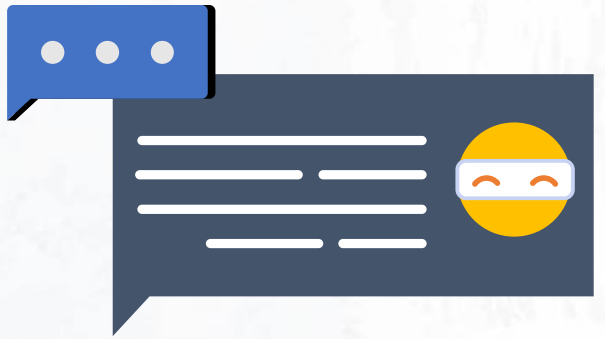
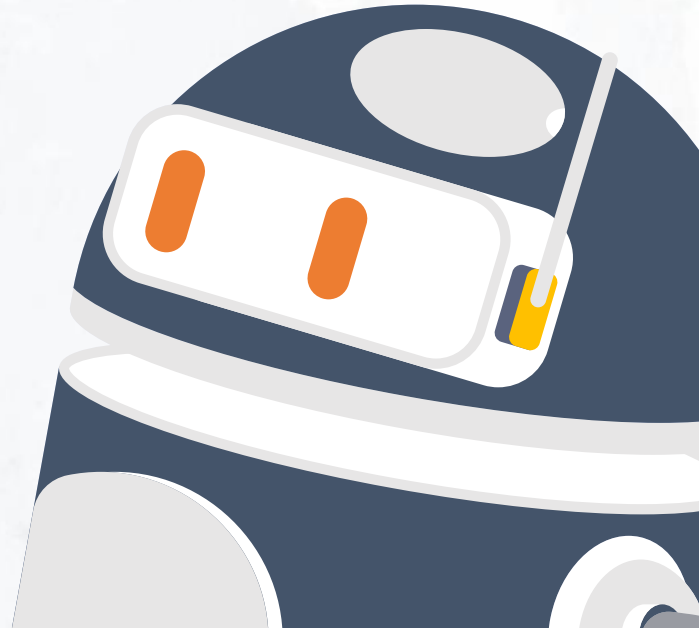
Ant Colony Optimization (ACO) for the Traveling Salesman Problem (TSP)





Name : **Mostafa Gamil**

(AI)

Table of contents

01

Algorithm Overview

(AI)

(ACO) = ⟶

Ant Colony Optimization

Ant Colony Optimization

- ACO is a population-based algorithm that uses a set of artificial ants to explore and exploit the search space.
- The algorithm iteratively constructs solutions by moving ants on the graph representation of the TSP.
- Ants deposit pheromone trails on edges, representing the attractiveness of each edge based on the quality of the solutions found.

02

ACO Components

(AI)

ACO Components

(a) Graph Representation ⟶

Cities are nodes, distances are edges.

(b) Ant ⟶

Represents a potential solution.

Constructs a tour by probabilistically selecting the next city based on pheromone and heuristic information.

ACO Components

(c) Pheromone ⟶

Trails on edges indicate path desirability.

Ants deposit pheromone based on solution quality.

(d) Heuristic Information ⟶

Ants consider distances between cities when selecting the next city.

(e) Local Updates ⟶

After each ant constructs a tour, pheromone trails on visited edges are updated.

03 →

ACO Variants

(AI)

ACO Variants

(a) Ant Colony System (ACS) ⟶

ACS is a variant of ACO that emphasizes the exploitation of good solutions.
It uses a local pheromone update rule and incorporates an elitist strategy to exploit the best solution found so far.

(b) Elitist Ant System ⟶

Elitist Ant System focuses on the intensification of the search process.

It uses a global pheromone update rule that reinforces the edges of the best solution found by each iteration.

(c) Max-Min Ant System (MMAS) ⟶

MMAS is another variant of ACO that balances between exploration and exploitation.

It dynamically regulates the pheromone levels to ensure diversity during the early iterations and intensification later on.

04 →

Implementation Details

(AI)

Advantages of artificial intelligence

The presented code demonstrates the implementation of the ACO algorithm for the TSP.
It includes classes for representing the TSP graph, ants, and various ACO variants.
The code initializes a set of random cities, creates ant colonies, and runs the ACO algorithm.
After running, the algorithm prints the best tour found and its total distance.
Additionally, it provides a visualization of the best tour using matplotlib.

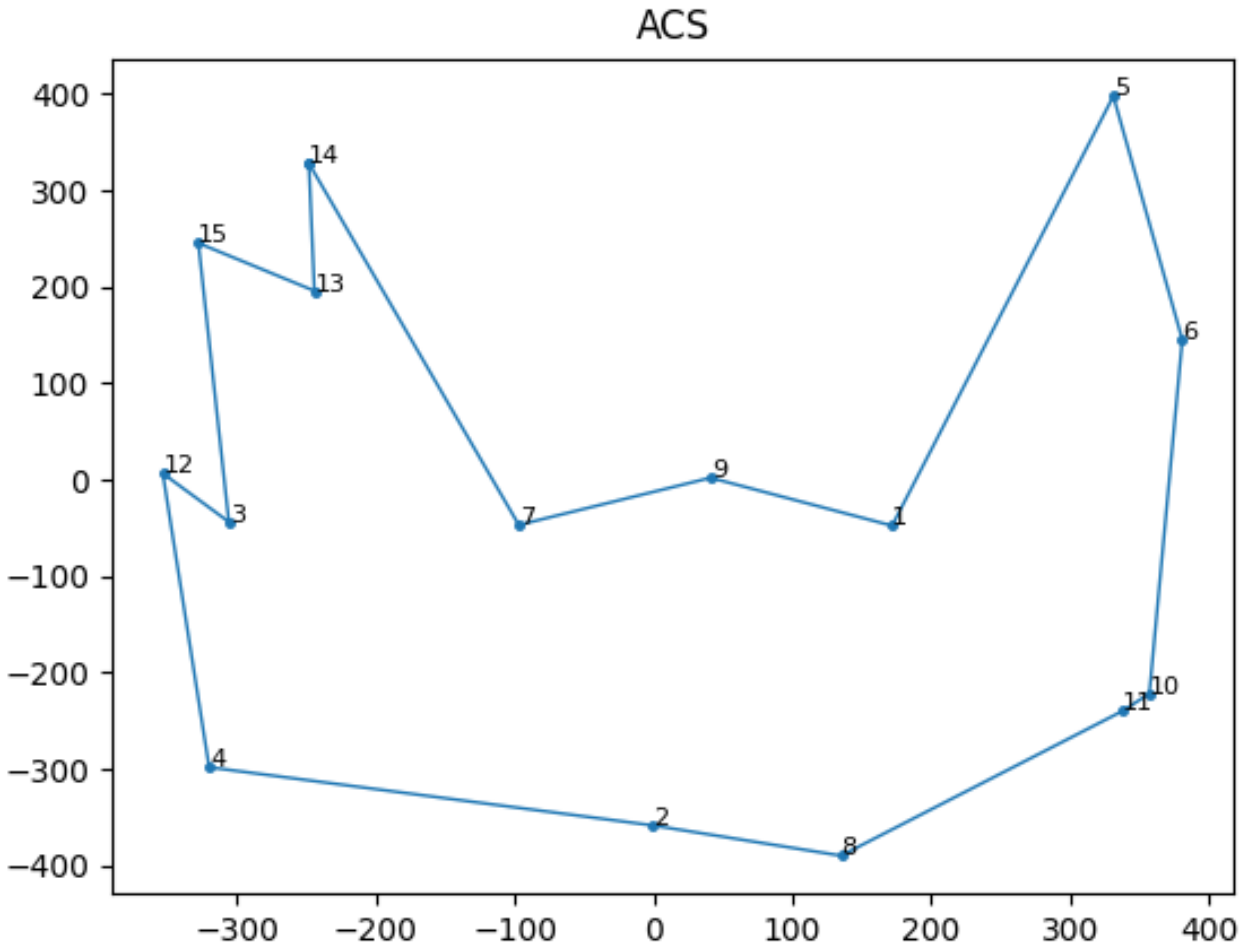
05

Results

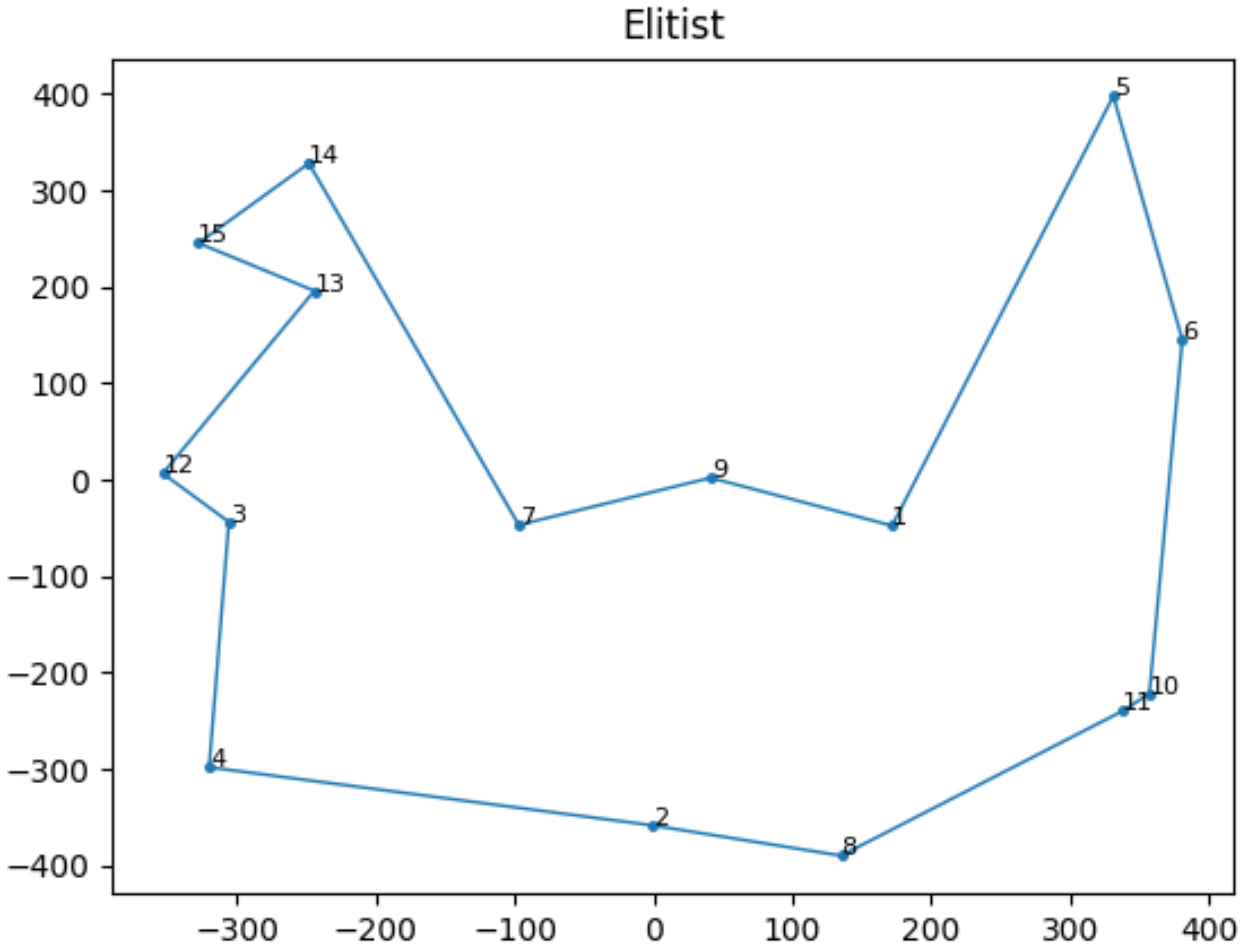
(AI)

Results

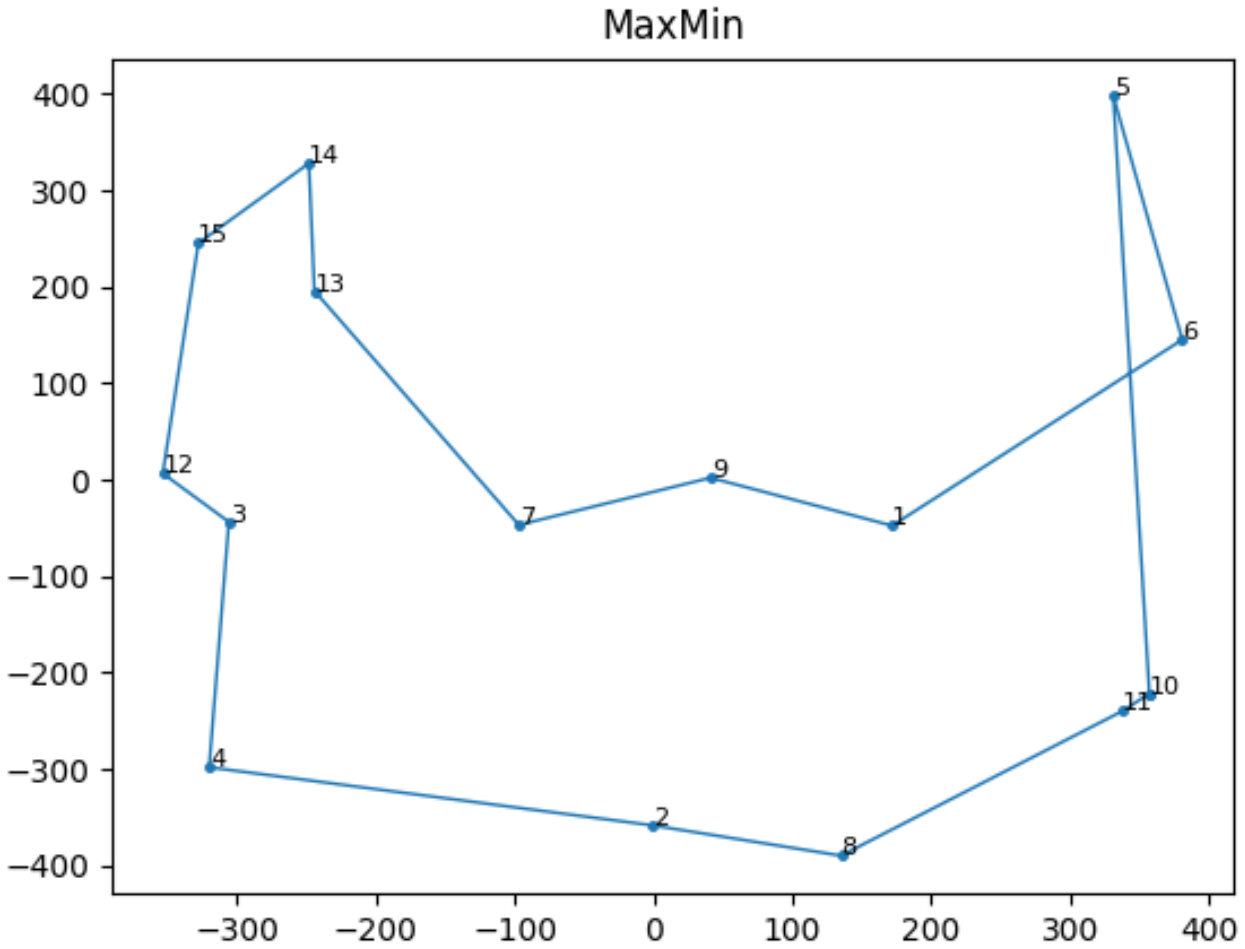
The ACO algorithm explores the search space by iteratively constructing solutions using artificial ants.
The algorithm balances exploration and exploitation to find good solutions to the TSP.
The quality of the solutions improves over iterations, and the algorithm converges to a near-optimal solution.
The visualization shows the best tour found by the algorithm, representing the shortest path to visit all cities.



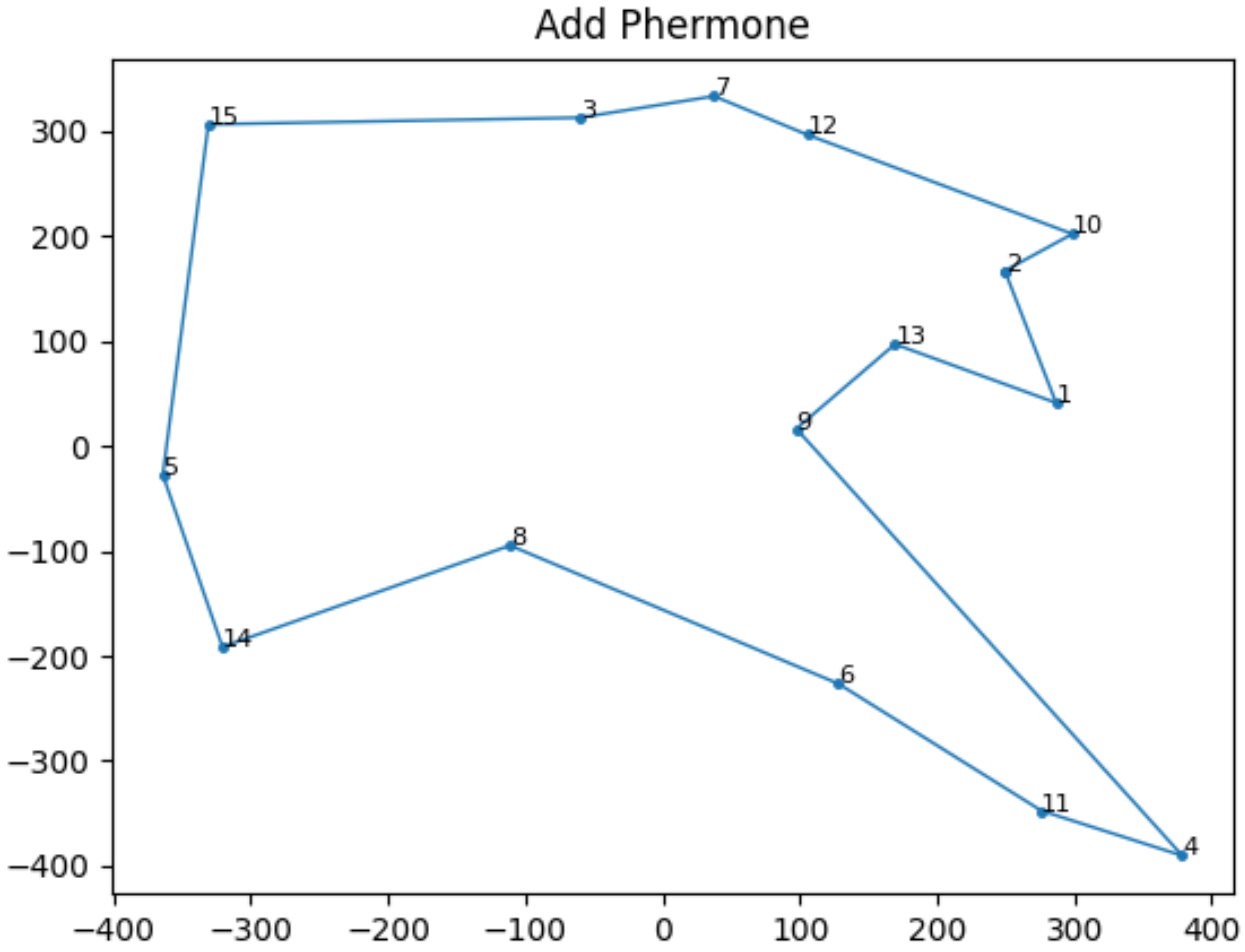
Started : ACS
Ended : ACS
Sequence : <- 14 - 13 - 15 - 3 - 12 - 4 - 2 - 8 - 11 - 10 - 6 - 5 - 1 - 9 - 7 ->
Total distance travelled to complete the tour : 3425.49



Started : Elitist
Ended : Elitist
Sequence : <- 14 - 15 - 13 - 12 - 3 - 4 - 2 - 8 - 11 - 10 - 6 - 5 - 1 - 9 - 7 ->
Total distance travelled to complete the tour : 3284.89



Started : MaxMin
Ended : MaxMin
Sequence : <- 10 - 11 - 8 - 2 - 4 - 3 - 12 - 15 - 14 - 13 - 7 - 9 - 1 - 6 - 5 ->
Total distance travelled to complete the tour : 3284.99



Started : Add Phermone
Ended : Add Phermone
Sequence : <- 2 - 10 - 12 - 7 - 3 - 15 - 5 - 14 - 8 - 6 - 11 - 4 - 9 - 13 - 1 ->
Total distance travelled to complete the tour : 2898.22

Thanks!

Any questions?

linkedin.com/in/mostafa-gamiil

+20 100 194 5639