Travelling Salesman Problem

Ibai Guillén Pacho

Master's Degree in Artificial Intelligence,
Universidad Politécnica de Madrid, Madrid, Spain.

Contributing authors: ibai.guillen@alumnos.upm.es;

Abstract

This article is a summary of the whole work carried out in [1]. The aim was to improve the system presented in [2]. In addition, an implementation has been made in the *python* programming language. The implementation has been carried out over the cities of Argentina, the dataset has been obtained from [3]. Apart from a faithful implementation, different parameter configurations have been tested and improvements on the system have been analyzed.

Keywords: travelling salesman problem, ant colony system algorithms

1 Introduction

The travelling salesman problem is an NP-hard combinatorial optimization problem. The problem assumes a list of cities and the distances between them. The goal is to obtain the shortest route through all the cities, passing only once through each one and returning to the starting city.

This article discusses improvements to [2] ant colony system for the solution of this problem including its implementation in *python*. The implementation, which will be delivered separately is available in the following repository [4], is a *jupyter notebook* with both code and brief documentation for execution and comprehension. This code is an implementation of the system designed by [2] to solve the problem with a set of cities extracted from the [3].

The article is ordered as follows. First, in the 2 section, the details of the dataset used are explained. Then, in the 3 section, a brief summary of the implemented algorithm is discussed. In the section 4, the improvements made to the original system are discussed. The 5 section discusses the experiments carried out, the results obtained and the conclusions. Finally, the future work is discussed in the 6 section.

2 Data set

The dataset used has been taken from the repository of the [3]. The organisation has datasets of instances taken from the World TSP. From all the countries available in the repository, Argentina has been chosen. The first steps that have been taken have been to clean up the file and eliminate duplicities. First, all the unnecessary data was eliminated from the file until only the coordinates of the cities were kept. Then, duplicate cities, i.e., pairs of coordinates that appear more than once, were eliminated. Finally, a small subset of the data has been extracted to work with. For the experiments, I have worked with a subset of data of a size of 15 cities.

3 System behaviour

This section discusses how the algorithm designed by [2] has been implemented. The pseudocode in 1 algorithm is a representation of the implementation made in *python*. For its execution, a set of cities expressed in the form of Cartesian coordinates is required, which is obtained as explained in the 2 section.

The pseudocode of the algorithm will not be explained due to the lack of space, but in case of needing an explanation of it please refer to the extension of this article [1].

Algorithm 1 Travelling Salesman Problem

```
Require: $SetOfCities$
  $InitializeGlobalVariables()$
  for $ant$ in $range(K)$ do
      $LocalPheromone \leftarrow GlobalPheromone$      ▷ Overwrite local pheromone with global one.
      $bestSolution \leftarrow [\,]$
      for each $iteration$ do
          $initCity \leftarrow SetOfCities[0]$      ▷ Get the first city as starting point.
          $addToMemory(initCity)$      ▷ Add city to memory.
          while not $isRouteFinished()$ do      ▷ Complete the route.
              $nextCity \leftarrow searchNextCity(nextCity)$
              $addToMemory(nextCity)$
          end while
          $foundSolution \leftarrow getMemoryRoute()$      ▷ Get route from memory.
          if $cost(bestSolution) > cost(foundSolution)$ then
              $bestSolution \leftarrow foundSolution$      ▷ Overwrite the best route.
          end if
          $updateLocalPheromone(foundSolution)$
      end for
      $updateGlobalPheromone(bestSolution)$
  end for
```

4 Extensions

Several modifications have been made to the behaviour of the system to try to improve its performance.[1]

4.1 Stopping criterion

To improve the performance of the system, one of the measures taken has been to reduce the execution time. To achieve this goal, a stopping criterion has been established. When some ants repeat a path continuously, whether or not it is the best one, the pheromone trail deposited will condition the others, making it very probable that they will repeat it. To avoid wasting time, we can study how many ants are needed to condition a solution, and then, establish that when this number of ants repeats the same path the algorithm stops since the other ants will only repeat it. For this project we worked with a number of 50 ants and a stopping criterion of 4 ants.

4.2 Changes in behaviour

The most important modifications have been made to the behaviour of the algorithm, being one of the parts that has been worked mostly during this project.

4.2.1 Decreasing exploration rate

The exploration rate is decisive in finding a good solution. In the original article, it is static and favours repeating the same paths with slight changes. To solve this, an exploration rate of 70 per cent has been set for the first iteration of each ant, and in each iteration, this rate decreases uniformly until it reaches 10 per cent in the last iterations. In this way, priority is given to exploring new routes, and as they alter the pheromone and the exploration rate drops, better results can be obtained.

4.2.2 Decreasing closeness importance

Decreasing the importance given to the proximity of cities makes the value of the pheromone more important. This has been implemented so that ants tend to choose the busiest roads rather than the closest ones. This is done by starting with closeness importance of 3 and reducing this value uniformly for each ant until a value of 1 is reached.

4.2.3 Modification of global pheromone evaporation

It has been observed that with a global evaporation rate identical to the local evaporation rate, not much route variety is achieved. This is because the first ant strongly conditions the path of the following ants. To avoid this, global

[1]Improvements are considered all those changes that empirically demonstrate an improvement in the execution time of the algorithm or in its performance.

pheromone evaporation rate has been modified so that it becomes half its value. Because of this, the first ants are much less dependent on each other.

4.3 Parameter optimization

Parameter optimisation has been another of the areas where most work has been done during the development of the project.

It is important to have a ratio of one ant for every 2 iterations up to a maximum number of 200 iterations (from 200 onwards no improvement can be appreciated). Based on this, on the one hand, it can be said that for a set of 50 ants it is advisable to have at least 100 iterations for each one of them. On the other hand, if a number of 200 iterations is set, it is recommended to have at least 100 ants, although it could be more.

The high exploration rate may lead to good results in the beginning but over time prevents the algorithm from converging to an optimal solution. The best solution to this problem has been the decreasing exploration rate, starting with a 70 per cent exploration rate and end with a 10 per cent of exploration rate.

Pheromone evaporation is one of the most decisive parameters of the system. The best results have been obtained with an evaporation rate of 10 per cent. Various tests and evaporation rates have been tested. On the one hand, higher evaporation rates increase the dependency on the first route constructed. On the other hand, the lower ones prevent that when a good solution has been found it has a significant effect on future routes.

5 Results and conclusions

In this section, special emphasis is placed on the results obtained after implementing the original system and the improvements. To carry out this analysis, each system has been executed 10 times on a system with 12GB RAM and an Intel Core i5-7200U at 2.5GHz, the results obtained have been noted in the Table 1.

At first glance, one of the improvements that have had the greatest effect on the experiments is the **decreasing exploration rate**. By starting at a very high rate and reducing it uniformly, routes can be found at a lower cost. The downside of this improvement is that it requires more ants on average since exploration reduces the possibility of meeting the stopping criterion in the first iterations of the system, which translates into an increase in execution time.

The effect of **decreasing closeness importance** is a key element combined with the previous one because as exploration decreases, the importance given to the proximity of cities will also decrease. This will prioritise the pheromone value deposited on the different routes taken by the ants. Furthermore, as can be seen in the table, the standard deviation is also reduced with this improvement. This is because when much importance is given to the proximity of the city, the pheromone deposited in the routes is not capable of deciding the choice of the ant and, therefore, the algorithm ends up selecting only from the closest cities. This, together with the first iterations of exploration of different routes, results in a large difference in the distances of the different routes, since the route will initially depend on the pheromone and, if this improvement is not applied, it will depend on the proximity.

The modification of the **global evaporation rate** allows the system to achieve better results than when it is not applied, however, this improvement also does not show as positive results as the previous ones. Given the small difference in performance, further tests should be carried out to verify the true effect of this improvement.

6 Future work

The results obtained for this system are only interpretable for this set of data. For a follow-up of this work, it would be appropriate to take several subsets of data to check how both the improvements and the parameterisation perform. In addition, it would also be necessary to study how the algorithm behaves with a larger population of target cities.

Due to the computational requirements of the implementation and lack of time it has not been possible to perform further experiments to check the stability of the statistics shown and thus demonstrate that the above conclusions are solid. To affirm this hypothesis, a larger number of tests should be performed.

Time is a decisive factor in such a system and, in this case, the execution time required by the algorithm is very significant.

On the one hand, the reason why the system requires so much time should be analysed. It is known that the computational resources available to carry out the tests were short, so to clarify whether this is the reason why the algorithm requires so much time, it should be run on a system with more resources and study the performance.

On the other hand, the implementation should be analysed to optimise it and also reducing times. By substituting loops and other elements and optimising the algorithm, the execution time could be reduced considerably.

References

[1] Guillén, I.: Travel Salesman Problem. GitHub (2022)

[2] Dorigo, M., Gambardella, L.M.: Ant colonies for the travelling salesman problem. Biosystems **43**(2), 73–81 (1997). https://doi.org/10.1016/S0303-2647(97)01708-5

[3] University of Waterloo: World Travelling Salesman Problem. https://www.math.uwaterloo.ca/tsp/world/countries.html

[4] Guillén, I.: Travel Salesman Problem with Ant Colony System. GitHub (2022)

Experiment configuration			Experiment results					
Decreasing exploration rate	Decreasing closeness importance	Modification of global pheromone evaporation	Average number of ants used	Time	Mean	Median	Mode	Standard deviation
X	**X**	**X**	**23.1**	**1h 17m 5s**	**85,083.74**	**84,501.49**	**81,608.79**	**3,476.03**
-	**X**	**X**	**22.1**	**1h 14m 17s**	**92,883.29**	**93,318.15**	**93,318.15**	**4,365.54**
X	**-**	**X**	**30.7**	**1h 40m 54s**	**85,654.49**	**86,336.87**	**81,608.79**	**3,554.56**
-	**-**	**X**	**21.3**	**1h 21m 30s**	**94,105.52**	**93,318.15**	**93,318.15**	**4,667.19**
X	**X**	**-**	**22.6**	**1h 26m 53s**	**85,474.13**	**85,731.34**	**81,608.79**	**2,654.96**
-	**X**	**-**	**18.9**	**1h 9m 54s**	**92,866.85**	**93,318.15**	**93,318.15**	**4,343.37**
X	**-**	**-**	**17.9**	**1h 6m 34s**	**85,857.63**	**86,336.87**	**86,336.87**	**2,342.31**
-	**-**	**-**	**14.2**	**52m 57s**	**93,699.05**	**93,061.56**	**100,535.80**	**4,748.99**

Table 1: Experiments carried out with 50 ants, 100 iterations, $\alpha = 0.1$, $\beta = 3$, stopping criterion of 4 ants and 70% of initial exploration rate in case of decreasing exploration rate or 10% if the rate is constant. Each experiment has been repeated 10 times and the results in the table show the statistics of these 10 runs.