the tester and the organization which is developing the software. One good thing about software testing is that testers need not to be a programming connoisseur, i.e. a software tester need not be specialized in the area and having skill over the software under test.

There are mainly two types of software testing techniques namely, black box testing also known as functional testing and white box testing also known as structural testing. In functional testing we just give input to the software and verify the output. If the output is correct then we treat software as correct else erroneous, but in structural testing we are concerned about the underlying code which is used.

The popular ways of structural testing includes control-flow based testing which depends upon the control paths. Another way is data-flow based software testing and one other is mutation testing. There can be many paths in a software code which needs to be tested using test cases, but even in a small program there can be very large number of test cases that need to be formed in order to test the program. Thus, an effectual path selection would be viable for testing.

This paper proposes an algorithm which is an extension to ant colony optimization algorithm and helps in effective selection of feasible paths in a graph. There can be N no. of possible path from start to end node in a CFG, and thus, there can be 2^N no. of path that needs to be tested for complete testing, which is quite infeasible. So, we need a proficient path prioritization technique which helps in selecting feasible path not more than once and order the paths which can make testing task easier and lighter.

Ant colony optimization (ACO) is a very notorious technique which was used for several purposes such as path sequencing or shortest path selection in travelling salesman problem and many more. There are quite a few extension of this algorithm given by many researchers which were then used in different field of computer science and mathematics.

In computer science and engineering, ant colony is termed as probabilistic technique in order to solve computational problems. Ant colony algorithm uses graphs for finding the superior paths. Ant colony algorithm was used to generate test sequences for state based testing [1]. This algorithm was used to find the shortest path between the start node and any other random intermediary or destination node [2], this algorithm has loom to cover all the nodes in execution state sequence graph (ESSG) but unsuccessful to do so at higher level or strong level.

Ant optimization technique was majorly applied to the area of testing, where one needs path sequencing in a set of paths to be tested. This algorithm was focused on finding the test data for control flow based testing. A novel approach of testing was given for data flow testing via ant colony optimization algorithm [3].

There are some common extensions of ACO algorithm but, in this paper, the major prominence is given on the selection of shortest feasible path which needs to be tested first in order to get efficient algorithm. As we have discussed earlier, ACO is based on graph thus, we have nodes and edges collectively forming a graph which then needs to be traversed in order to get path sequence which can then be tested after applying test cases.

In ACO we calculate probability of each path and on the basis of probability the priority is measured.

There are four parameters on which probability depends.

1) Feasibility of path (f_{ij})

2) Pheromone value (τ)

3) Heuristic value (μ)

4) visited status (V_s).

Feasibility (f_{ij}) can be defined as the availability of edge from node i to j.

$\mathbf{F_{ij}}$ = 1 if possible path exists from i to j

$\mathbf{F_{ij}}$ = 0 if possible path does not exist from i to j

Pheromone value (τ) pheromone helps ants to make decision in prospect. It keeps a trace from path i to j. The pheromone value is updated after each path is traversed.

Heuristic value (μ) it indicates the visibility of a path for an ant at current vertex i to j.

Visited status (V_s) it shows the status of all nodes traversed by any ant p for any state i.

V_s = 0, node is not traversed by ant p.

V_s = 1, node is already traversed by p.

A node can be simply denoted using N and edges can be denoted using E.

There can be infinite no of paths in a graph if any loop exist on any node, for finding out maximum independent paths in graph we use Cyclomatic complexity, it is a very popular complexity metric used for limiting or finding out the maximum number of independent paths in a graph. An independent path is path which has at least one extra different node from other paths.

Cyclomatic complexity was proposed by McCabe in 1976 which was broadly accepted and helped in finding out the independent paths in a control flow graph [11].

Cyclomatic complexity is defined by V(G). There are three different ways by which Cyclomatic complexity can be computed.

1. V(G) = no. of closed regions in any graph
2. V(G) = no. of predicate nodes + 1
3. V(G) = E – N + 2

Predicate nodes can also be termed as decision nodes.

Combining it with the concepts of vector space and basis (a set of independent paths), we can give that Cyclomatic complexity is, precisely, the least number of paths that can, in (linear) amalgamation, generate all feasible paths through the section [8].

2. RELATED WORKS

There are many applications of ACO algorithm, as we have discussed earlier ACO is a meta-heuristic approach and is used in many fuzzy techniques, sorting and shortest path routing algorithms and many more. In general intellect a person will constantly like to follow the shortest path [9].

In [10] they extended ACO in order to sequence the selection of path that need to be executed first. In that paper they proposed an extended algorithm and calculated strength of the path. The path with greater strength will be prioritized over others on index level, in this approach; for the most part the longer paths will achieve greater strengths which will automatically prioritize the longer path to be executed first, disparate the testers demand.

In [3] the author proposed an algorithm using ACO algorithm which sequence the path on the basis of strength and def-use pairs, this approach also figure out the longer path first unlike the author demand.

Ant colony is also used for finding out the shortest distance between start and end node in a travelling salesman problem. In TSP it is applied as it is using pheromone value and update parameters one can find out the shortest distance between the start and the end node.

Ant colony optimization technique is used for path prioritization most of the time, in [12] the author proposed another technique which removes the shortcoming of [9,11] i.e. selection of larger paths first. In this paper author proposed a cumulative approach of finding out the paths that need to be executed first using test cases. This algorithm uses the path length which can be calculated using the count parameter which will calculate the path or no. of edges traversed by an ant from start to end node.

3. EXAMPLE ANALYSIS

We start with program “triangle” from [14] and will be finding out the CC and infeasible paths in the graph,

```
void Triangle ( int x, int y, int z )
{
bool isTriangle;
1 if ( (x<y+z) && (y<x+z) && (z<x+y) )
2 ifTriangle=true;
else
3 ifTriangle=false;
5 if ( ifTriangle )
{
6 if ( x==y && y==z )
7 printf (“triangle is Equilateral\n”);
else
8 if (x!=y && y!=z && x!=z)
9 printf (“triangle is Scalene\n”);
else
10 printf (“triangle is Isosceles\n”);
}
else
13 printf (“it is Not a Triangle\n”);
}
```

The analysis of the above program “triangle” is as follows:

1) From Fig. 1 we can calculate the Cyclomatic complexity as
   V (G) = e – n + 2 = 17 – 14 + 2 = 5; or
2) V (G) = p + 1 = 4 + 1 = 5.

As the Cyclomatic complexity is 5 there can be a maximum of 5 dissimilar independent paths in the graph shown for program triangle.
These paths are as follows:

P1: 1→3→4→5→13→14

P2:1→2→4→5→13→14

P3:1→3→4→5→6→7→12→14

P4:1→3→4→5→6→8→9→11→12→14

P5:1→3→4→5→6→8→10→11→12→14

Now we will make use of these paths and calculate the probability of each path.

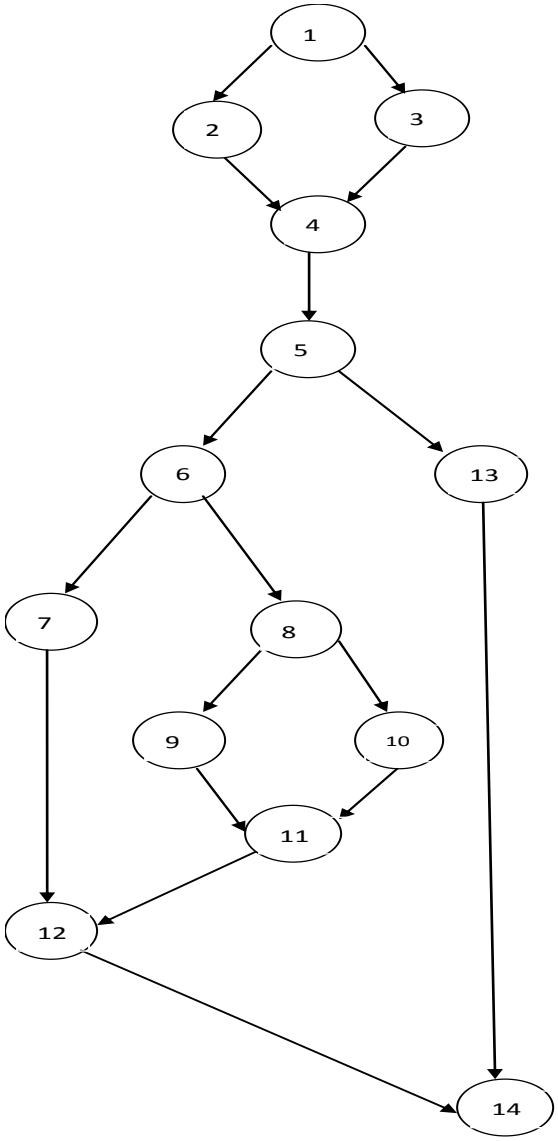


Fig 1: CFG for program Triangle

4. PROPOSED METHOD

Here we are going to present the proposed method which is an extension to [12] and the comparison is done with previous results and the result will show that the proposed approach is better than the previous one.

Ant colony optimization formula is given by:

$$P_{ij} = (\tau_{ij}).^{\alpha} * (\mu_{ij}).^{\beta} / \sum.(\tau_{ij}).^{\alpha} * (\mu_{ij}).^{\beta}$$

Pij denotes the probability of node selection between two nodes, (τ) tau symbol denotes pheromone value and 'mu' (μ) represents the heuristic value, the alpha and beta parameters are the controlling parameters of desirability and visibility respectively. ΣP_{ij} denotes the sum of probability of selection of each node in any given path.

In general,

$$\Sigma P_{ij} = P_{12} + P_{23} + \text{-----} + P_{n-1,n}$$

For testing any path we need all path and branch coverage. By using ACO we can find out probability of selection of any node from the no. of nodes. The node with highest probability will be executed first using test cases.

Let us take a path:

P4:1→3→4→5→6→8→9→11→12→14

In this path P4 we have 10 nodes and 9 directed edges. After finding out the total sum of probability i.e. Pij we can further modify the algorithm by dividing the total probability of a path with the total no. of predicate nodes involved in the particular path. With the help of this we will find out the average path probability denoted by P_{avg}.

$$P_{avg} = \frac{\sum P_{ij}}{\sum N_{ij}}$$

Here N_{ij} represents the total no. of predicate nodes in a path.

The application of the extended formula shows the result helps in improving path sequence.

5. APPROACH OF ALGORITHM

Starting from the first node, here 1, an ant A is placed at node 1. For this program, the algorithm will perform in this manner

Initialize all the values

Pheromone value (τ) = 1

Heuristic value (μ) = 1

Visited status $V_s = 0$

Set Probability = 0

α = 1 and β = 1 both the value are taken as 1, they represents desirability and visibility of any ant P at any vertex i (here 1 at start).

Probability of any path when j to be selected when ant is at vertex i at present is given by:

$$P_{ij} = (\tau_{ij}).^{\alpha} * (\mu_{ij}).^{\beta} / \sum.(\tau_{ij}).^{\alpha} * (\mu_{ij}).^{\beta}$$

Symbols have their respective meaning.

From figure 1 we can easily depict that at vertex "1" there is a predicate node with 2 nodes in competition thus probability will be 1/2 as all values in the above formula are taken as 1. Now let one node "3" get selected after the applying the formula and now here at 3 we can see there is no predicate node thus we will get value of P_{ij} as 1 and so on. At "1" we have a predicate node, thus probability can be computed as:

$$P_{ej} = (1).^1 * (1).^1 / \sum .(1).^1 * (1).^1$$

$$P_{ej} = 1/2$$

In the similar way we can proceed and find out the probability for each node in a any path.

For each path having feasible set from start node "1" to end node "14" we compute average probability using the formula:

$$P_{avg} = \frac{\sum P_{ij}}{\sum N_{ij}}$$

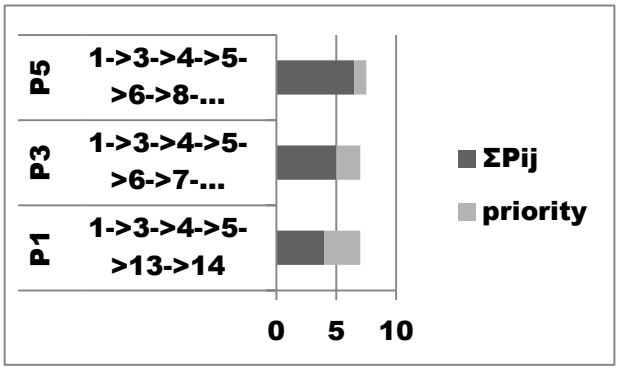
The table below will show how average probability will help in choosing the path out of several paths in the CFG

We show the probability calculation for three nodes P1, P3, and P5 of graph triangle.

TABLE I. **PROBABILITY USING ACO**

path	nodes traversed	ΣPij	priority
P1	1→3→4→5→13→14	4.0	3
P3	1→3→4→5→6→7→12→14	5.0	2
P5	1→3→4→5→6→8→10→11→12→14	6.5	1

The above table shows the normal prioritization of paths on the basis of probability calculation using ACO.



GRAPH 1: RESULT USING ACO

In the graph above it shows priority for path P1 is 3 which is the shortest path in the graph which needs to be sorted out.

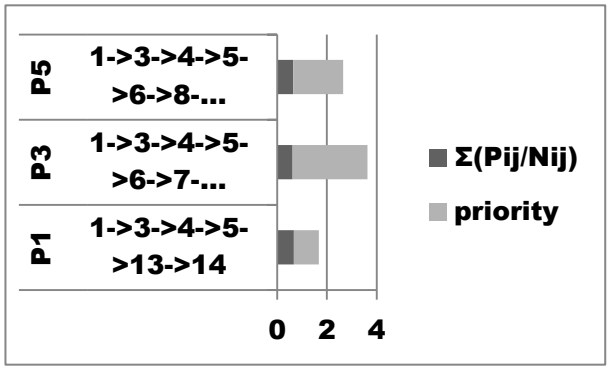
A tester would like to execute shortest feasible path first [16]. Thus, we calculate average probability for the above graph after calculating the probability for individual nodes. Results are displayed in table 2.

Taking P_{ij} value from Table 1 and as we have no. of nodes in each path we can, by using the proposed formula, make the following table.

path	nodes traversed	$\Sigma(P_{ij}/N_{ij})$	priority
P1	1→3→4→5→13→14	0.67	1
P3	1→3→4→5→6→7→12→14	0.625	3
P5	1→3→4→5→6→8→10→11→12→14	0.65	2

TABLE II. **PROBABILITY USING EXTENDED ACO**

In the TABLE II now we can easily figure out that the P1, which is the shortest path in the graph, has the highest probability and now it will be executed first by applying test cases. Length of P3 is less than P5 but probability is lesser for P3 thus P5 will be executed after P1.



GRAPH 2: RESULT USING EXTENDED ACO

Graph 2 shows the bar chart for table 2, now we can depict that the smallest path P1 has the priority 1 so it would be executed first before other longer paths.

6. CONCLUSION

Path prioritization is major necessity to efficiently test all the paths involved in CFG so we can prioritize the paths for testing using ant colony optimization algorithm by prioritizing the paths by calculating probability of selection of each node. In path testing we start from the shortest path first. Thus, the proposed approach allows tester to find out the probability for each path and priority of the shortest path comes out to be maximum i.e 1(first).

7. ACKNOWLEDGEMENT

First of all I would like to thank my parents for supporting me through all phases of my life, then I would like to thank my mentor for being a great motivator and helping me in all aspects, finally I will thank my friends to motivate me for writing the paper.

8. REFERENCES

[1] C. Peng LAM and Huaizhong LI, "An Ant Colony Optimization Approach to Test Sequence Generation for State based Software Testing", Proceedings of the Fifth International Conference on Quality Software (QSIC'05), pp 255 – 264,2005.

[2] Mohan V and Jeya Mala, "intelligent tester–Test Sequence Optimization framework using Multi-Agents",

journal of computers, VOL. 3, NO. 6, Academy Publishers, 2008.

[3] A new data flow testing technique via ACO by Ahmed S. Ghiduk, Universal Journal of Computer Science and Engineering Technology 1 (1), 64-72, Oct. 2010. © 2010 UniCSE.

[4] wiki/ant_colony_optimization

[5] T. Stützle et H.H. Hoos, *MAX MIN Ant System*, Future Generation Computer Systems, volume 16, pages 889-914, 2000

[6] ACO based on ASRank and MMAS for VRPSPD, T zhang 2007.

[7] Arthur H. Watson and Thomas J. McCabe, "Structured testing: a testing methodology using the cyclomatic complexity metric," NIST Special Publication, September 1996.

[8] "An Improved Algorithm for Basis Path Testing" Du Qingfeng Dong Xiao **©2011** IEEE

[9] P. R. Srivastava "An Approach of Optimal Path Generation using Ant Colony Optimization" IEEE TENCON 2009, india 2009.

[10] Thomas J. McCabe "A Complexity Measure" IEEE Transactions on Software Engineering, Volume SE-2, No. 4, 1976.

[11] S.Sriavstava "basis path testing using ant colony optimization algorithm" ICRITO'2013 ISBN: 978-93-81583-85-2

[12] Aditya P. Mathur "Foundation of Software Testing" First Edition Pearson Education 2007.

[13] An Improved Method of Acquiring Basis Path for Software Testing Zhang Zhonglin, Mei Lingxia IEEE-ICCSE 2010

[14] Marco Dorigo "The Ant Colony Optimization Meta-heuristic: algorithms, Applications, and Advances, International Series in Operations Research & Management Science", vol. 57, Springer NY, 2003.

[15] Marnie L.Hutcheson, "Software Testing Fundamentals Methods and Metrics" Posts & Telecom Press, 2007-09