International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 2 440 - 442

Performance Analysis of A*Algorithm and Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem

Shraddha Ramteke
M.Tech Research Scholar, CSE
Chattrapati Shivaji Institute of Technology
Durg, India
shraddharamteke7@gmail.com

Mrs. Deepty Dubey
Associate Professor, CSE
Chattrapati Shivaji Institute of Technology
Durg, India
deepty.shukla@gmail.com

Abstract: Traveling Salesman Problem (TSP) is one of the well-known NP-Hard problem to find the optimal travel route between the two cities among many intermediate cities and multiple paths. Many approaches are used to solve this problem. In this research paper, the commonly known heuristic search algorithm namely A* Algorithm is compared with the Ant Colony Optimization (ACO) Algorithm on the basis of their computational time to solve the Traveling Salesman Problem is given. For less number of cities and routes among these cities, any of the three algorithms can give nearly same computational time. But on increasing the number of cities and routes among these cities, the computational of each algorithm vary from another two algorithms. Therefore, the comparisons have been carried out to know that which algorithm will give the shortest and optimal path in each and every case.

Keywords: *Traveling Salesman Problem, A* Algorithm, Ant Colony Optimization Algorithm*

*****

I. INTRODUCTION

For a traveling salesman, it is most important for him to follow the shortest and optimal path having the minimum cost so as to gain maximum profit and to deliver the goods consuming lesser time as possible. Path optimization is one of the major problems while travelling from the source to destination city visiting each city only once. Path can be calculated using many strategies but to decide the best possible strategy according to the number of cities is a difficult task.

[1]TSP is a combinational problem consisting of a set of cities and a set of edges existing from one city to the other. TSP can be represented by a graph G such that (V,E) ∈ G, where V is the set of vertices (cities) and E is the set of edges(path) between all the two vertices present in the graph. TSP is to find the shortest path in a graph G creating a least cost Hamiltonian cycle. If there exists a path between the two cities i and j, then the distance between these cities d_{ij} can be computed as-

$$d_{ij} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}$$

Input

Network formed from *n* cities

Cost c(i,j) of traveling from one city to next city, w*here* *i* & *j* = 1,2,3 . . , *n.*

Start with initial city.

Output

A least cost Hamiltonian cycle.

II. PROPOSED METHODOLOGY

A. Applying A* Algorithm for solving TSP

A Star is typically the most popular choice for path finding, because it's reasonably flexible and works extremely well in an array of contexts. A Star is just like Dijkstra's algorithm as it enables you to find a speediest path. A Star is like Greedy Best-First-Search as it can make use of a heuristic to manual itself. In the simple case, it is as fast as Best-First-Search. The secret for its success is that it combines the information that Dijkstra's criteria uses (favoring vertices that are near to the starting point) as well as information that Best-First-Search uses (favoring vertices that are near to the goal). In the standard terminology used when speaking about A Star, g(n) represents the complete cost of the trail from the starting point to any vertex and, and h(n) presents the heuristic estimated cost from vertex n to the goal. Each time A Star chooses the vertex n that has the lowest f(n) = g(n) + h(n).

Algorithm-

Procedure Proposed A* algorithm for TSP
Initialize ClosedSet =: empty
Initialize ,OpenSet := start_node
Initialize FromNode := empty
Initialize CoveredNodes := set of all nodes in graph
Initialize g_score[start] := 0
Calculate
f_score[start]:=g_score[start]+heuristic_cost_estimation

Loop Until OpenSet is not empty
Choose the current node from OpenSet with lowest f_score
If current node is not equal to start node **Then**
Reconstruct path from current_node
Remove current node from OpenSet
If current node not in covered nodes **Then**
Add current node to ClosedSet
For Each neighbour in neighbour nodes of current
If neighbour in ClosedSet or in Covered Nodes **then**
Continue loop;
Calculate tentative_score := g_score_current + distance between current and neighbour
If neighbour not in OpenSet or tentative_score < g_score of current **Then**
Update FromNode with neighbour
Update g_score with tentative f_score
Update f_score=g_score+heuristic_cost_estimation
If neighbour not in OpenSet **Then**
Add neighbour to OpenSet
End For
End Loop
End

B. Applying ACO algorithm for solving TSP

[2][3]The Ant Colony Optimization (ACO) technique proposed by Marco Dorigo in 1991 for finding optimal path for Travelling Salesman Problem is a simulation of behavior of ants while searching for food. In the first stage ants wander randomly. Once an ant finds a source of food, it walks back to the colony leaving pheromones as markers, indicating that the path has food. The other ants come across the markers likely to follow the path with a certain probability. Then they populate the path with their own markers as they bring the food back. Because the ants drop pheromones every time they bring food, shorter paths are more likely to be stronger, optimizing the solution. Once the food source is depleted, the route is no longer populated with pheromones and slowly decays.

Algorithm-

Procedure Proposed ACO algorithm for TSP
Set parameters, initialize pheromone trails
Calculate the maximum entropy
Loop /* at this level each loop is called iteration*/
Each ant is positioned on a starting node according to distribution strategy (each node has at least one ant)
For k=1 to m **do /*at this level each loop is called a step */**
At the first step moves each ant at different route
Repeat
Select node j to be visited next (the next node must not be visited by the ant) according to
A local updating rule is applied
Until ant k has completed a tour
End for
Local search (2-opt, 2.5 opt) apply to improve tour
A global updating rule is applied
Compute entropy value of current pheromone trails
Update the heuristic parameter
Until End_condition
End

III. RESULTS & DISCUSSIONS

The discussion about algorithm's comparison can be based on information obtained from the analysis of computational time. Some major results obtained on analyzing the working of three algorithms are:

1. According to the computational time, for less number of inputs, both the algorithms are proved to be efficient and sometimes the calculation time does not vary much.
2. According to the computational time on increasing the number of inputs, Ant Colony Optimization algorithm slows down. In this case, A* algorithm is proved to be more efficient than Ant Colony Optimization algorithm.
3. For extremely large no. of inputs, Ant Colony Optimization stops working goes to unstable state and results in the least cost weight as INFINITY. But in such cases also, A* provides the optimal path for TSP according to heuristics.

Table1: For 30 Nodes

	Ant Colony	A Star
Time (in Milliseconds)	10	4

Table2: For 100 Nodes

	Ant Colony	A Star
Time (in Milliseconds)	Unstable State	80

IV. CONCLUSION

The performance of A* algorithm is dependent upon the selection of a good heuristic function and guarantees to find the shortest and optimal Hamiltonian path within the linear time. On application of correct heuristics to A* technique has proved to be the better technique as compared to the Ant Colony Optimization algorithm for obtaining solution to Travelling Salesman problem.

References

[1] http://en.wikipedia.org/wiki/Travelling salesman problem

[2] M. Dorigo and L. M. Gambardella, The colony system: A cooperative learning approach to the traveling salesman Problem, IEEE Transactions on Evolutionary Computation, Vol.1, No.1, April, 1997.

[3] Zar Chi Su SuHlaing, May Aye Khine; An Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem, International Conference on Information Communication and Management IPCSIT vol.16 (2011).

[4] C-MihaelaPintea, D. Dumitrescu, Improving Ant System Using A Local Updating Rule, Proceedings of the Seventh International Symposium and Numeric Algorithms for Scientific Computing (SYNASC'05), IEEE 2005.

[5] K. S. Hung, S.F. Su, S. J. Lee, Improving Ant Colony Optimization for Solving Traveling Salesman Problem, Journal of Advanced Computational Intelligence and Intelligent Informatics, 2007.

[6] L. Li, S. Ju, Y. Zhang, Improved Ant Colony Optimization for the Traveling Salesman Problem, International Conference on Intelligent Computation Technology and Automation, IEEE, 2008.

[7] Y. Zhang, Z-l. Pei, J-h.Yang, Y-c. Liang, An Improved Ant Colony Optimization Algorithm Based on Route Optimization and Its Applications in Traveling Salesman Problem, IEEE 2007.

[8] MariusDramski, A comparison between Dijkstra's algorithm and simplified ant colony optimization navigation, Scientific journals, Maritime University of Szczecin.