ANT COLONY OPTIMIZATION

FOR TSP

Assumptions

- As I was increasing n, the number of cities, I was also increasing the values that the coordinates of a city could take. Exactly [-n,n].
  I was hoping that it would make the results prettier and less cluttered

Methodology:

PFA as a separate pdf *methodology.pdf*
Default Values: n=4, l=10, alpha=1, beta=3, rho=0.8, Q=1, iterations=1000
The AntColony class accepts these values as parameters.

Algorithm:

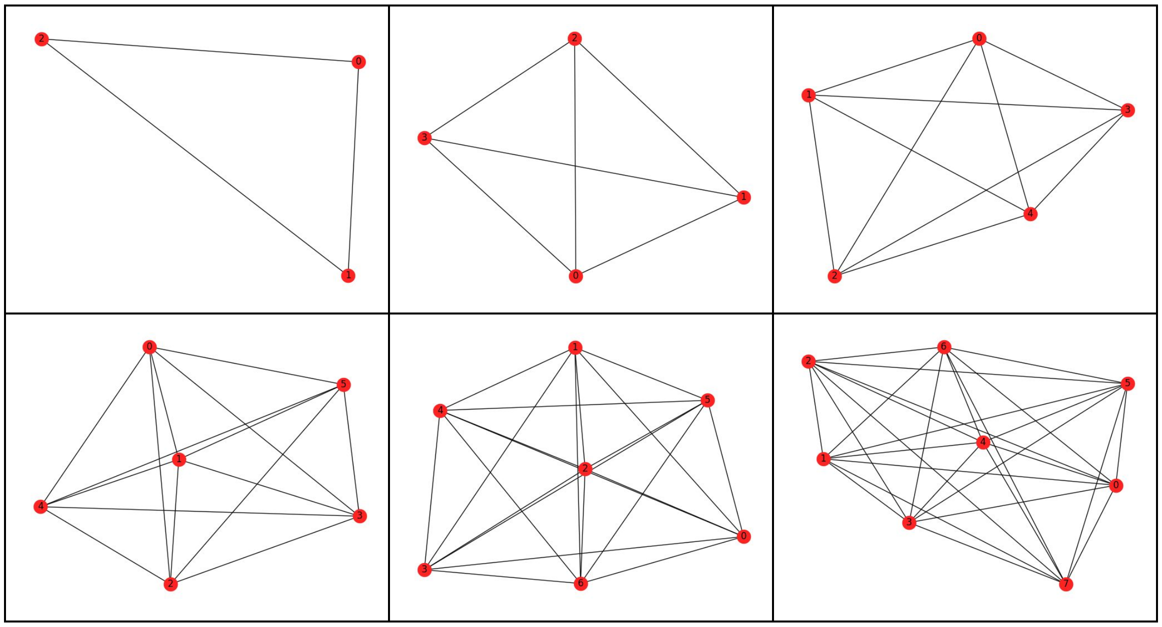
The most basic ACO algorithm was used:

```
FOR ITERATION IN RANGE(1000):
    FOR ANTS IN ALL_ANTS:
        FOR MOVE IN RANGE(N):   #N = Number of cities
            Move ant according to $P_{ij}^{k}$
        Calculate $L_k$
    Update pheromone level
```

The formulae for the pheromone update and calculation of probability of going to the next city is given in the Methodology document.

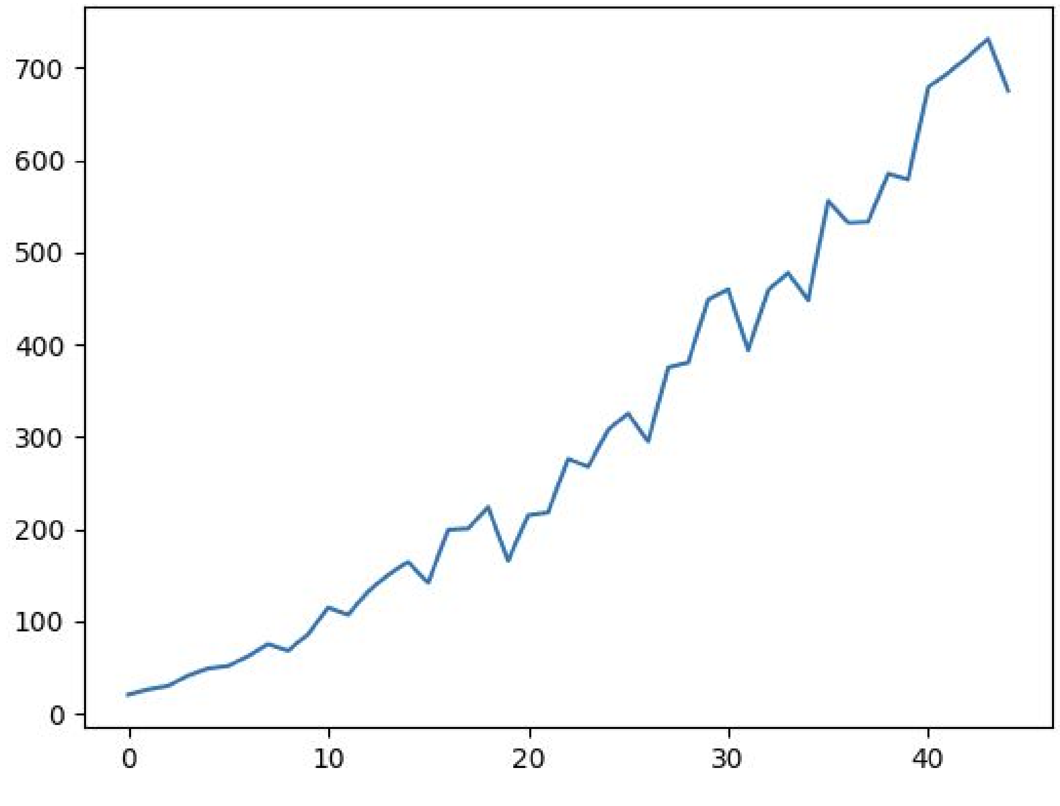
Observations and Results:

a) Visualizing the distance matrix using Graph

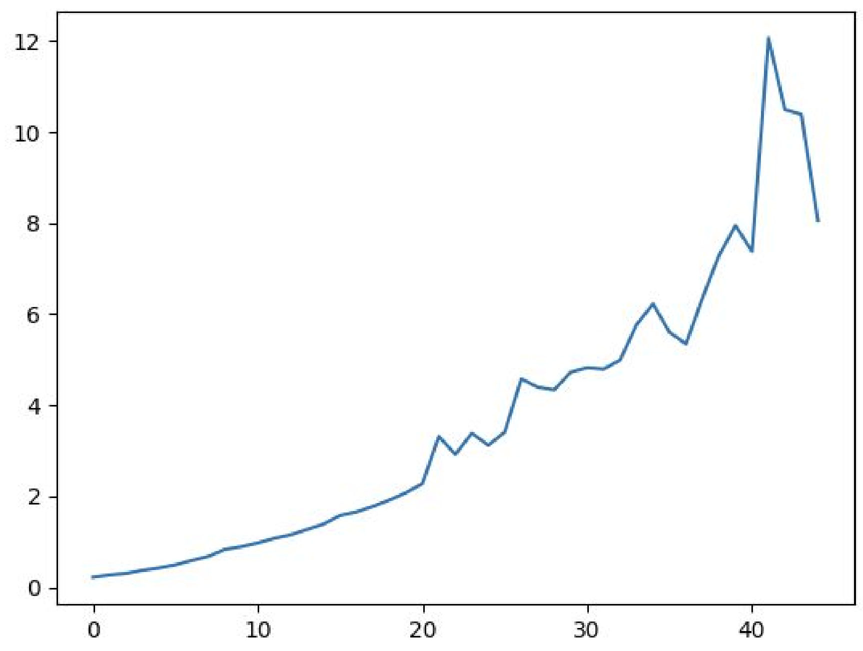


b) Visualizing the results

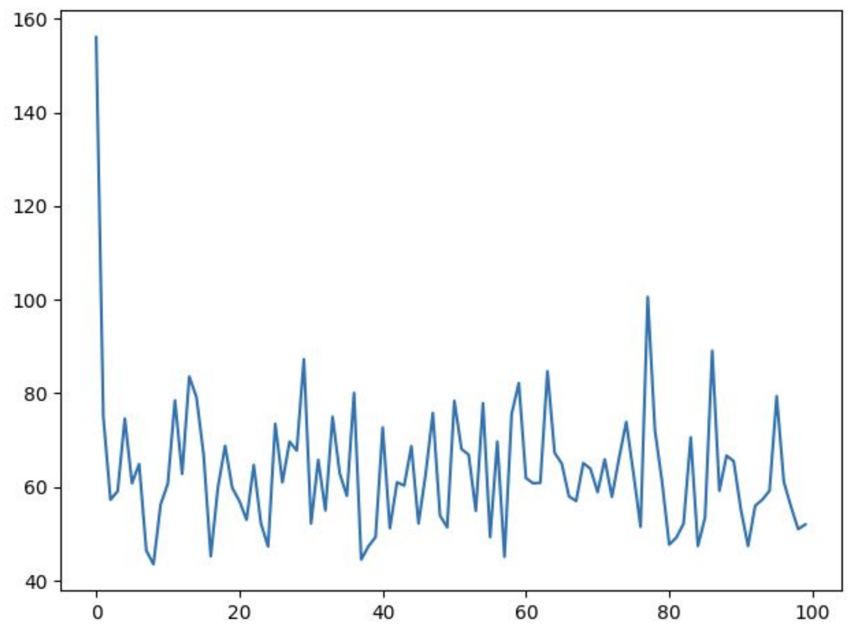
Number of cities VS Shortest Distance
PS - x axis starts from 5-50



Number of cities vs Time taken
PS - x axis starts from 5-50

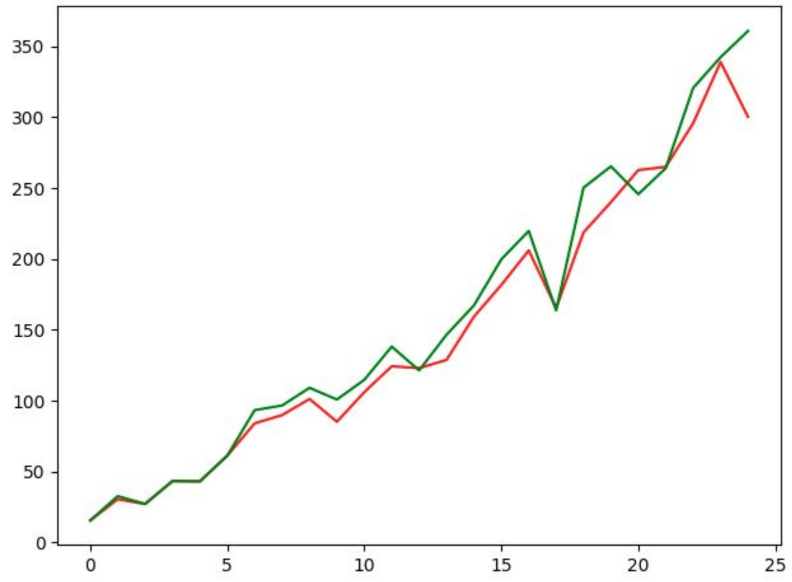


Number of iterations vs average distance travelled(n = 5)

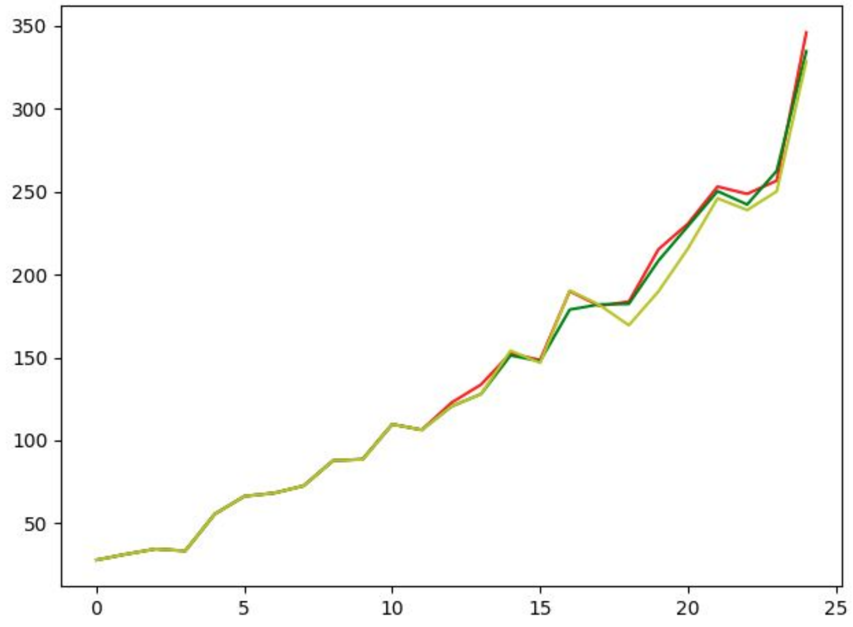


c) Compare and Contrast observations by changing parameters of ACO
All the figures below are Number of cities vs Distance
PS - x axis starts from 5-30 for all.

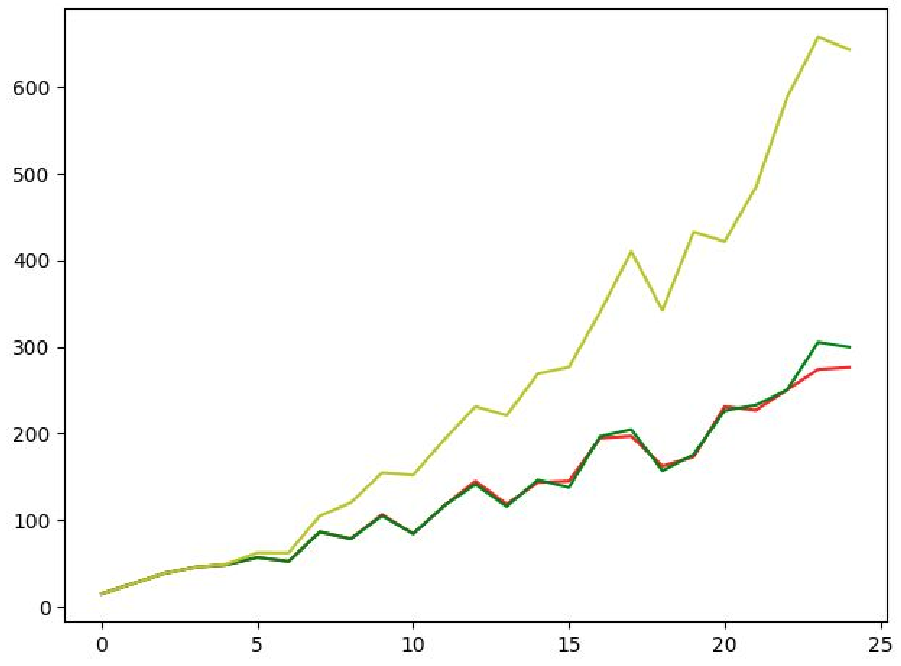
i) Rho = 0.8(Red) vs Rho = 0.001(Green) in 10 iterations



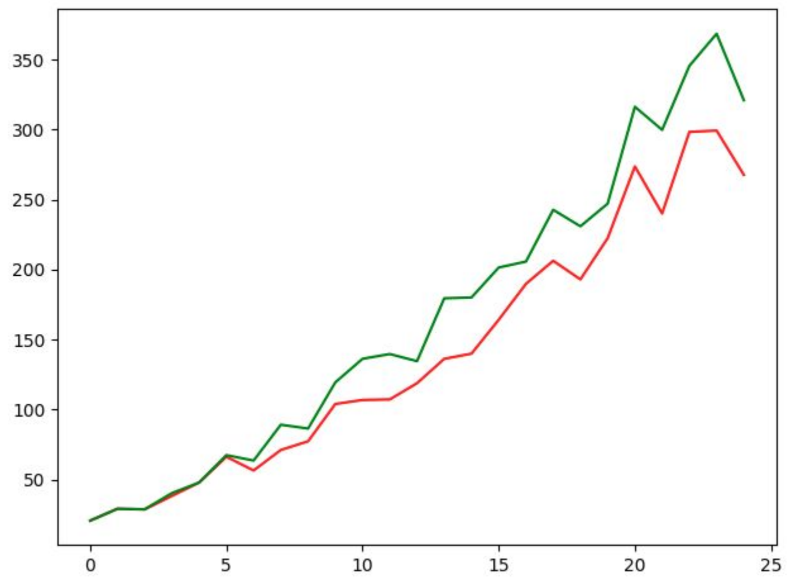
ii) Q = 1(Red), Q = 10(Green), Q = 100(Yellow)



iii) Alpha = 1, Beta = 3(Red)
Alpha = 0, Beta = 3(Green)
Alpha = 1, Beta = 0(Yellow)



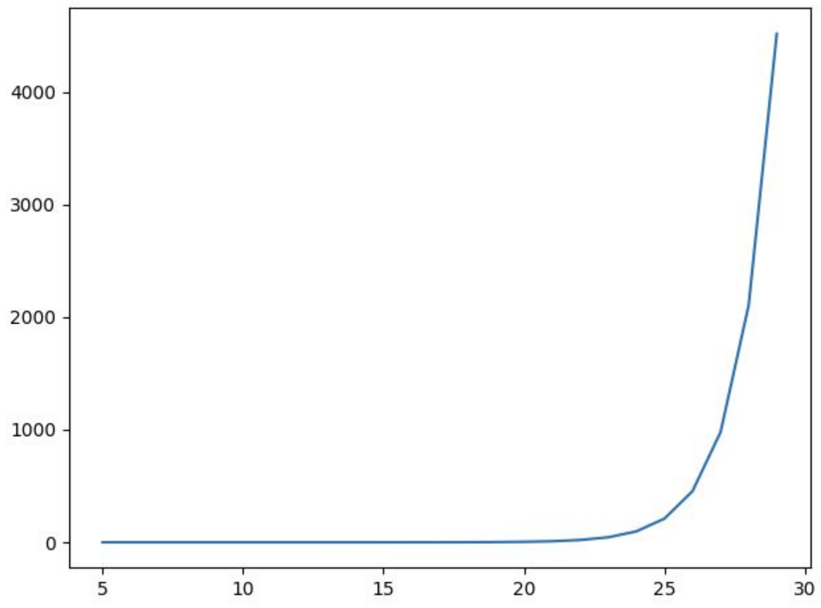
iv) Iteration = 1000(Red) vs Iterations = 100(Green)



Inferences

- Complexity is n^2*2^n. Hence, solving a 50 city puzzle should take around 11258999.0684 seconds. The ACO solves it in just 13 seconds. Even though optimality is not guaranteed in my solution, this is amazing.
  [Assuming the system performs 10^8 computations in 1 second]

Number of cities vs time taken



- Since the algorithm is always probabilistic, in order to converge it faster we can use some epsilon greedy method, where we will gradually become more and more greedy in choosing the most probable path and with probability 1-epsilon, we will pick the next city from the given distribution.
- A moderate Rho - 0.8 works better than a very low rho - (0.001)
- The value of Q doesn't really matter as the results didn't vary much