COSC 3P71 - Assignment 3 Report

Shubham Amrelia
Brock University
sa19ss–6846877
COSC 3p71 – Assignment 3
December 04, 2022

I. Introduction

Evolution and animal behavior have been very important for the evolution of real-life practical algorithms that have helped create various smart machines such as robot vacuum cleaners and so on. Genetic algorithms that have been discussed in the last assignment show how can we improve the certain algorithms using chromosome selection, crossover type, crossover rate and mutation as well. One such branch of biology that has inspired a lot of algorithms is Swarm Intelligence. The idea is fairly simple, for instance, ants are tiny creatures with a very limited amount of intelligence as compared to humans, chimpanzees or dolphins. They live their life and interact with other ants based on simple principles. They do not have a cognitive brain but only carry simple actions such as, walking, eating, and carrying stuff. To imagine a world of humans behaving like an ant sounds …. uneventful and chaotic.

So, how do these ants survive and navigate their way through different part of the space? This is where Ant Colony Optimization comes in. These ants while searching for food, drop a chemical called *pheromone*. Ants can smell pheromone so, whenever they smell a strong amount of pheromone, they explore thee places ***instinctually***. Hence, strong pheromone means more and more ants will follow the path. However, pheromone tends to evaporate over the time, it is important to note that if an ant travels from A to B and it a long path, then the pheromone tends to evaporate and the strength of pheromone decreases. Hence, shorter paths lead to stronger pheromone levels, resulting in more ants following them.

Here, we are trying to solve the famous ***Travelling Salesman Problem (TSP)*** using Ant Colony Optimization (ACO). TSP consists of a set of cities such that each city is connected to every other city, *directly*. The distances between that cities are also called *weights* in terms of graph theory. The problem is to find a **complete** path from one city to another city such that **every city** is **visited only once,** and the path has a **minimal distance** as compared to other such paths. For this experiment, I have been given with a file that contains 51 cities, each having 'x' and 'y' coordinates to map their locations. My goal is to reach the input and calculate the distance from each city to every other city (as all cities are connected to each other) and then introduce the ants, pheromones, heuristics, weight of pheromone and heuristic and probability for moving forward.

II. Background

Following is the way my application to solve TSP using ACO works:

- Set the following crucial parameters to a value
  - α (weight of pheromone)
  - β (weight of heuristic)
  - ρ (evaporation constant = 0.95)
  - random seed
  - generation size
  - number of ants
- Scan the given file input and extract the x and y coordinate values and store them in separate array lists

- Initialize the pheromone matrix and fill it up with a random **double** value between 0 and 1
- Calculate the ***Euclidean*** distances between all the cities and make a heuristic matrix for storing the values
    - It is important to note that CityY,CityX will have the same distance a CityX,CityY as they are directly connected
    - When CityX = CityY the value is set to a high value 999999
    - This will result in distance/heuristic matrix to be diagonal
- Then, I make the probability matrix according to the given formula

$$P_{ij}(t) = \tau_{ij}(t)^{\alpha} \left(\frac{1}{d_{ij}} \right)^{\beta}$$

    - Here, the first variable is the pheromone value at a given (i, j) position and second value is the inverse of the distance of cityI to cityJ.
    - α is the weight of the pheromone and β is the weight of the heuristic distance
- Thereafter, I run a for loop that covers the generation of given number of ants and finds the best and average tour lengths for each generation
- My pseudocode is as follows:
    - Initialize heuristic, pheromone and probability matrices
    - Initialize array lists for storing paths per generation and best paths for each generation
    - for each generation {

```
  constructAntSolution ;
  update_Pheromones;
  update_Probabilties ;
  get_best&avg_tourLength ;
}
```

    - For the run: Get the best path and its weight

Following sections contain the explanation of my methods.

A. *getDistance()*

Initialize a heristic distance matrix of size 51 x 51 and fill it up with the distances of CityX to CityY for all X, Y -> [0,50]. Note that it is 0 to 50 and not 1 to 51 because X and Y are indices.

B. *getPheromones()*

Initialize the pheromone matrix and fill it up with a random **double** value between 0 and 1.

C. *getProbability()*

This method makes the probability matrix according to the given formula

$$P_{ij}(t) = \tau_{ij}(t)^{\alpha} \left(\frac{1}{d_{ij}} \right)^{\beta}$$

Here, the first variable is the pheromone value at a given (i, j) position and second value is the inverse of the distance of cityI to cityJ.

α is the weight of the pheromone and β is the weight of the heuristic distance.

D. *constructAntSolution()*

This is the most important method/function in my code as it makes a new path for all the ants per generation. *This method can be found running in pseudocode for the number of generations*. It works as follows

- Start will a for loop that runs for the number of ants there are
- Firstly, create a counter and a temporary arraylist for storing a single ant's travel path
- Second, a while loop runs while the temp arraylist size is not 51 as it is supposed to visit all the cities
- If the counter is 0 i.e., while starting out, the currentCity will be a random city between [0,51) and add the city index to the temp path. Note that this is because method is using indices and not actual city numbers [1,51]

- If the counter is not zero i.e., path already started, the currentCity will be the latest value of temp path
- NOTE: The illegal paths are not considered for choosing (explained further below)
- Make a copied list of the probability row of the current city from the probability matrix
- Update the copied list and set the new probability = probability/(sum of probabilities). Do this to make sure the values are adjusted between 0-1
- Create a variable for cumulative sum
    - If currentCity is the 0th index i.e., 1st city, then cumulative sum equals the new updated probability of the next index
    - If currentCity is not zero, then cumulative sum equals the new updated probability of the 0th index
- Create a random value between 0 and 1 to compare with new probabilities
- For loop that runs for the size of list of probabilities, here it is 51 as we have 51 cities
    - Condition 1: index i is not the same as current city
    - Condition 2: temp does not contain index i

      These conditions stop the ant from travelling back to the visited cities hence **eliminating illegal moves**
    - IF condition 1 AND condition 2 are satisfied then CHECK if cumulativeSum < random double AND index i is not '0'. If these 2 conditions meet, we add the current index's updated probability to cumulative sum
    - ELSE add the index i to temp and BREAK out of the for loop.
    - This means that we have found a new city and added it to our single ant's path
- Increase the counter at the end of while loop
- WHILE loop will terminate when we find a path that is an arraylist of 51 single time visited cities
- Add the temporary path a ***global*** array list called '***ants***' of arraylist of integers i.e., list of the paths
- When the main for loop terminates, ***ants*** list will have number of paths equal to the number of ants in a generation

E. *updatePheromones()*

This method updates the pheromone matrix to new values because it incorporates a new matrix called Edge Matrix. It works as follows:

- ⇨ Initialize Edge Matrix with 0
- ⇨ Loop through the edge matrix and increment the values with 1 at the indices are present in an ant's path
- ⇨ This means if (1,2) is visted, (2,1) is also visited and value is incremented at both
- ⇨ Do this for all the ants
- ⇨ Update the pheromone matrix with formula

  $$(1-\rho)\tau_{ij}(t)$$

  where ρ is the evaporation rate constant
- ⇨ Update the edge matrix with formula

  $$\sum_{\substack{k\in colony \\ used\ edge}} \left[\frac{Q}{L_k}\right]$$

  Means dividing the visited edges by the total number of cities (51 in this case)
- ⇨ Add the edge matrix and updated pheromone matrix and assign those values to edge matrix

III. Experimental Setup

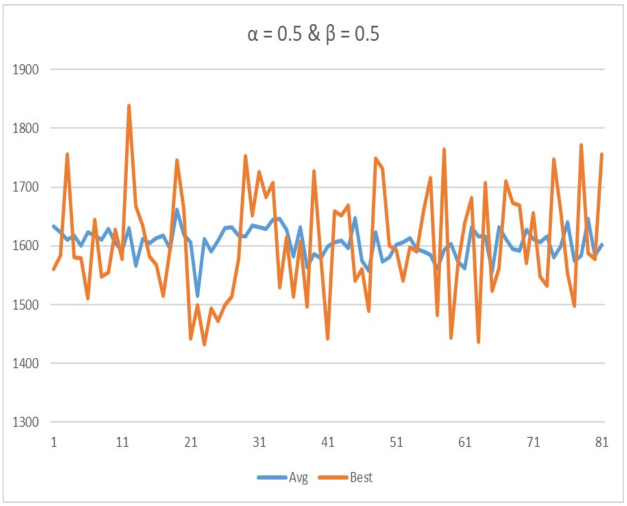
The experimental setup for my assignment contains a file aco_tsp.java which runs with some preset parameters that the user can modify. My output runs and prints given parameters: random seed, α, β, ρ, generation size, number of ants. Next, it prints the best tour length and the average tour length for each generation of ants. Lastly it prints the best tour length for the run i.e., ***best of the best tour lengths*** for all generations, along with printing its ***path***. For the weight of pheromone α, it accepts a double i.e., 90% = 0.9 and the same goes for the weight of the heuristic β. Now, I have obtained the results for all the 5 cases that I was supposed to test. For the 5th parameter's values, I chose a value that would give differentiating results. These values are α = 0.2 and β = 0.8.

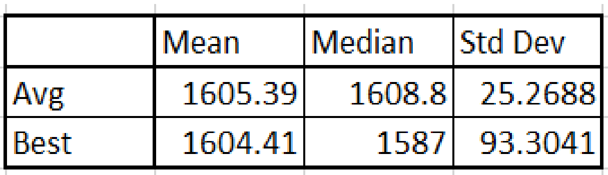
Number of ants is set to 50 as it deemed sufficient for the ACO, although it can be changed. I have set my generation span to 80 as it was good enough for testing the ACO_TSP algorithm.

IV. Results

In this section I will list down my observations from the results of my ACO_TSP using excel graphs. Note that all the observations I have noted down here are for 4 different values of α and β with random seed x. It shows the fundamental differences of using different parameters on each run.

A. α = 0.5 and β = 0.5

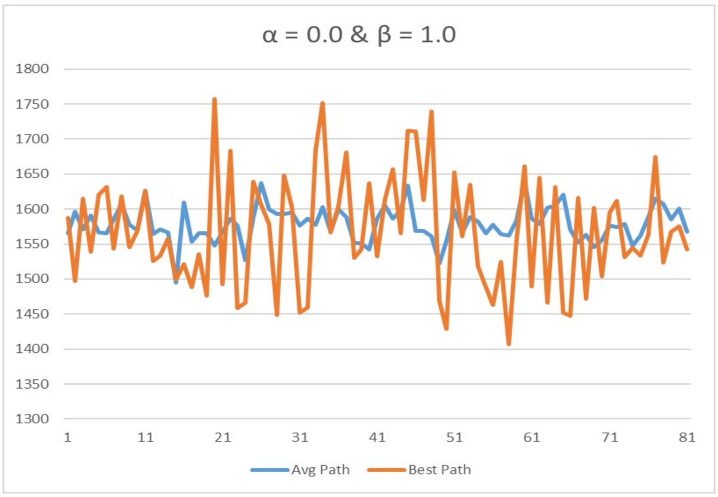


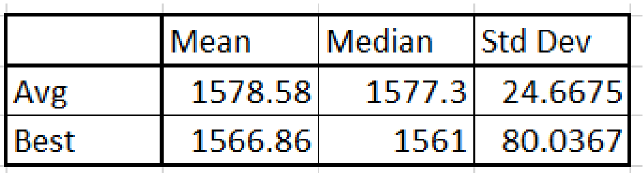


	Mean	Median	Std Dev
Avg	1605.39	1608.8	25.2688
Best	1604.41	1587	93.3041

Comparing the average paths with the best paths for this parameter set shows that the average path is more stable as it has less variance, hence the lesser standard deviation. Moreover, there is mean of the best paths is lesser than the average path, showing improvement.

B. α = 0.0 and β = 1.0

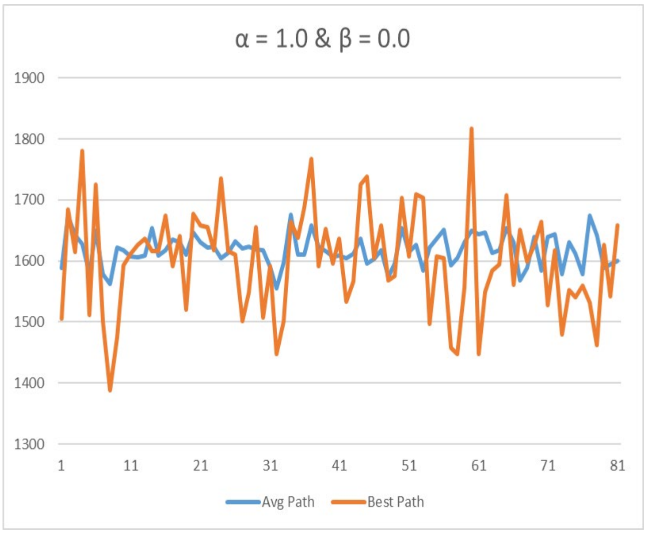




	Mean	Median	Std Dev
Avg	1578.58	1577.3	24.6675
Best	1566.86	1561	80.0367

Comparing the average paths with the best paths for this parameter set shows that the average path is more stable as it has less variance, hence the lesser standard deviation. Moreover, there is mean of the best paths is significantly lesser than the average path, showing a lot of improvement.

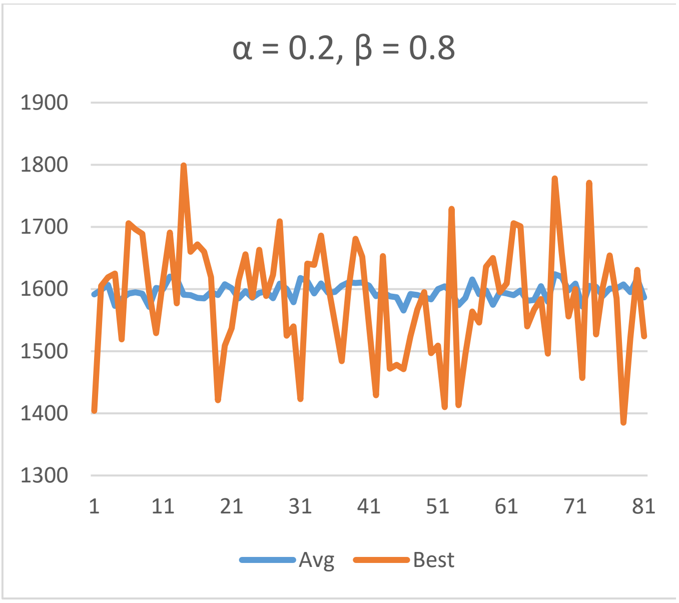
C. α = 1.0 and β = 0.0



	Mean	Median	Std Dev
Avg	1617.36	1617.7	25.6704
Best	1600.51	1608	84.6483

Comparing the average paths with the best paths for this parameter set shows that the average path is more stable as it has less variance, hence the lesser standard deviation. Moreover, there is mean of the best paths is lesser than the average path, showing improvement.

D. α = 0.2 and β = 0.8



	Mean	Median	Std Dev
Avg	1595.68	1594.88	12.2767
Best	1585.68	1595	91.1634

Comparing the average paths with the best paths for this parameter set shows that the average path is more stable as it has less variance, hence the lesser standard deviation. Moreover, there is mean of the best paths is lesser than the average path, showing improvement. Also, the path gets longer as we increase α!

E. Statistical Analysis

The statistical analysis that I have performed is by taking the best and the average tour lengths of the generations for 5 different parameters using random seed 0 and the document provided.

The analysis shows that:

- The standard deviation for best paths is higher than that of average paths. In terms of statistics, this means that there is bigger variability in the data set of the best paths found by ACO. I have emphasized this a lot when observing graphs too. Here we just get the proof of it
- Notice that the average tour length (mean) for decreases as we lower the value of α while increasing the value for β

V. Discussions and Conclusions

The results that I have shed light upon in this report are obtained by using the two important parameters: random seed 0 and eli51.txt. Therefore, there are so many possibilities that my results could vary from when changing these crucial parameters to something else. My ACO does allow the user to change these parameters and get the results desired for those parameters, however, it can differ from my analysis. However, the general observations I have made using my algorithm for the used parameters shows that it works as it is supposed to, by lowering the overall tour lengths over the generations while also showing different results depending on the pheromone weight, and heuristic weight.

User can also change the antsSize variable to whatever they want, or the genSize which changes the number of generations to run the ACO.

VI. References

All my 20 excel experiments can be found in the submission file under the folder ***Excel_Experiments.***