Ant Colony Optimization

Athavan Jesunesan

November 2022

1 Introduction

In the field of computer science, there are plenty of example optimization problems. One, for example, is the Travelling Salesman Problem (TSP). This problem, as famous as it is, is very important to understand and the solution has plenty of utility in real-world situations. In TSP there is a given graph that represents the paths that one can take from a given city to another, in this case, from a given city to any other city. Traveling from one city to the next, just like in real life, has a cost. When trying to travel from city A to city B sometimes the optimal solution is to go to city C before city B. The goal of TSP is to find the shortest path which includes a visit to each city without revisiting a city. The solution may appear trivial at first with a brute force method, however; when looking at larger samples of cities it is quickly apparent that it is complicated. The solution to such a problem can be very useful. For example when considering how an amazon driver must deliver packages to several locations this solution is applicable. One proposed solution is to use an Ant Colony Optimization (ACO) algorithm.

2 Background

ACO is based on the way that ants communicate with one another when searching for food. While traveling to and from the nest and the food, ants will deposit pheromones on the ground. This creates a pheromone trail. The other ants then pick up on these pheromones and choose, probabilistically, paths marked by strong pheromone concentrations. This pheromone trail-laying and following behavior is the basis of the ACO algorithm [Mar04]. There are four main components to this algorithm; heuristic matrix, pheromone matrix, probability matrix, and ants. The algorithm once all the components are fully composed, looks like the following:

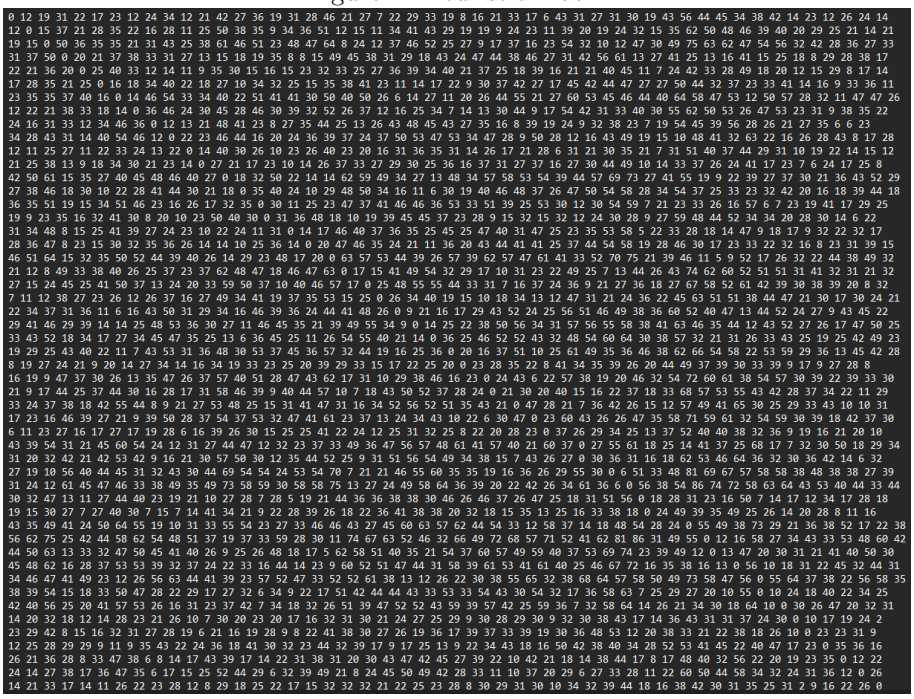
$CreateHeuristicMatrix()$
$CreatePheromoneMatrix()$
$CreateProbabilityMatrix(HeuristicMatrix, PheromoneMatrix)$
while $NotMaxGeneration$ **do**
    $CreateAnts(ProbabilityMatrix)$

$UpdatePheromoneMatrix()$
$UpdateProbabilityMatrix(HeuristicMatrix, PheromoneMatrix)$
end while

2.1 Heuristic Matrix

The heuristic Matrix contains all of the costs of traveling from one city to the next. Figure 1 is the heuristic matrix that was used in this experiment. The first row, second entry represents the cost of going from the 1st city to the second city. This is also why the entry in the second row and the first entry are the same. There are 51 cities in this data set and so the matrix is 51x51. This will be the case for the other matrixes too since they are based on this matrix. Depending on the format of the data set, the methods to create this matrix will vary.

Figure 1: Heuristic Matrix



2.2 Pheromone Matrix

The pheromone matrix contains the pheromones that are on a given edge from one city to the next. The pheromone matrix requires two functions; creating and updating the pheromones. Initializing the pheromones in the matrix is done completely randomly. A value from 0 to 1 is created and placed along

each edge in the matrix (Algorithm 1). This randomness provides a good and realistic starting point for the ants to find the optimal path. After the ants have traveled the edges and created a path, the next generation will do the same. To ensure that more optimal paths are taken and poor solutions are not encouraged the pheromone matrix must be updated [Mar04]. When updating the pheromone matrix there is an evaporation rate that is applied to the values. "Pheromone evaporation reduces the influence of the pheromones deposited in the early stages of the search, when artificial ants can build poor-quality solutions" [Mar04]. First, a matrix of the edges that were traveled on is created. The values of this matrix align with the number of ants that traveled on it divided by the number of cities, in this case, 51. Then, all the values in the pheromone matrix are multiplied by one minus the evaporation rate. Finally, this newly updated pheromone matrix is combined with the values in the edge matrix to make the new pheromone matrix for the next generation (Algorithm 2). This pheromone matrix and the heuristic matrix are then used to update the probability matrix which will then be used by the ants and the cycle continues.

Algorithm 1 Create Pheromone Matrix

```
for i = 0; i<51; i++ do
    for j = i; j<51; j++ do
        if j == i then
            PheromoneMatrix[i][j] = 0
        else
            rand ← Random Number from 0 to 1
            PheromoneMatrix[i][j] = rand
            PheromoneMatrix[j][i] = rand
        end if
    end for
end for
```

Algorithm 2 Update Pheromone Matrix

```
EdgeMatrix[i][j]
for i = 0; i<51; i++ do
    for j = i; j<51; j++ do
        if j == i then
            PheromoneMatrix[i][j] = 0
        else
            value = PheromoneMatrix[i][j]*(1-rate) + EdgeMatrix[i][j]
            PheromoneMatrix[i][j] = value
            PheromoneMatrix[j][i] = value
        end if
    end for
end for
```

2.3 Probability Matrix

The probability matrix is what the ants use to determine which path they should choose next. The implementation of this is trivial once the pheromone and heuristic matrixes are created (Algorithm 3). To create the matrix the value of the corresponding edge in the pheromone and heuristic are multiplied together. The only thing to note is that the values are to the power of alpha and beta. These alpha and beta values can drastically change the outcome and which can be seen in the analysis.

Algorithm 3 Create Probability Matrix

$ProbabilityMatrix[i][j]$
for `i = 0; i<51; i++` **do**
  for `j = i; j<51; j++` **do**
    $value = (PhermoneMatrix[i][j])^a * (1/HeuristicMatrix[i][j])^b$
    $ProbabilityMatrix[i][j] = value$
    $ProbabilityMatrix[j][i] = value$
  end for
end for

2.4 Ants

All the matrixes that have been discussed serve one purpose, to ensure that the ants can make the most informed decisions and find the optimal path. The ants use the probability matrix to decide the next node. This probability matrix has already been refined and designed such that the ant is most likely to take a path that other fellow ants have taken and that these paths are more likely shorter. The way the ants choose is by first choosing a random city to start with. Then take the row of that random node. Create a random value from 0 to 1 and choose the node in which the sum of all prior cities in the row is greater than this value (Algorithm 4). The result is the i-1 node in the row (Algorithm 4).

Algorithm 4 Ants

$rand \leftarrow$ `Random value from 0 to 1`
$sum = 0$
$i = 0$
$row =$ `Probility Row of Chosen Node`
while $rand > sum$ **do**
  $sum+ = row[i]$
  $i++$
end while

3 Experimental Setup

The experiment conducted included all of the functions listed above. In this experiment, I had an evaporation rate of 40 percent. The set of alphas and betas were (0.5, 0.5), (0, 1), (1, 0), and (0.2, 0.8) respectively. Using these parameters and the function listed, this experiment should be replicable.

4 Results

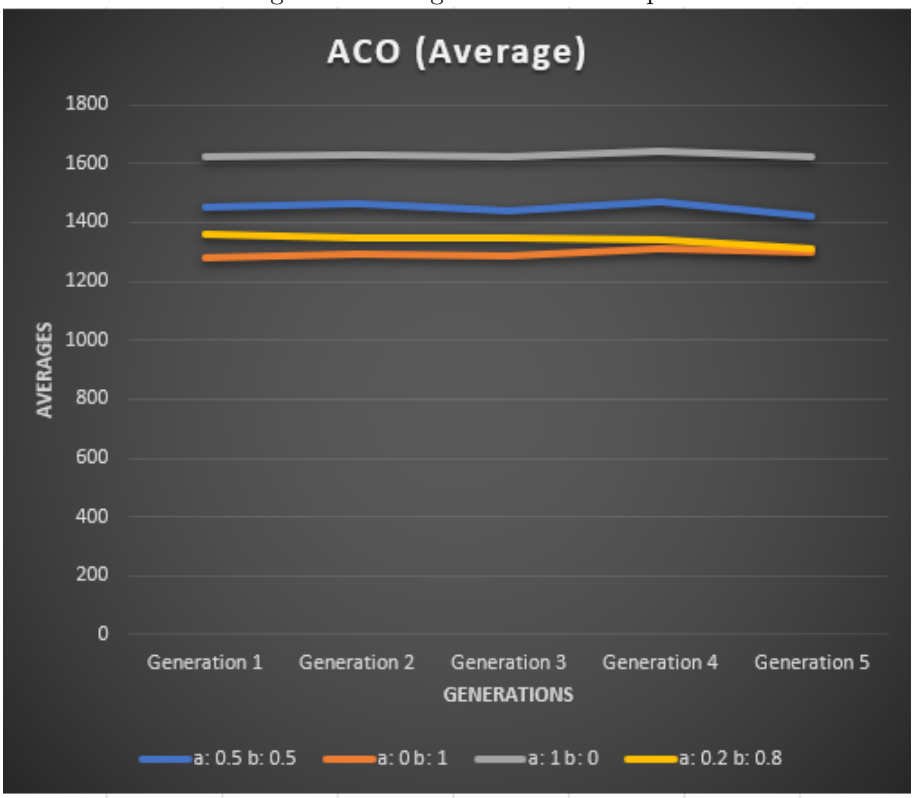
After conducting the experiment I was met with some surprising results. Due to the large values, the graph does not do not justify the differences in the result well. The table does a better job of illustrating the increase. Starting with figure 2 the average started off high and then went higher in the next generation for the first set. There from this point, the generations seem to repeat this decline and then incline while ending on a minimum. This set has my rises and falls which seems to be the pattern with the averages. This first average finishes better than it started which is also the case for set 4. The set with the lowest minimum is set 2, however; this set finished off worse than it started. The set with the worst average path was set 3 with an alpha of 1 and a beta of 0. The set with the best standard deviation was set 1 with 17.7. Set 4, which performed similarly, has a standard deviation of 15.3 which came in second (Figure 3). Looking at how the best ants performed, the best solution had a travel of 1009 (Figure 5). The graph for the best was much better at illustrating the results as there was much more variation. Although set 1 performed better than set 2 in the averages, it performed better in terms of the best solution. Set 4 came in second for the averages, however; it performed much better than set 1 in terms of the best solutions. Set 4 has a standard deviation of 62.1 whereas set 1's is 32.1.

5 Discussions and Conclusions

For this experiment, I implemented the functions as described and used various sets of values for the alpha and beta. There were multiple sets that were used but I found these results to be the most interesting. Set number 4 seems to have performed the best out of all the values that were used in this experiment. Although it did have the best performance in the averages, it heavily outweighed the competition in the terms of the best solutions. Set 4's standard deviation of 62.1 was a clear sign that this combination performed better for this experiment. Set 1 had 1200 in generation 1 whereas set 4 had 1146. By generation 5 that gap changed from 54 to 183. This result was more than 3 times better than that of set 4. Set 4 and set 2 performed better than set 1 and both of these sets have larger beta values. Set 3 performed the worst out of all the sets and had the largest alpha value. I came to the conclusion that a larger beta value performs better than a larger alpha value. I believe this to be the case since the larger the beta value the more a poor heuristic is punished in terms of

the probability of being chosen by the next generation. However, setting alpha completely removes the entire nature of the ant's pheromones, which makes calling this an ACO redundant. Although different evaporation rates were not tested, I can imagine that any reasonable rate would provide similar results. A higher evaporation rate may provide better results in cases where the alpha is larger, as this would result in poorer paths having reduced probability. Overall these results were as expected and it seems that for this problem set, having a larger beta with a reasonable evaporation rate (such as the 40 percent used) is the optimal choice to solve the TSP.

Figure 2: Average Generation Graph



References

[Mar04] Dorigo Marco. *Ant Colony Optimization*. 2004. DOI: https://doi.org/10.7551/mitpress/1290.001.0001.

Figure 3: Average Generation Table

Ant Colony Optimization (Average)				
Alpha	0.5	0	1	0.2
Beta	0.5	1	0	0.8
Generation 1	1450	1281	1622	1358
Generation 2	1465	1292	1628	1346
Generation 3	1440	1288	1621	1345
Generation 4	1469	1313	1640	1342
Generation 5	1420	1300	1626	1312
Min:	1420	1281	1621	1312
Max:	1469	1313	1640	1358
Median:	1450	1292	1626	1345
Standard Deviatio	17.76963702	10.97998179	6.8	15.30490118

Figure 4: Best Generation Graph

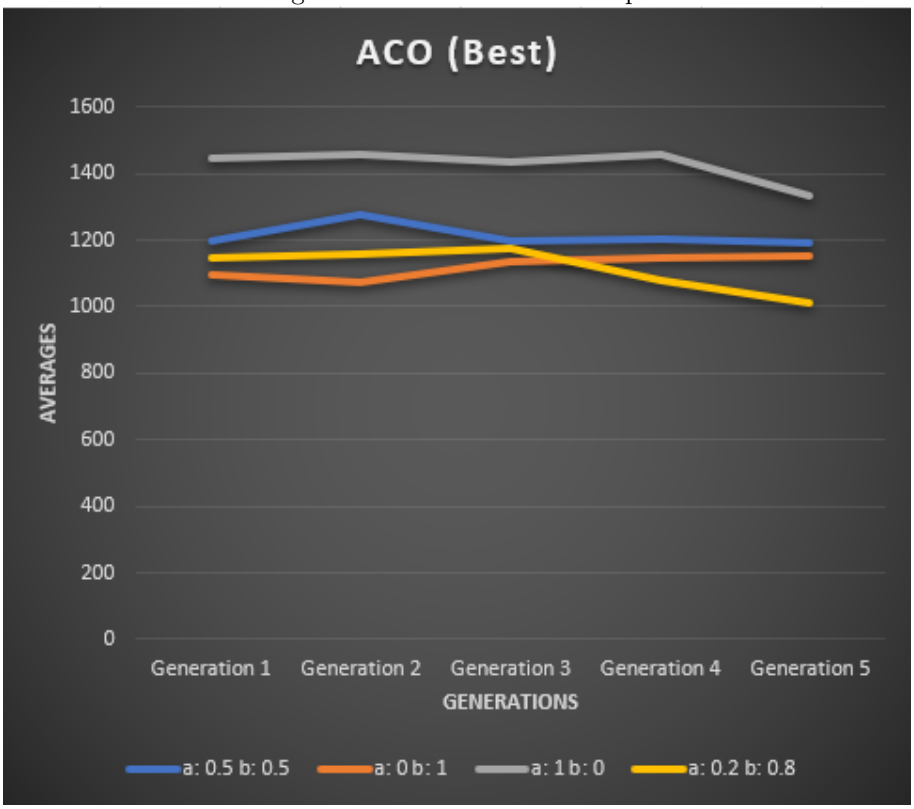


Figure 5: Best Generation Table

Ant Colony Optimization (Best)				
Alpha	0.5	0	1	0.2
Beta	0.5	1	0	0.8
Generation 1	1200	1095	1446	1146
Generation 2	1278	1074	1458	1159
Generation 3	1198	1135	1432	1176
Generation 4	1202	1146	1456	1077
Generation 5	1192	1154	1330	1009
Min:	1192	1074	1330	1009
Max:	1278	1154	1458	1176
Median:	1200	1135	1446	1146
Standard Deviatio	32.17452408	30.96707929	48.08991578	62.10507226