CS312 - Artificial Intelligence Lab Assignment 4

Arun R Bhat (190010007), Mohammad Sameer (190010024)

1 Introduction

We were given the task of solving famous *Travelling Salesman Problem* in as optimal way as possible, i.e. - We had to find the best possible tour involving all the given cities exactly once such that total distance travelled is close to minimum.

2 Optimization

We tried implementing all three algorithms we learnt - *Genetic algorithm, Simulated Annealing and Ant-Colony Optimization. Simulated annealing* was the simplest to implement and produced the worst result among them. *Genetic algorithm* gave better results but not as good as *ACO*. We can reason that genetic algorithms are more random in nature and hence can take too long to get near optimum. On the other hand, *ant colony optimization* uses pheromone as a measure which is very smart and hence arguably gives the best results for *travelling salesman problem.*

3 Pseudo-Code

The pseudo code of the Ant Colony Optimization is given below:

```
def ACO(cityDistances, alpha, beta, rho, Q, N, m, numItr):
    #N = number of cities
    #m = number of ants
    #numItr = number of iterations

    ants = [ant * m]     #set of all ants
    for _ in range(numItr):
        for k = 1 to m ants:
            for i = 1 to N:
                move ant[k] from city i to j based on probability $P_{ij}^k$
            $L_k$ = findTourCost(ant[k])
        updateBestSolution()
        updatePheromoneContent()
```

4 Stats

For the final version of our code, we have obtained the tour best as mentioned below:

Input File	**Tour Cost**
euc_100	1586
noneuc_100	5237

5 Working and Conclusion

We shuffled the cities in the beginning to start with in every iteration, and then moved in the direction of higher pheromone concentration. Parameters used were as follows:

- α : gives concentration of pheromone as a probability measure.
- β : gives distance between cities as a factor in calculating probability.
- Q : updates pheromone in every iteration in order to induce slight randomness.
- ρ : the evaporation constant that reduces over-dependence on pheromones and helps in not getting stuck at local optima.

In our case the number of ants were equal to number of cities. This made the algorithm run slower, so we decided to pick only a few best of the ants, meaning the ants that had smaller tour cost. To pick the number of ants we have used a factor named *convergenceRate*. In our trial and run, we found out that this number when equal to $0.1 * \{number\,of\,cities\}$ yielded best results.

After a lot of trial, error and fine tuning, we have found the parameters to give a good result in general:

- $\alpha = 10$
- $\beta = 10$
- $Q = 0.5$
- $\rho = 0.5$
- $convergenceRate = 0.1 * \{number\,of\,cities\}$