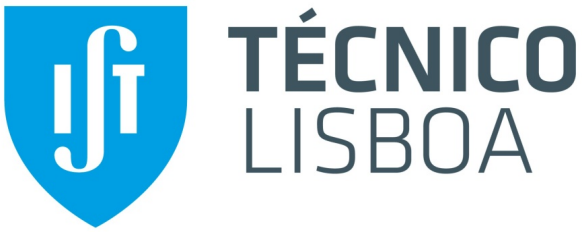


Instituto Superior Técnico

Integrated Master in Electrical and Computer Engineering

Object Oriented Programming

Travelling Salesmen Problem by Ant Colony Optimization

Authors:
António Fernandes, 84370
João Lousada, 84274
Miguel Rodrigues, 84147

Supervisor:
Prof. Alexandra Carvalho

May 9, 2019

Introduction

We study one of the most important combinatorial optimization problems, the Traveling Salesman Problem (TSP), becoming important in operations research and theoretical computer science. It belongs to the NP-complete problems class, i.e., it is NP-hard, meaning it is possible that the worst-case running time for any algorithm for the TSP increases superpolynomially (but no more than exponentially) with the number of cities (nodes on the graph). Therefore, approximations to the optimal solution are required. One possible approach is the Ant Colony Optimization (ACO). Algorithms based on ACO are usually a great tool to study and solve difficult optimization problems on graphs. The TSP aims at finding the shortest Hamiltonian cycle[1] in a weighted graph.

In order to solve this TSP problem, we implemented an ACO based algorithm in Java language. We intended to provide a general framework that, for every (necessary) input, simulates the ant colony and returns the best solution, i.e., the shortest Hamiltonian cycle. All the Java code will be presented throughout this report, where we look for commenting some of the more relevant methods.

As expected, our project receives an XML file correctly validated by an appropriate DTD, following the format described in the project description. Then, for the XML given parameters the simulation runs until the final instance (τ) is reached. For a sufficient large τ we hope to obtain the shortest Hamiltonian cycle.

[1] A cycle is said to be Hamiltonian if it contains all nodes and they occur only once in the cycle, except for the nest node which occurs twice, in the beginning and in the end.

TSP Problem Simulator - Overview

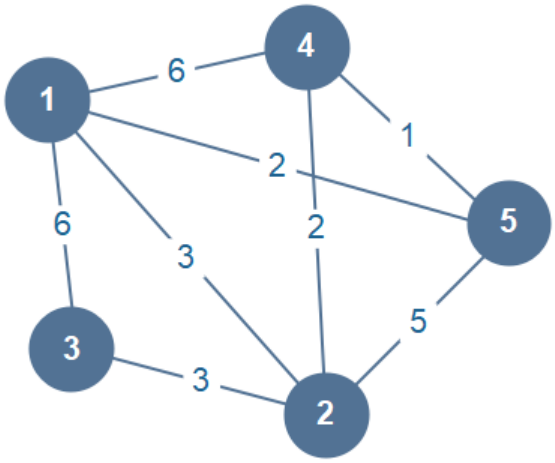
Our project is divided into 6 different packages:

1. ***graph*:** includes a general non-weighted finite graph, as an interface for further implementation;
2. ***pec*:** implements a general pending event container (PEC) system, for any event that extends from Event class, using a PriorityQueue as Data Structure;
3. ***aco_tools*:** includes some auxiliary tools for the simulator package (simulation);
4. ***simulation*:** contains an interface so the simulation can be implemented in a general way. The package also contains the implementation of the interface ;
5. ***eventHandler*:** contains the inherited classes from Event and stores and updates the ant movement.
6. ***xml_utils*:** provides a XML reader, retrieving the required information for simulation;
7. ***main*:** contains the main method, from were we can simulate the TSF problem.

Examples

5 Node Graph - Project Input

This example uses the .xml file given in the project description and serves as starting point of this analysis. It is a simple case , involving 5 nodes where it is easy to check, without the simulation, what the solutions will be.



Here, the most cost-efficient path is {1,5,4,2,3}, with path weight equal to 12, which was also the solution encountered by our simulation, but {1,4,5,2,3}, with path weight equal to 21, is also a possible solution. So it easy to check if the code is implemented according to the rules of the problem, where the Hamiltonian with the lower path weight is stored.

In this case, an estimate of around 600 Move events occurred in each observation event adding to a total of around 12000 total move events. Regarding the evaporation events, for each observation around 100 events occurred. which is expected since there can only be added as many evaporation events as there are edges in a graph.

15 Node Graph - Same Parameters (test_1.xml)

Using the same input parameters (simulation time, pheromone level and ant colony size) a more complex is used to test the validation of the algorithm for a larger Graph, to see if the number of move events and evaporations is proportional to the latter case.

The results obtained for the best Hamiltonian cycle is:
{1, 2, 7, 6, 3, 14, 4, 5, 8, 9, 12, 13, 15, 11, 10}

With a path weight of 30. The number of move/evap events was around 90.000/250 events, for a graph with 3 times more nodes.

15 Node Graph - High Pheromones (test_2.xml)

With a low pheromones threshold, of $plevel = 100$ instead of (0.5) and $\rho = 1$, instead of 10 , making the evaporations less drastic preserving the preferable path.

The results obtained for the best Hamiltonian cycle is:
{1, 6, 7, 5, 4, 10, 11, 15, 13, 14, 12, 9, 8, 3, 2}

With a path weight of 30. So evidently in this case, if the pheromone level is very hgih , one would expect that the first Hamiltonian cycle encountered would be the most attractive for other ants to follow, making it harder to find the best global solution. However, we see that the results for this graph match those obtained in the previous example, therefore high pheromone influence is not relevant to find a viable solution, the algorithm converges either way.

30 Node Graph - High simulation time (test_3.xml)

This case serves as a test to find the best Hamiltonian cycle in a very large and complex graph, using the same number of ants as the first example , but instead of 300 t.u , 2000t.u are used. Also it provides a comparison term with the next example.

The results obtained for the best Hamiltonian cycle is:
{1, 22, 13, 3, 24, 18, 11, 6, 2, 4, 14, 9, 30, 28, 29, 15, 10, 25, 27, 12, 19, 20, 17, 8, 16, 21, 5, 26, 23, 7}

With a path weight of 496, found after 190.000 move events and 1600 evaporations.

30 Node Graph - High Number of Ants and high simulation time (test_4.xml)

Only altering the number of ants from the previous case (ant colony size = 2000), it is expected to find a better solution because the system has more possibilities to test many different paths.

The found solution was a Hamiltonian cycle of the form,after 10 million Move events and 10.000 evaporations:

{1, 13, 22, 3, 18, 24, 11, 6, 2, 4, 28, 30, 9, 14, 29, 15, 10, 12, 25, 27, 19, 8, 20, 17, 16, 21, 26, 5, 23, 7}

With a path weight of 480, which is very similar to the previous case, although smaller. It is worth noticing that more simulations will not always lead to the Hamiltonian cycle found in this one, due to the complexity of the example.

So what we can conclude is that for these 2 examples, a ratio of 10 times the original ant colony size does not contribute significantly to find the best solution, in the same time frame with this kind of complexity.

It is expected that for a smaller number of nodes, having more ants would imply that all possible paths were covered in simulation time, which for this example is not the case.

15 Node graph - High Number of Ants and high simulation time (test_5.xml)

This example is useful to compare if a smaller node graph finds the best Hamiltonian cycle in a situation where we have a bigger colony size.

Using the same simulation time and ant colony size as the previous example, the results obtained for the best Hamiltonian cycle were:

{1, 4, 5, 10, 11, 15, 13, 12, 14, 3, 8, 9, 6, 7, 2}

With a path weight of 30. Meaning that this path is the same of those found in the previous 15 Node Graph Hamiltonians, even though the path found is different in any of the cases, leading to the conclusion that this example is still simple enough to converge very fast.

Conclusion

Summarizing the examples discussed above the main conclusion were found:

→ For the simple case of the 5 Node Graph, we conclude that the algorithm is able to choose between two possible Hamiltonian Cycles, finding the most optimal solution (known apriori),following the rules of the problem.

→ The 15 Node Graph was used to test if the number of total events was proportional to the first case, if a solution was also found and finally , most importantly, to serve as a base for comparison of the following examples.

→ The 15 Node Graph with high level of pheromones has disproved that, for small size graphs, a biased path is preferable by the ants (based on a comparison with previous results), making it more time-consuming to change the tendency and try other paths that may correspond to a better solution.

→ In a graph with a large number of nodes, having 10 times more ants doesn't imply that a better solution is found.

→ For larger size graphs, it is the simulation time parameter that has more influence in the solution.

→ Finding an optimal solution is strongly dependent with choosing the right parameters for the problem.

In conclusion, solving the travelling salesmen problem by ant colony optimization is a very efficient way to find a solution, even if it doesn't correspond to the shortest one. For simple cases, it is very straight-forward to reach the optimal solution in feasible runtime, although as the size and connections of the graph increase, the computational processing power to finish a long simulation is extremely high, making it almost impossible to reach an optimal solution with this algorithm.

One should expect this result when regarding the TSF problem is NP-hard: the time stamp of convergence to the best solution grows (at least) superpolynomially with the number of nodes on the graph. Although, and as previously shown, we can induce a faster convergence. For instance, setting a higher number of ants in the made ACO algorithm, might lead to a faster convergence of the TSF problem (recall we're reproducing a stochastic simulation, thus the higher number of ants does not strictly imply a faster convergence). Besides that, by varying some on the input parameters, such as the pheromone level (as shown), one may reach the optimal solution faster. Hence, for a faster convergence to the optimal solution one would want, for instance, a high number of ants and high level of pheromones.