ANALYSIS OF THE ISSUE OF ANT BEING STUCK IN THE CYCLE BY ANT COLONY SYSTEM ALGORITHM & SOLUTION

Dhaval Varia
Gujarat Technological University, Ahmedabad

Dr. A. M. Kothari
Atmiya University, Rajkot, Gujarat. India

ABSTRACT

At the point when the size of the shortest path discovering problem increment, the intricacy of the issue increments. In such a situation, the best possibility to take care of this sort of issue is meta-heuristic methodology. Conversely, notwithstanding the way that heuristic strategies can't locate an optimum solution, they can discover a sub-ideal solution inside an acceptable time limit. Moreover, since conventional algorithms don't have a decent structure for getting away from local optima, they can't converge to a great solution. In this way, the heuristic calculations which utilize random structures for finding solutions have been proposed. Such calculations called meta-heuristic can escape from local optimum points however much as could be expected and unite to good solutions. In this study, we have proposed the procedure to identify and eliminate the cycle in the decision-making process of the ant colony system. The aim of this study is to show the number of ants stuck in the cycle of the graph if eliminated results in improvements of the computational efficiency.

Keywords: Ant colony optimization, loop design problem, shortest path finding algorithm, SUMO, Traci, Ahmedabad City, Meta-heuristic approach

1. INTRODUCTION

Smart vehicle routing problem has become popular in the past decade to solve the issues like shortest path finding problems and capacitated vehicle routing problem. The best possible methodology to take care of this type of issues is meta-heuristic approaches especially Ant colony optimization (ACO). ACO algorithms is inspired by the capability of real ants to find the shortest path from the food source to their nest. There is very little research that happened on solving the issue of ants being stuck in the cycle.

Ant algorithm (M. Dorigo & Di Caro, 1999) (Marco Dorigo & Tzle, n.d.) Is a multi-agent framework in which the conduct of simple ant called artificial ants is motivated by the genuine ants. ACO is a calculation which applied to numerous issues. Ranging from classical traveling salesman problem to routing in telecommunication.

Here the problem is to find the shortest path of graph G=(N,A). ACO algorithm exploits a set of variable T= $\tau_{i,j}(t)$ called artificial pheromone associated with arcs (I, J) of the graph. Their pheromone trail is read and written by the artificial ant. Where $\tau_{i,j} \propto utillization\ of\ the\ arcs\ by\ ants$.

At every node-local pheromone which is stored, used in the stochastic approach to choose which junction/node to pick out of the accessible junction/node. When situated at a node *i* an ant *k* uses the pheromone trails τij to process the chance that p_{ij}^k of picking j as the next junction/node:

$$p_{ij}^k = \begin{cases} \dfrac{{\tau_{i,j}}^{\propto}}{\sum_{j \epsilon N_i^k} {\tau_{i,j}}^{\propto}} \; If \; j \epsilon N_i^k \\ \\ 0 \;\; if \; j \notin N_i^k \end{cases}$$

Where N_i^k is the practical neighborhood of ant k when in node i. At the start of the pursuit procedure, an amount of pheromone τ0 = 1 is assigned to all the arcs of the graph G to avoid division by zero. An ant over and over jump from node to node utilizing its choice strategy until it, in the end, arrives at the goal node. While an ant comes back to the source, it adds pheromone to the edges it crosses: during its return travel the generic ant k deposits an amount $\Delta\tau^{\,k}$ of pheromone on each edge, it has visited. In particular, if ant k at time t traverse the edge (i, j), it updates the pheromone value τij as follows:

$$\tau_{i,j\,(t)} = \tau_{i,j\,(t)} + \Delta\tau^{\,k}$$

By this standard, an ant utilizing the curve associating node i and j build the likelihood that expected ants will utilize a similar edge later on.

Presently the issue is how to compute the estimation of $\Delta\tau$ for the ant k. In Simple ACO this is commonly the same consistent incentive for all the ants.in this case, the marvel works are called differential path length impact: ants which have distinguished shorter way will store the pheromone prior that the ants which find the more drawn-out way. Along these lines, the short way turns out to be more attractive snappier than longer ways and this thus decides an expansion in the likelihood that anticipated ants will pick it.

Other than this constant value for all the ants we may derive some function to deposit the pheromone by ant k on the path it found. So, we required the amount of the pheromone deposited on the path is inversely proportional to the length of the path which ant K travelled. So,

$$\Delta\tau^{\,k} = \frac{1}{L^K} \; where \; L^K \; is \; the \; length \; of \; ant \; k^{'} s \; path.$$

To restrict the ant to choose a suboptimal path, the pheromone trail required to vanish. Evaporation is done by diminishing pheromone trails at exponential speed.

By and by, at every cycle of the calculation, the accompanying condition is applied to all pheromone trails:

$$\tau = (1-\rho) * \tau \; where \; \rho \; is \; pheromone \; evaporation \; rate \; \in (0,1]$$

Following is the overview of the ACO algorithm:

Table 1 Basic Ant colony optimization algorithm suggested by the M.Dorigo & Di Caro

Algorithm 1 The Ant colony Optimization Metaheuristic (M. Dorigo & Di Caro, 1999)

Set parameters and initialize pheromone trails
while termination condition not met do
ConstructAntSolutions
ApplyLocalSearch (optional)
UpdatePheromones
end while

In this paper we have proposed the algorithm to tackle the issue of ant being stuck in the cycle and isolate the nodes which participated in the creation of the cycle.

2. LITERATURE REVIEW

Eshghi and Kazemi (Eshghi & Kazemi, 2006) have proposed a solution for the single loop design problem. The proposed strategy is planned to locate the briefest closed loop along all divisions edges so that a closed circle is a non-crossing circle covering at any rate one edge of every office. This paper has demonstrated the examination of the proposed calculation with different procedures to tackle the single loop design problem.

Zhao & Tong (Zhao & Tong, 2009) analyzed research and discover the answer for the ant being stuck in the loop during looking for the shortest path. In this paper, the author is maintaining tabu-list if the current node is part of the tabu-list than ant will be backtrack and start searching again. This allows subsequent ants to stuck in the same loop. Our proposed solution makes the pheromone value of the specific edge of the path infinite so that the three things can be accomplished (1) Subsequent ants will avoid this path, bringing about the expulsion of the circle from search space (2) No compelling reason to maintain tabu-list (3) back-track of the ant is not required.

Elloumi et al. (Elloumi et al., 2014) have indicated improvements in the presentation for taking care of traveling salesman issues utilizing a Particle swarm optimization altered by the Ant colony technique. This paper has referenced that the ant system framework exhibited effectiveness to determine combinatorial optimization issues over PSO.

3. RESEARCH GAP

In an ant colony algorithm, the ant isn't allowed going in reverse to expel way repeat for example one node cannot be visited more than once by a similar artificial ant. In any case, the issue is, suppose there are n no. of ants, out of which say x no. of ants follow the way 1->2->3->4->5 and stuck each time during looking through the way towards goal D since this way contains a cycle, for example, 1->2->3->4->5->1, is the misuse of the computational asset. Instead of this, if we may identify and seclude the circle going over the way then ensuing ants follow another way to look through the goal. Also, this will permit converging solutions faster. Additionally, this the calculation assets will be spared.

For the most part, in the issue, where looking through the shortest path inside the map of the city, we cannot adjust the map/Graph to expel the repeat of the path. We have to build up another system to identify the

repeat in the graph and segregate it. With the goal that ant keeps away from this way during looking for the optimal way.

In ACR Algorithm (Zhao & Tong, 2009), If ants find the way where there are no nodes, other than choosing a node which makes trap (here it can be identified using taboo list) than ants begin following to past node, ant begins looking through another way. Perception is this calculation permits all the ants to follow this way and afterward backtrack the ants. Which thus overuse the computational assets and postpones convergence of the solution. So, we find another way to build a novel system.

4. IMPLEMENTATION & EXPERIMENTS

The implementation to prove point of this paper is based on the Simulation for urban mobility (SUMO) and Traci (Python). The ant colony system (ACS) algorithm (M. Dorigo & Di Caro, 1999) implements in the Traci. Here the aim is to travel the vehicle from source to destination, using the shortest path. The shortest path finding is done using the Ant colony algorithm. The issue with the ACS is if more and more ants are stuck in the loop, it results in the deposition of pheromone on this loop. This will, in turn, become more attractive and ants would get trapped in this loop. For the purpose of experimentation, we have considered a physical map of Ahmedabad City, Gujarat-India as shown in figure 1.

Figure 1 Considered a graph part of Ahmedabad city

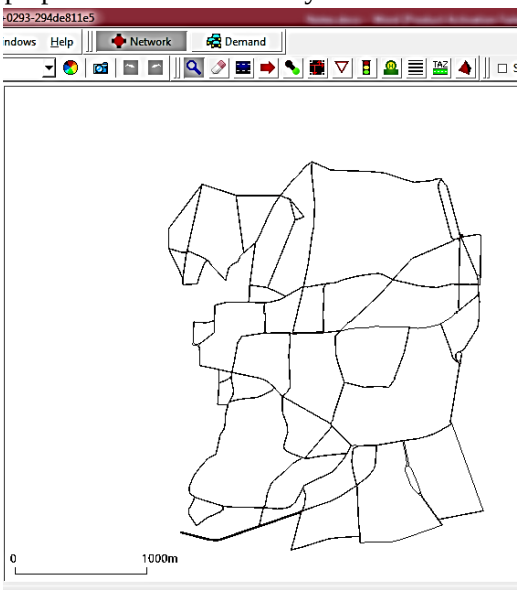
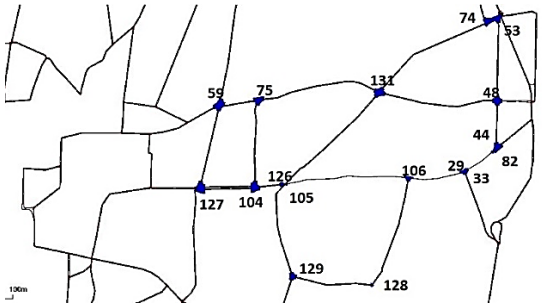


Figure 2 Map to illustrate loops find by the ants during the search of the path from node 106 to 132

Figure 3 Identified Routes contains the loop



Route 1 : 106 128 **129 131** 75 104 126 105 106

Route 2 : 106 29 33 82 44 48 131 129 128 106

Route 3: 106 128 129 131 75 59 127 104 126 105 106

Route 4 : 106 29 33 82 44 48 53 74 **131 129** 128 106

Route 5 : 106 29 33 82 44 48 131 75 104 126 105 106

***Note: There is a direct path between 131 and 129**

For the test-setup, we used 512 ants and the Total journey considered is 39 with 10 iterations each. For the other parameter, we choose fixed default values: $\alpha = 0.5, \beta = 0.5,$ evaporation rate = 0.75, and initially default pheromone set to all the paths are 1. The parameters are in the present model fixed; this makes it simpler to think about the outcomes. In the beginning, all the ants are assigned the source node as the start node of the journey. We have considered the following procedure to run the simulation:

Step 1: Initialize the journey by giving an array of start node and End node [Nodes are the cross-ways of the graph represented in figure 1]:

source_node = [87, 38, 88, 132, 48, 132, 145, 145, 116, 51, 81, 71, 97, 99, 99, 98, 46, 109, 101, 59, 38, 132, 36, 134, 60, 120, 97, 87, 113, 39, 39, 40, 40, 40, 47, 84, 26, 44, 41]
destination_node = [128, 27, 132, 126, 99, 106, 129, 128, 105, 48, 20, 44, 131, 83, 74, 91, 44, 128, 132, 118, 131, 75, 74, 35, 33, 44, 35, 12, 131, 129, 106, 35, 19, 56, 106, 83, 98, 51, 74]

Step 2: Start simulation of Algorithm 2 with given source and destination as (source_node[i], destination_node[i]) where i iterate from 0 to 38 to cover all the 39 journeys.

Note: Number in the array represents the Junction/node of the graph shown in figure 1.

Table 2 shows the proposed algorithm for the detection of the cycle during the selection process of ACS & Table 3 shows the proposed algorithm for the removal of the cycle within the path.

Table 2 Proposed method for the detection of the cycle during the selection process of Ant colony system algorithm

Algorithm 2 Modification in the edge selection process of ACS Algorithm
Input: ants, graph

Output: Detection of the cycle and set pheromone to infinite for all/some of the edges available in the trace

For all the ants
For all the available next_nodes for selection in the graph
If the pheromone of the selected edge is greater than 0
If trace of ant(i) >= 3
If a node in trace
print("Cycle Detected")
Function: set_phermonone_to_inifite for all/some of the edges available in the trace

Table 3 The proposed algorithm for the removal of the cycle within the path

Algorithm 3 Process to set pheromone to infinite for all the edges available in the trace of ant, which is part of the cycle

Input: **trace, graph**
Output: **set the pheromone matrix to infinite for the edges part of the cycle**

```
get_nodeid=0
Loop through all the nodes of trace where loop detected
    Check if any of the nodes in the trace is having more than two outgoing edge
        IF yes then get that get_nodeid
        Break
if(get_nodeId ==0)
 loop through all the nodes of trace
    pheromone[x][x+1] = -1
Else
   pheromonde = default pheromone i.e. 1
   loop through all the nodes of trace
          if trce[i] ==  get_nodeid
             pheromonde = -1
          pheromone[x][x+1] = pheromone
```

In the experiments, our graph contains 145 nodes and 300 edges in total. Decision making for each ant depends on the two matrices: 1. Distance matrix 2. Pheromone matrix. Each matrix is of size 145X145 (Where 145 is the total nodes the graph consists of.) As the ants progress the solution, ants store the node it has visited in the trace array.

Generally, the cycle consists of a minimum of three nodes. So if the trace contains nodes equal or more than three then only there is a chance of cycle. Now if the node to choose is already in the trace of ants as shown in figure 3 i.e. there is a cycle from that node to the rest of the trace. For example, if Trace = [128 **129 131** 75 104] of the ants and the next node to choose is 129, then there is a cycle detected: [128 **129 131** 75 104 129].

Now the Algorithm 3 which we have proposed is the removal of the detected cycle in Algorithm 2. Algorithm 3 has shown that, if all the nodes are having more than two outgoing edges, then there are chances

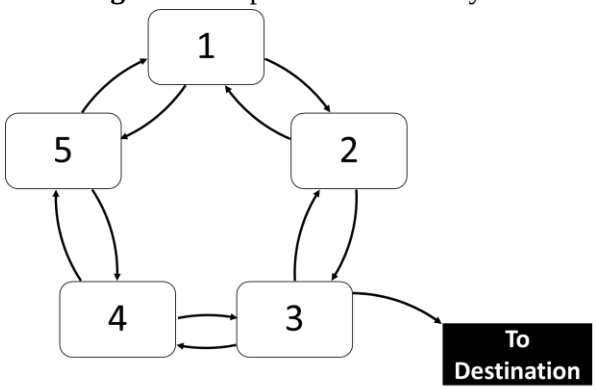
that the ant can move out of the cycle and follow the way towards the goal (refer figure 4). But if the outgoing edges are 1 or 2 then there is zero chance that cycle would bring about the ideal way for the goal.

In this way,

If *none of the nodes in the trace of ant having more than 2 outgoing edges, then set the pheromone of all the respective nodes to infinite.*

Else *Set pheromone to infinite, from the node which more than two outgoing edges.*

Figure 4 The path contains the cycle



For instance, as appeared in figure 4 [Path contains the cycle] If the trace contains [1 2 3 4 5] and the following conceivable node to select is 1 then this is an event of the cycle in the graph. So if all the nodes having only two outgoing edges [i.e 2 to 3 and 2 to 1] then there is no hope to reach the destination. But here node number "3" having 3 outgoing edges [3 to 2,3 to 4, 3 to possible towards destination], so the proposed algorithm will make pheromone to interminable for edges after 3. Edges other than this [i.e. 1-2 and 2-3] will remain as it is.

5. RESULTS AND DISCUSSION

Figure 5 shows the average time t (In seconds), ants spent to find the solution in the journey Ji Where total journey in the consideration is 39 planned for the map given in figure 1. From figure 5, it is observed that the average time spent by each ant gets reduced by some amount. As per the statistics, we obtained this is 5.26% less time taken to solve the problem over the existing Ant Colony system.

Figure 5 Average time t (in Seconds) ants spent to find the solution for all the 39 journey

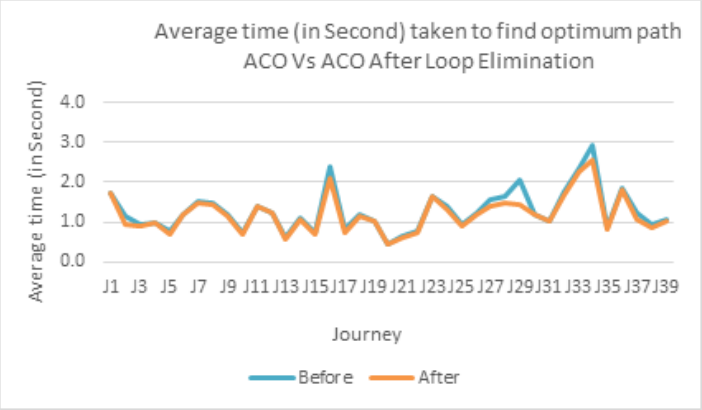
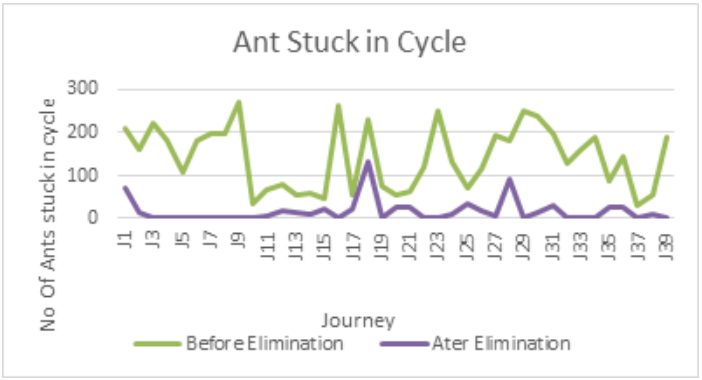
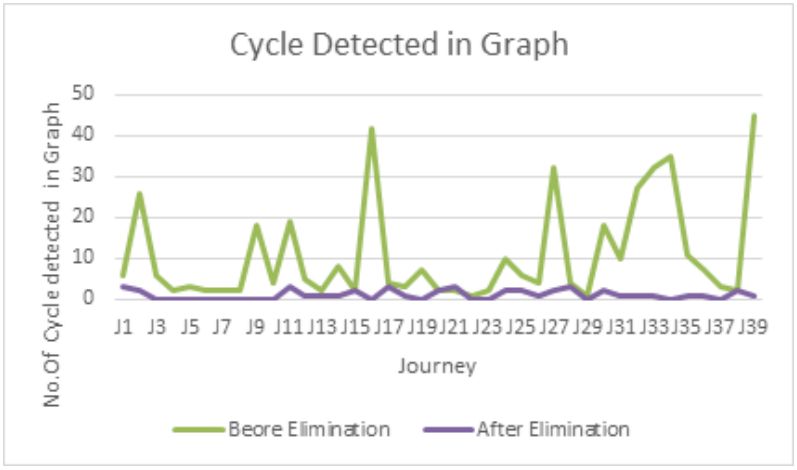


Figure 6 Some ants stuck in the cycle before and after the elimination of the cycle for all the journeys.



As shown in figure 6, the proposed algorithm reduces the ant to stuck in the cycle of the graph. This will let the ants contribute more to generate optimum solutions instead to get stuck in the cycle. As per the experimental statistics, the proposed algorithm reduces 85.4677% of ants to get stuck in the cycle.
Figure 7 shows no. Of cycles detected in the graph before and after the execution of the proposed algorithm. The proposed algorithm detects almost 76 % of cycles in the graph.

Figure 7 Number of Cycle detected in the graph for all the journey



6. CONCLUSION

In this paper, the problem of ant being stuck in the cycle during the decision-making process of an ant colony system has been identified. Our proposed solution detects 76% cycles in the graph and if eliminated, will results in 5.26% less time taken to solve the problem over the existing Ant Colony system. The proposed calculation reduces the ant to stuck in the cycle of the diagram, this will let the ants contribute more to generate optimum solutions rather than stall out in the cycle. According to the exploratory insights, the proposed calculation decreases 85.4677% of ants to stuck in the cycle and improves the computational efficiency.

REFERENCES

1. Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 1470–1477. https://doi.org/10.1109/CEC.1999.782657
2. Dorigo, Marco, & Tzle, T. S. (n.d.). *An Experimental Study of the Simple Ant Colony Optimization Algorithm*. 6.
3. Elloumi, W., El Abed, H., Abraham, A., & Alimi, A. M. (2014). A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP. *Applied Soft Computing*, *25*, 234–241. https://doi.org/10.1016/j.asoc.2014.09.031
4. Eshghi, K., & Kazemi, M. (2006). Ant colony algorithm for the shortest loop design problem. *Computers & Industrial Engineering*, *50*(4), 358–366. https://doi.org/10.1016/j.cie.2005.05.003
5. Zhao, J., & Tong, W. (2009). Solution to the problem of ant being stuck by ant colony routing algorithm. *The Journal of China Universities of Posts and Telecommunications*, *16*(1), 100–110. https://doi.org/10.1016/S1005-8885(08)60187-9