TRAVELING SALESMAN PROBLEM WITH ANT COLONY OPTIMIZATION

BY,

HARSHIL KHARA BU ID – **U07281822** Email- **harshilk@bu.edu**

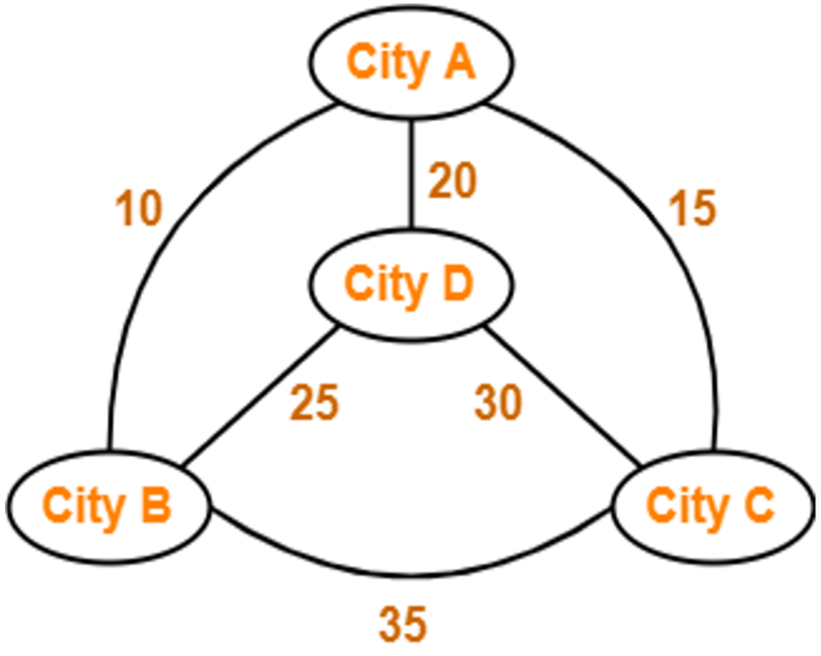
NYRIKA BHARGAVARAM RENUKA BU ID – **U60948012** Email- **nyrikabr@bu.edu**

PRANJAL PATEL BU ID – **U32773567** Email- **pranjalp@bu.edu**

PROBLEM STATEMENT

Traveling salesman problem asks the following question-

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city **exactly once and returns to the origin city**?



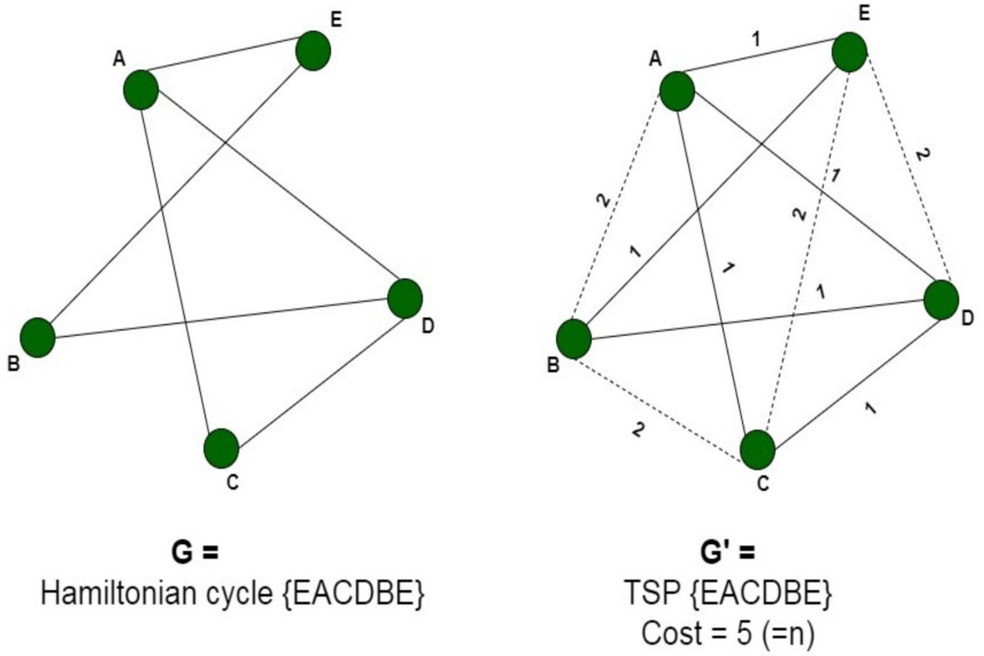
Travelling Salesman Problem

NP HARD PROBLEM

Traveling salesman problem is a benchmark for **NP (Non-Deterministic Polynomial) Hard problem.**

<u>Why is it a NP-Hard Problem?</u>

Since it is not in NP problem, it cannot be in NP-complete.
It takes exponential time to solve NP, the solution cannot be checked in polynomial time. Thus, this problem is NP-Hard but not in NP.
Also, it can be reduced to a known NP-Hard problem that is Hamilton Cycle Problem.



Therefore, all the approaches to this unique problem are not absolutely accurate.

VARIOUS APPROACHES USED

There are various approaches used to solve this problem. Some of the approaches are-

- Naive and Dynamic Programming
- Using approximate Minimum Spanning Tree
- Genetic Algorithm
- Greedy Algorithm
- Ant Colony Optimization

We are going to use the **Ant Colony Optimization** approach.

ANT COLONY OPTIMIZATION

Originally proposed in 1992 by Marco Dorigo, ant colony optimization (ACO) is an optimization technique inspired by the path finding behaviour of ants searching for food.

ACO is also a subset of swarm intelligence - a problem solving technique using decentralized, collective behaviour, to derive artificial intelligence.

Typical applications of ant colony optimization are combinatorial optimization problems such as the traveling salesman problem, however it can also be used to solve various scheduling and routing problems.

One advantage ant colony optimization algorithms have over other optimization algorithms is their ability to adapt to dynamic environments - a feature that makes it great for applications such as network routing, where there are likely to be frequent changes to accommodate to.

ANTS IN NATURE

To fully understand ant colony optimization, it's important to first understand the natural behaviour of ants which first inspired the algorithm.

When searching for food ants follow a relatively basic set of rules.

Although simple, it's these rules which allow ants to communicate and cooperatively optimize their paths to food sources.

One of the key characteristics behind all these rules is the use of pheromone trails. Pheromone trails are essentially what ants use to communicate to other ants that a food source has been found, and how to get to it.

When other ants come across pheromone trails they can typically expect to find food if they decide to follow it.

However, ants don't follow every pheromone trail they find.

Depending on the strength of the pheromone trail, an ant may decide to take a different path, or perhaps a completely random path which has no pheromone on.

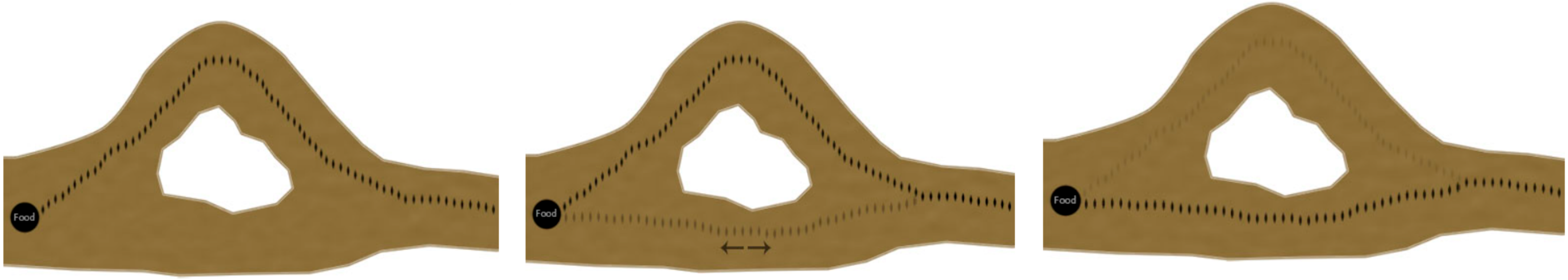
However, on average the stronger a pheromone trail is, the more chance there is of an ant taking it.

Over time, unless reinforced by other ants, pheromone trails will gradually evaporate.

This means that pheromone trails which no longer lead to a food source will eventually stop being used, promoting ants to find new paths and new food sources.

EXAMPLE

To understand how this process over time also enables the colony to optimize their paths, consider the following example:



THE ALGORITHM

The bulk of the ant colony optimization algorithm is made up of only a few steps.

First, each ant in the colony constructs a solution based on previously deposited pheromone trails.

Next ants will lay pheromone trails on the components of their chosen solution, depending on the solution's quality.

In the example of the traveling salesman problem this would be the edges (or the paths between the cities).

Finally, after all ants have finished constructing a solution and laying their pheromone trails, pheromone is evaporated from each component depending on the pheromone evaporation rate.

These steps are then ran as many times as are needed to generate an adequate solution.

CONSTRUCTING A SOLUTION

As mentioned previously, an ant will often follow the strongest pheromone trail when constructing a solution.

However, for the ant to consider solutions other than the current best, a small amount of randomness is required in its decision process.

In addition to this, a heuristic value is also computed and considered helping to guide the search process towards the best solutions.

In the example of the traveling salesman problem this heuristic will typically be the length of the edge between the city being considered - the shorter the edge, the more likely an ant will pick it.

For the traveling salesman problem a state would represent a single city on the graph. Here, an ant would be selecting the next city depending on the distance to the next city, and the amount of pheromone on the path between the two cities.

EQUATION

Let’s take a look at how this works mathematically:

$$p_{ij}^{k} = \frac{[t_{ij}]^{\alpha} \cdot [n_{ij}]^{\beta}}{\sum_{l \in N_i^k} [t_{il}]^{\alpha} \cdot [n_{il}]^{\beta}}$$

This equation calculates the probability of selecting a single component of the solution. Here, t_{ij} denotes the amount of pheromone on a component between states i and j, and n_{ij} denotes it’s heuristic value. α and β are both parameters used to control the importance of the pheromone trail and heuristic information during component selection.

LOCAL PHEROMONE UPDATE

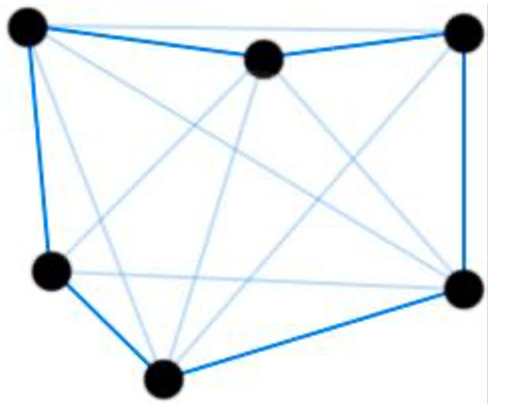
The local pheromone update process is applied every time an ant successfully constructs a solution.

This step mimics the way ants lay pheromone trails after finding food in nature.

As you may remember from earlier, in nature better paths acquire more pheromone due to ants being able to traverse them quicker.

In ACO, this characteristic is replicated by varying the amount of pheromone deposited on a component by considering how well the completed solution scores.

If we use the traveling salesman problem as an example, pheromone will be deposited on the paths an ant took between cities depending the total tour distance.



This image shows pheromone trails laid on the paths between cities in a typical traveling salesman problem.

OPTIMIZATION

There are many variations of ant colony optimization, two of the main ones being elitist and MaxMin.

Depending on the problem it's likely these variations will perform better than the standard ant colony system we have looked at here.

ELITIST ALGORITHM

In elitist ACO systems, either the best current, or global best ant, deposits extra pheromone during it's local pheromone update procedure. This encourages the colony to refine it's search around solutions which have a track record of being high quality. If all goes well, this should result in better search performance.

The math for the elitist local pheromone update procedure is almost exactly the same as the the local update procedure which we previously studied, with one small update:

$$\Delta\tau_{ij}^{bs} = \begin{cases} Q/C^{bs} & \text{if component}(i,j) \text{ belongs to an elite ant} \\ 0 & \text{Otherwise} \end{cases}$$

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs}$$

Where, e is a parameter used to adjust the amount of extra pheromone given to the best (or "elite") solution.

MAXMIN ALGORITHM

The MaxMin algorithm is similar to the elitist ACO algorithm in that it gives preference to high ranking solutions.

However, in MaxMin instead of simply giving extra weight to elite solutions, only the best current, or best global solution, is allowed to deposit a pheromone trail.

Additionally, MaxMin requires pheromone trails are keep between a maximum and minimum value.

The idea being that having a limited range between the amount of pheromone found on trails, premature convergence around sub-optimal solutions can be avoided.

In many MaxMin implementations the best current ant is initially the ant which lays pheromone trails, then later, the algorithm switches so that the global best ant is the only ant which can lay a pheromone trail.

This process helps encourage search across the entire search space initially, before eventually focusing in on the all time best, hopefully making a few final amendments.

Finally, the maximum and minimum value can either be passed in as parameters or adaptively set with code.

EQUATION

The pheromone update is implemented as follows:

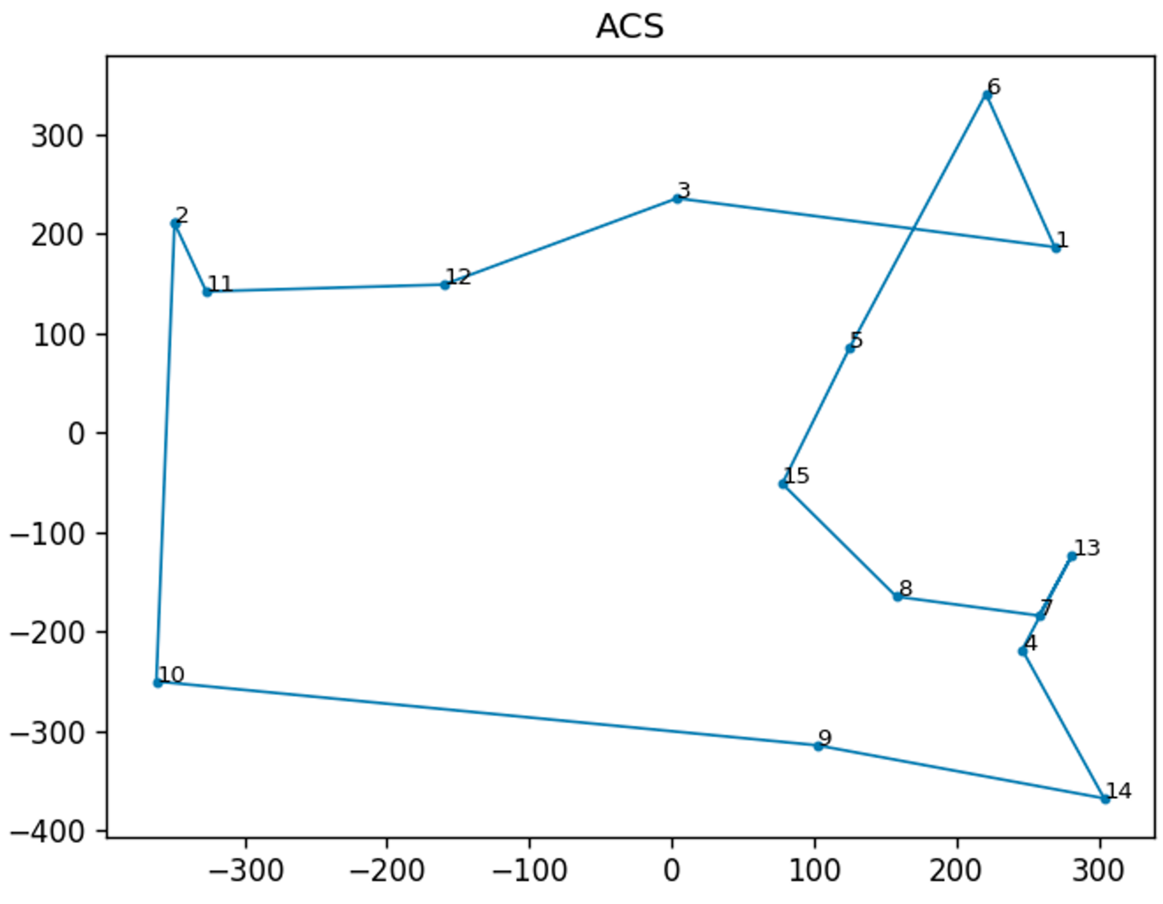
$$\tau_{ij} \leftarrow \left[(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{\text{best}}\right]_{\tau_{\min}}^{\tau_{\max}}$$

The operator [x] a b is defined as:

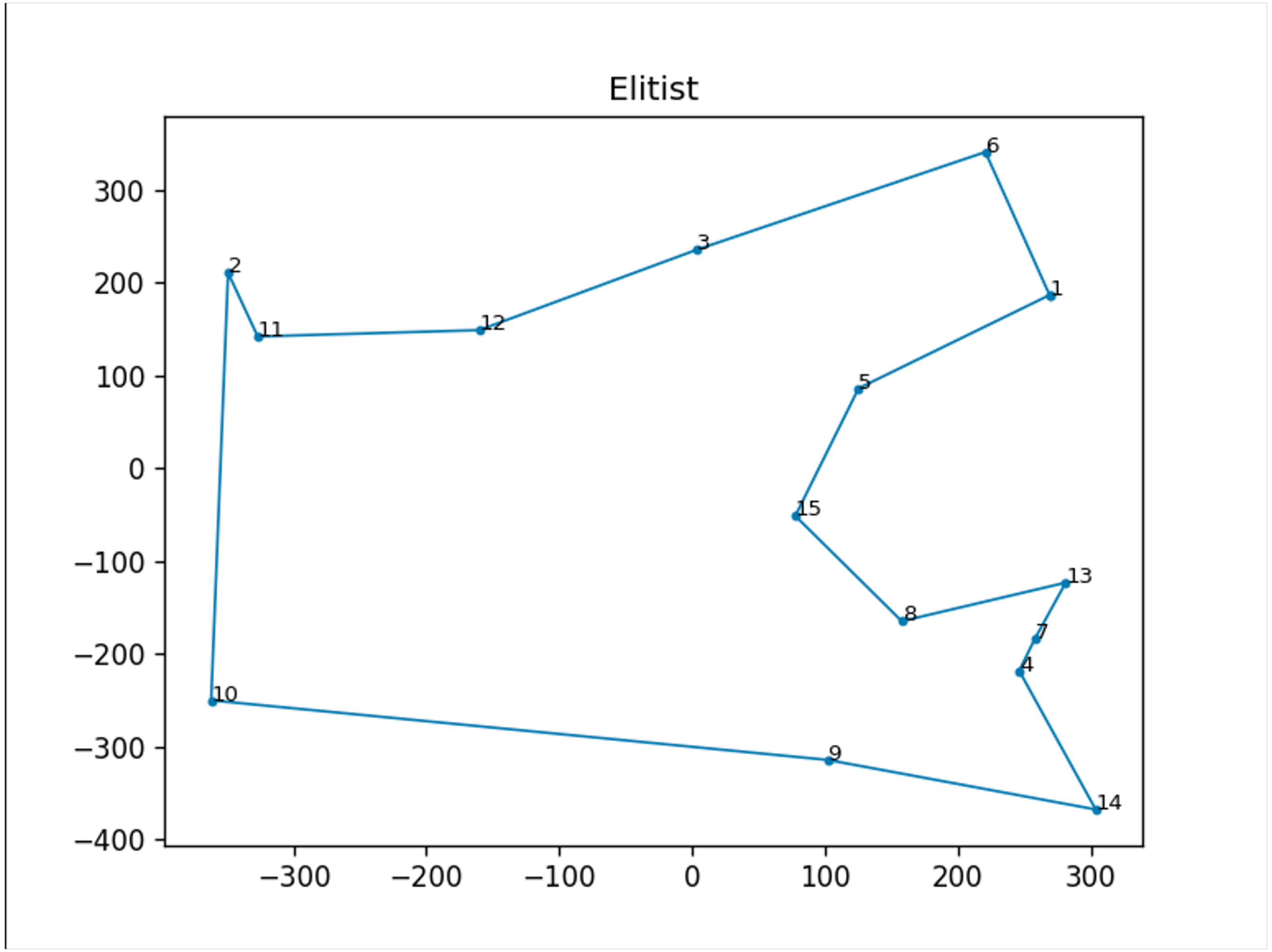
$$[x]_b^a = \begin{cases} a & \text{if } x > a, \\ b & \text{if } x < b, \\ x & \text{otherwise;} \end{cases}$$

$$\Delta\tau_{ij}^{\text{best}} = \begin{cases} 1/L_{\text{best}} & \text{if } (i, j) \text{ belongs to the best tour,} \\ 0 & \text{otherwise,} \end{cases}$$

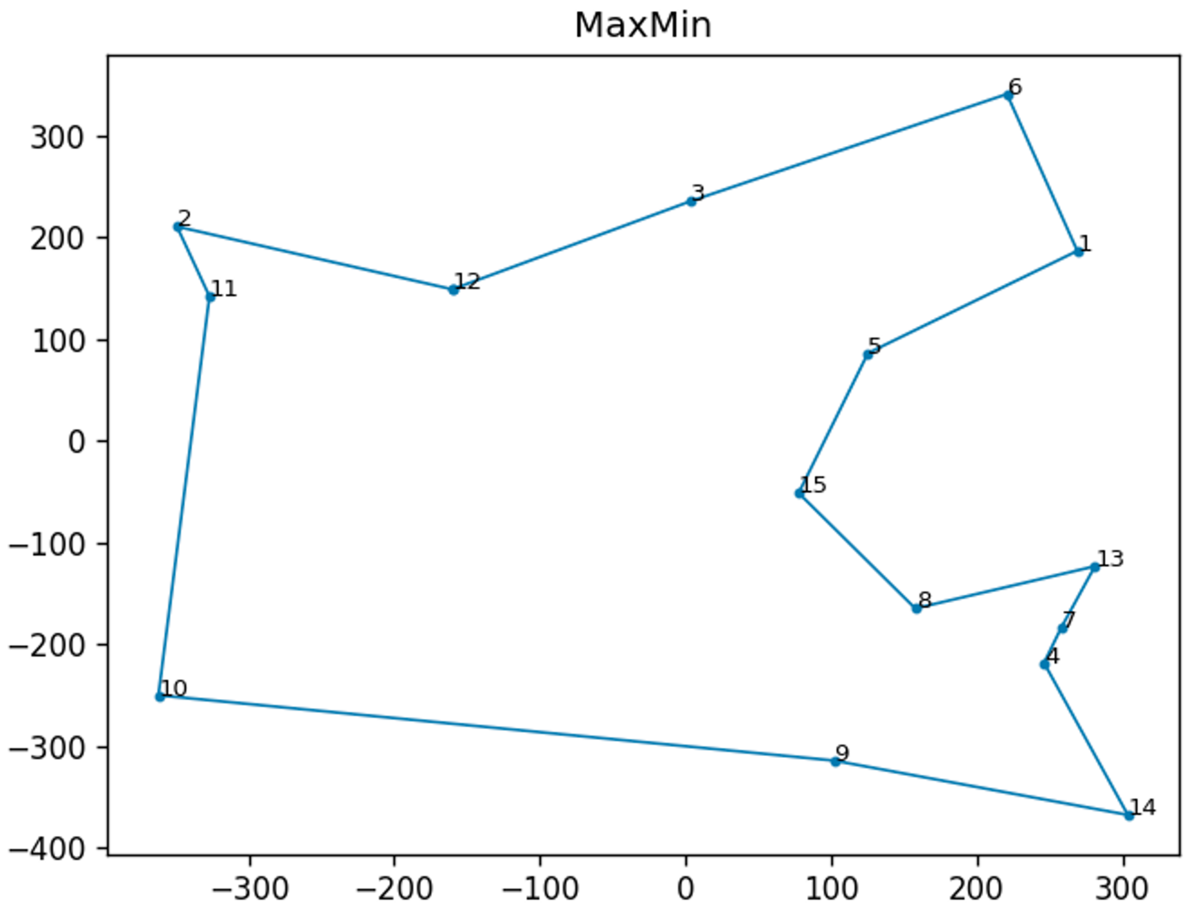
NETWORK WITH ACO



NETWORK WITH ELITIST ACO



NETWORK WITH MINIMAX ACO



NETWORK STIMULATION

Output

```
Started : ACS
Ended : ACS
Sequence : <- 2 - 11 - 12 - 3 - 1 - 6 - 5 - 15 - 8 - 7 - 13 - 4 - 14 - 9 - 10 ->
Total distance travelled to complete the tour : 2977.62

Started : Elitist
Ended : Elitist
Sequence : <- 1 - 6 - 3 - 12 - 11 - 2 - 10 - 9 - 14 - 4 - 7 - 13 - 8 - 15 - 5 ->
Total distance travelled to complete the tour : 2815.67

Started : MaxMin
Ended : MaxMin
Sequence : <- 12 - 2 - 11 - 10 - 9 - 14 - 4 - 7 - 13 - 8 - 15 - 5 - 1 - 6 - 3 ->
Total distance travelled to complete the tour : 2780.43
```

Thus, we can see the most optimal (shortest) distance is found with MaxMin.

TIME COMPLEXITY

The approximate time complexity for our Ant Colony Optimization for n evaluations is $O\left(\frac{m \Delta l \log(\Delta l)}{\rho}\right)$

Where,

m is number of edges

l is maximum number of edges on any shortest path

ρ is the evaporation factor of pheromones ($0 < \rho < 1$)

APPLICATIONS

In computer science, it is used to find the most efficient route for data to travel between various nodes.

In Biology, it is used for DNA Sequencing.

It is used in Vehicle Routing Problems.

Planning & Scheduling Problems.

Thank you.