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

In this work the aim was to improve the system presented in [1]. 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 [2]. 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.

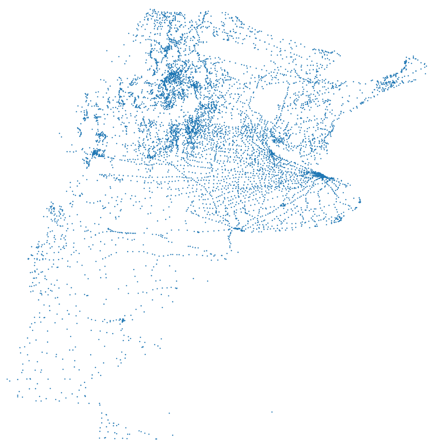
This article discusses improvements to [1] ant colony system for the solution of this problem including its implementation in *python*. The implementation, which will be delivered separately is available in the following repository [3], is a *jupyter notebook* with both code and brief documentation for execution and comprehension. This code is an implementation of the system designed by [1] to solve the problem with a set of cities extracted from the [2].

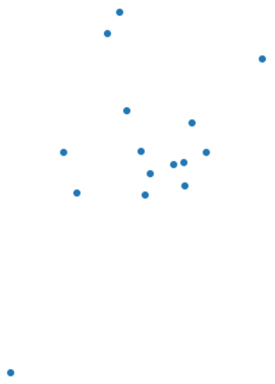
The article is ordered as follows. First, in the 2 section, the details of the dataset used are explained. Then, in the 3 section, the details of the original system and how they have been implemented are discussed. In the section 4, the improvements made to the original system are discussed. The 5 section 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 [2]. The organisation has datasets of instances taken from the World TSP of 25 countries. From all the countries available in the repository, Argentina has been chosen. In this dataset there are 9,152 cities of which only 6,723 are unique, which implies that there are 2,429 duplicate cities. The cities are expressed as Cartesian coordinates and the distance between them is obtained by calculating the Euclidean distance. The cities in the dataset can be seen in Fig. 1a.

Due to this, 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. From the initial 9,152 cities, 2,429 have been eliminated, and from the remaining 6,723 cities, the subset of data has been obtained. For the experiments, I have worked with a subset of data of a size of 15 cities, which can be seen in Fig 1b[1].


(a) Visualisation of all cities in the dataset.


(b) Visualisation of the subset of cities.

It should be mentioned that for some tests a subset of 30 cities has been extracted, but these have only been used to analyse the behaviour of the algorithm and not for the construction and optimisation of the system, and therefore, will not be mentioned further below.

[1]Note: The scale of the Fig. 1b does not correspond to the scale of Fig. 1a

3 System behaviour

This section discusses how the algorithm designed by [1] has been implemented. The aim is to summarise the key points of the article and explain how the ant colony algorithm has been implemented. Therefore, this section is organised as follows. Subsection 3.1 explains how the algorithm works and the implemented pseudocode. Subsection 3.2 explains how the global variables are initialised and for what purpose each one is used. Subsection 3.3 justifies how routes are generated and how destination cities are chosen. Finally, subsection 3.4 describes how the local and global pheromone is updated.

3.1 Algorithm

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, visualised in Fig. 1b.

The pseudocode of the algorithm will be explained in the following sections in detail. First the initialisation of the global variables, then the construction of routes and finally the update of the pheromone.

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
```

3.2 Global variable initialisation

Once the execution of the algorithm begins, the first task is to initialise the global variables. Inside this group are the pheromone (τ) and heuristic (η) matrixes, β, α, K, τ_0, Q_0, M_k, and the iterations to be performed for each ant.

3.2.1 The pheromone matrix (τ) and heuristic matrix (η)

follow the same structure, both are represented as a two-dimensional matrix. The column of each matrix implies the source city, while each row indicates the destination city. Thus, the matrix value $v_{r,u}$ indicates the value for a source city r and a destination city u. For the pheromone matrix the v value of the matrix is the amount of pheromone for the path from city r to city u $[\tau(r,u)]$, while for the heuristics matrix the v value is the heuristic value associated with the same path $[\eta(r,u)]$.[2]

$$\begin{pmatrix} 0 & v_{1,0} & v_{2,0} & \cdots & v_{r,0} \\ v_{0,1} & 0 & v_{2,1} & \cdots & v_{r,1} \\ v_{0,2} & v_{1,2} & 0 & \cdots & v_{r,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{0,u} & v_{1,u} & v_{2,u} & \cdots & v_{r,u} \end{pmatrix}$$

On the one hand, in the pheromone matrix all values are initialised to $1/NumberOfCities$, which in this case equals 1/15. On the other hand, in the heuristics matrix the values are initialised to $1/Distance$, so each value is inversely proportional to the distance of the path from r to u.

3.2.2 β

is used to weigh the importance given to the proximity of cities. Unlike the previous ones, this does not have a fixed value assigned to it, as the improvements made lead to the exploration of values for this parameter, so its value will be developed in the 4.3 section.

3.2.3 α

represents the evaporation rate of the pheromone, it determines by which percentage the pheromone of a path decreases in each iteration. Its value has been defined as 10 per cent for local pheromone updates and 5 per cent for global pheromone updates. In this way, the impact of the best paths found by the first ants does not have much impact on the path selection of the following ants.

3.2.4 K

number of ants. The amount of ants that will build a path in each iteration.

3.2.5 τ_0

is the initial value of pheromone for each path. The value is inversely proportional to the number of cities involved in the problem, so the value set for this hyperparameter is 1/15.

[2]For the management of the distances between cities the same matrix structure has been used.

3.2.6 Q_0

is a constant that defines the probability of an ant taking the best route or exploring another route according to a probability distribution. Many tests and modifications have been performed on this hyperparameter, which will be discussed in the section 4.3.

3.2.7 M_k

is the memory of cities that have been visited. The constraints of the problem imply that a city cannot be visited twice, so it is important to remember which ones have been visited to not repeat and to remember the order to subsequently calculate the distance of the distance travelled.

3.2.8 Iterations to be given by each ant

this constant defines the number of paths each ant must build before finishing its job. Once the ant has finished all the iterations it gives place to the next ant, and so on.

3.3 Route construction

After initialising all global variables, a city is set as the initial city for all ants and is added to the memory. Then the search for the next city that the ant has to pass through begins. This search is conditioned by the value Q_0, since according to the probability of this variable, the best path will be chosen or a new one will be explored according to a probability distribution. In case the exploration criterion is not satisfied, the destination city u will be chosen according to the equation 1. Otherwise, the destination city will be chosen according to a probability distribution, where for each city s the probability of being selected is calculated according to the equation 2. If city s is in the memory M_k its probability will be 0 so it cannot be selected.

$$\arg\max_{u \notin M_k} [\tau(r,u)] \times [\eta(r,u)]^{\beta} \tag{1}$$

$$\frac{[\tau(r,s)] \times [\eta(r,s)]^{\beta}}{\sum_{u \in M_k} [\tau(r,s)] \times [\eta(r,s)]^{\beta}} \tag{2}$$

After searching for the best destination city, it is added to the route and to the memory. This process is repeated until a route that passes through all cities has been created.

3.4 Pheromone update

3.4.1 Local pheromone update

Each time a route is created, the local pheromone matrix is updated according to the equation 3.

$$\tau(r,s) \leftarrow (1-\alpha) \times \tau(r,s) + \alpha \times \tau_0 \tag{3}$$

This process of creating routes and updating the pheromone matrix is repeated as many times as iterations the ant has to perform.

3.4.2 Global pheromone update

Once an ant finishes its iterations, the global pheromone matrix is updated with the best constructed route (the one with the shortest length). The formula for updating the global pheromone differs from the local one and is performed according to the equation 4. Where $\Delta\tau(r,s)$ represents the inverse of the distance of the shortest route constructed, i.e. $1/Distance$ of the best route.

$$\tau(r,s) \leftarrow (1-\alpha) \times \tau(r,s) + \alpha \times \Delta\tau(r,s) \tag{4}$$

Once the global value of the pheromone has been updated, the next ant is taken, and so on until all the ants have built their routes. The route constructed by the last ant is considered the optimal solution to the problem.

4 Extensions

Several modifications have been made to the behaviour of the system to try to improve its performance. Firstly, the 4.1 section addresses the improvement of adding a stop criterion. Secondly, the 4.2 section explains the changes made to the behaviour of the system and how its performance has been improved. Finally, section 4.3 analyses the different parameter configurations that have been explored.[3]

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. After analysing the system in detail it has been observed that it is common that many ants to converge to the same solution. Because of this, it has been established that when some ants take the same route without interruptions, this is taken as the final route, so that it would not be necessary to wait for all the ants to build their paths.

This idea is based on the fact that 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. 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.

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

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. The most relevant modifications are the implementation of a decreasing exploration rate, a decreasing closeness importance rate (in order to give more relevance to the pheromone value) and a pheromone update function with different evaporation coefficient.

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 $\beta = 3$ and reducing this value uniformly for each ant until a value of $\beta = 1$ 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, the equation 4 has been modified so that α becomes $\alpha/2$. With half the value of α, 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. The effects of the different parameters on the overall performance of the system have been explored and the following conclusions have been drawn.

It is important to have a ratio of a number of ants and iterations that is not too unequal. This implies that very little can be achieved with few ants and many iterations or with many ants and few iterations. It is true that better performance is found when there are more iterations than ants and the ratio is 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 implemented improvements, which allows benefiting from the exploration of new routes. Due to the effect of the pheromone, if these routes are considerably better, they will be more likely to be chosen in the next route when there is less probability of exploration. The parameters discussed in the section on this improvement explain that the best values are to start 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, which is detrimental to route exploration and to obtain an optimal result. 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.

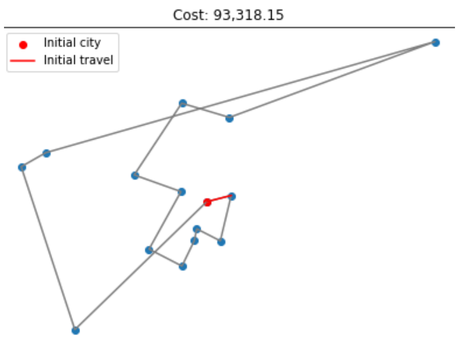
At first glance, one of the improvements that have had the greatest effect on the experiments is the **decreasing exploration rate**. By starting at a very high rate and reducing it uniformly, routes can be found at a lower cost. The downside of this improvement is that it requires more ants on average 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.

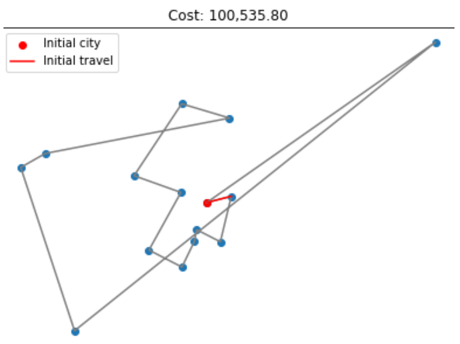
The modification of the **global evaporation rate** allows the system to achieve better results than when it is not applied, however, this improvement also does not show as positive results as the previous ones. Given the small difference in performance, further tests should be carried out to verify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

5.1 Original system routes

The most relevant final routes generated by the original system are those present in Fig. 2. Both final solutions, the one in Fig. 2a and the one in Fig. 2b, occur with the same frequency in the set of experiments performed (3 times each in the 10 experiments performed). However, the route in the Fig.2b is not only a common final route but also the most frequented route by all the ants in the system.



(a) Most common final solution.



(b) Most common final solution and most frequented route for ants.

Fig. 2: Routes generated by the original system.

5.2 Upgraded system routes

The most repeated route by the ants in the improved system and the most repeated final solution have been the one in Fig. 3.

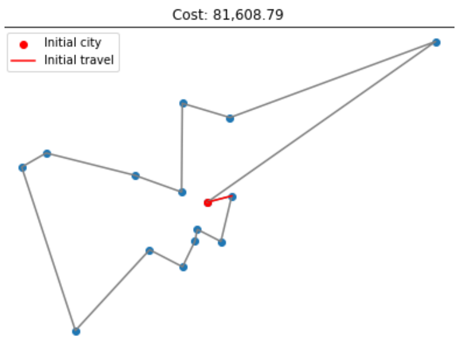


Fig. 3: Most common route and final solution generated by the improved system.

6 Future work

The main conclusions are present in the previous results section (5), so this section will focus on 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. In the experiments carried out, as previously mentioned, only a population of 15 cities was used, so to evaluate the improvements presented above as real improvements, they should also improve the performance of the algorithm with a larger population.

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] 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

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

[3] Guillen-Pacho, I.: Travel Salesman Problem with Ant Colony System. GitHub (2022). https://github.com/Ibaii99/TSP-ACS

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.