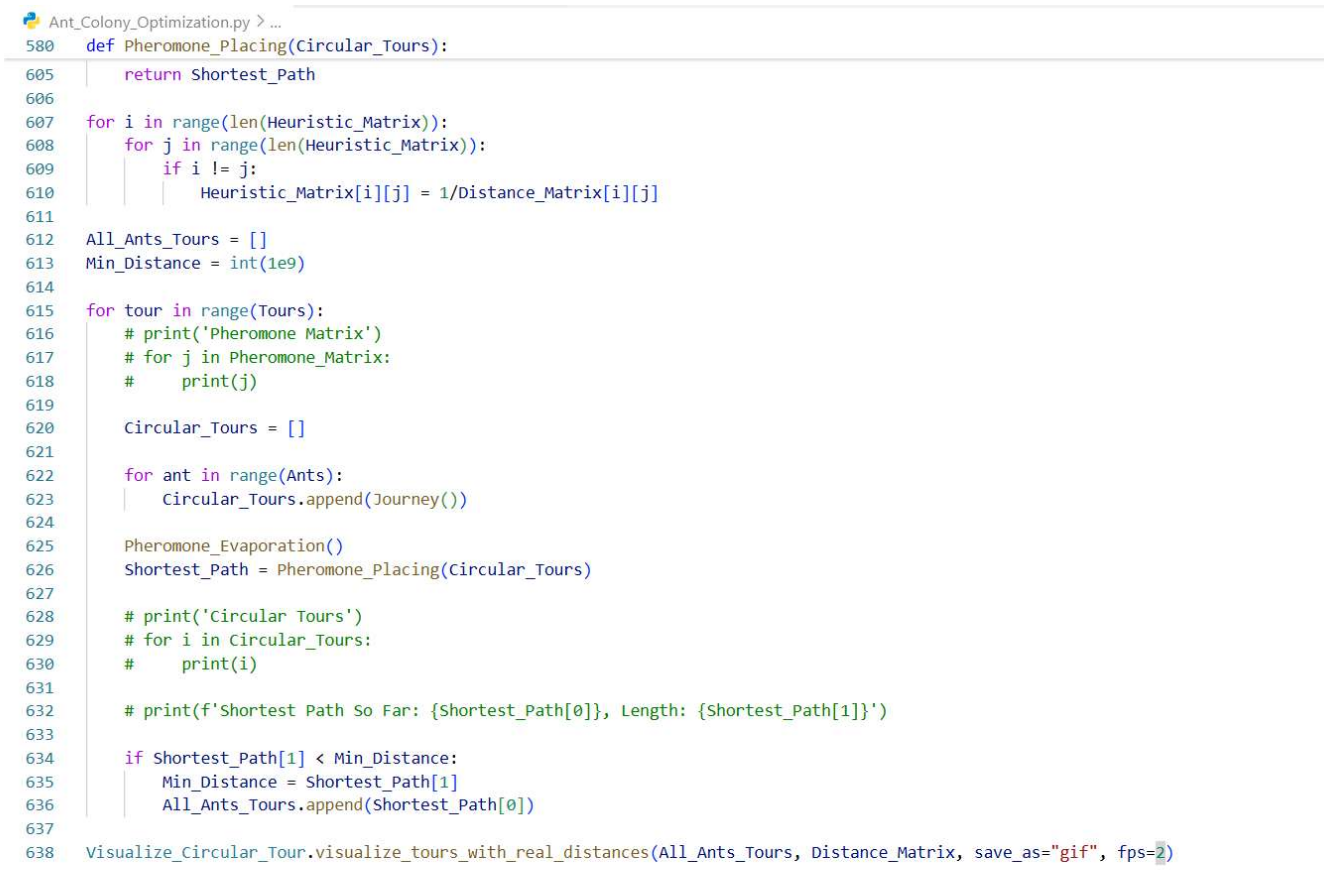
Experiment No: 11

Objective: Solve the Traveling Salesman Problem with Ant Colony Optimization Algorithm.

Theory: The Traveling Salesman Problem (TSP) aims to find the shortest route that visits all cities once and returns to the start.
The Ant Colony Optimization (ACO) algorithm solves it by mimicking how real ants find food. Ants leave pheromone trails on paths, and shorter routes get more pheromones. Over time, ants follow these stronger trails, helping the algorithm discover the shortest path efficiently.

Source Code:



Ant_Colony_Optimization.py > ...

```python
def Pheromone_Placing(Circular_Tours):
    return Shortest_Path

for i in range(len(Heuristic_Matrix)):
    for j in range(len(Heuristic_Matrix)):
        if i != j:
            Heuristic_Matrix[i][j] = 1/Distance_Matrix[i][j]

All_Ants_Tours = []
Min_Distance = int(1e9)

for tour in range(Tours):
    # print('Pheromone Matrix')
    # for j in Pheromone_Matrix:
    #     print(j)

    Circular_Tours = []

    for ant in range(Ants):
        Circular_Tours.append(Journey())

    Pheromone_Evaporation()
    Shortest_Path = Pheromone_Placing(Circular_Tours)

    # print('Circular Tours')
    # for i in Circular_Tours:
    #     print(i)

    # print(f'Shortest Path So Far: {Shortest_Path[0]}, Length: {Shortest_Path[1]}')

    if Shortest_Path[1] < Min_Distance:
        Min_Distance = Shortest_Path[1]
        All_Ants_Tours.append(Shortest_Path[0])

Visualize_Circular_Tour.visualize_tours_with_real_distances(All_Ants_Tours, Distance_Matrix, save_as="gif", fps=2)
```

Source Code On GitHub:
https://github.com/MdImranKhanSiam/Problem_Solving/tree/main/Practice/All_Program/Python/Ant_Colony_Optimization

Algorithm:

1. Initialize the distance matrix with distances between all cities.
2. Initialize the pheromone matrix with small starting values for all paths.
3. Set algorithm parameters: Alpha, Beta, Rho, Pheromone Level, number of ants, numbers of iterations.
4. Repeat for each iteration:

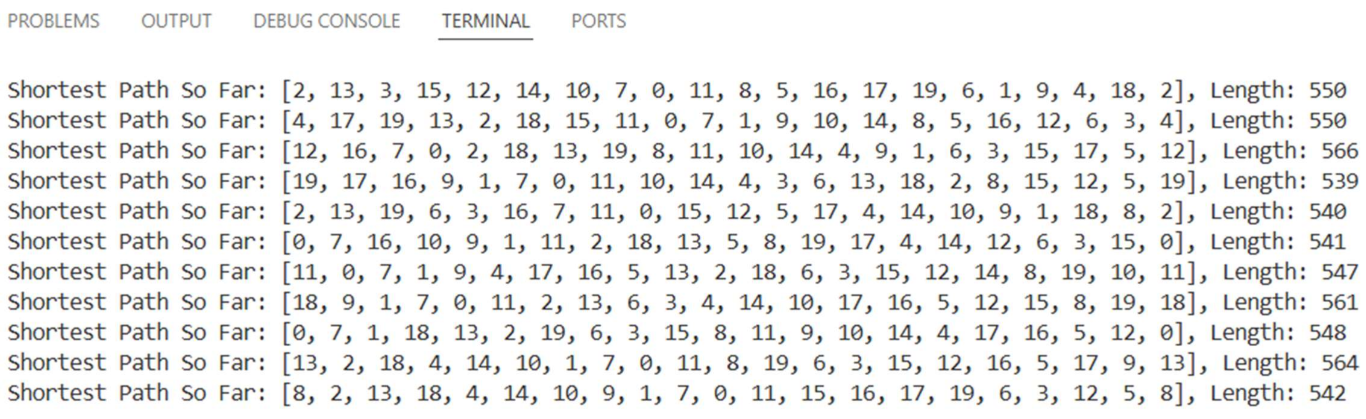
   a. Place ants on starting cities.

   b. Each ant constructs a tour by probabilistically choosing the next city using pheromone values and distances.

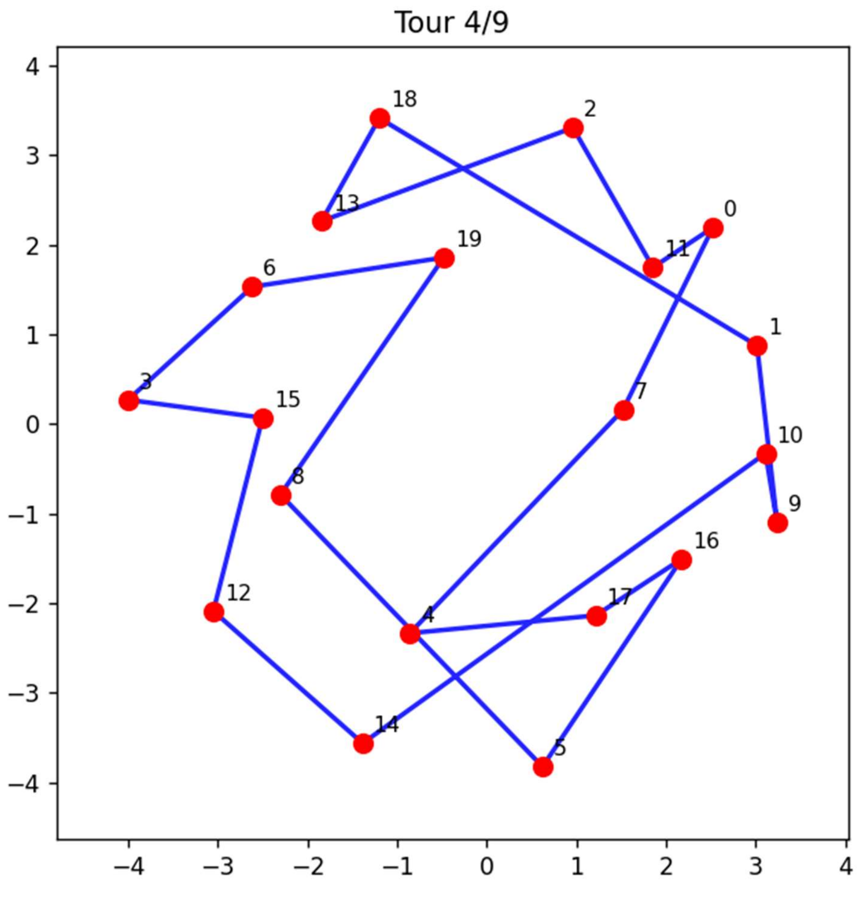
   c. Calculate the total distance of each tour.

   d. Update pheromones: evaporate existing values and deposit new pheromones on paths used by ants (more for shorter tours).
5. Track the shortest tour found so far.
6. Repeat until stopping condition is reached.
7. Output the best tour and its total distance.

Sample Input/Output:



```
Shortest Path So Far: [2, 13, 3, 15, 12, 14, 10, 7, 0, 11, 8, 5, 16, 17, 19, 6, 1, 9, 4, 18, 2], Length: 550
Shortest Path So Far: [4, 17, 19, 13, 2, 18, 15, 11, 0, 7, 1, 9, 10, 14, 8, 5, 16, 12, 6, 3, 4], Length: 550
Shortest Path So Far: [12, 16, 7, 0, 2, 18, 13, 19, 8, 11, 10, 14, 4, 9, 1, 6, 3, 15, 17, 5, 12], Length: 566
Shortest Path So Far: [19, 17, 16, 9, 1, 7, 0, 11, 10, 14, 4, 3, 6, 13, 18, 2, 8, 15, 12, 5, 19], Length: 539
Shortest Path So Far: [2, 13, 19, 6, 3, 16, 7, 11, 0, 15, 12, 5, 17, 4, 14, 10, 9, 1, 18, 8, 2], Length: 540
Shortest Path So Far: [0, 7, 16, 10, 9, 1, 11, 2, 18, 13, 5, 8, 19, 17, 4, 14, 12, 6, 3, 15, 0], Length: 541
Shortest Path So Far: [11, 0, 7, 1, 9, 4, 17, 16, 5, 13, 2, 18, 6, 3, 15, 12, 14, 8, 19, 10, 11], Length: 547
Shortest Path So Far: [18, 9, 1, 7, 0, 11, 2, 13, 6, 3, 4, 14, 10, 17, 16, 5, 12, 15, 8, 19, 18], Length: 561
Shortest Path So Far: [0, 7, 1, 18, 13, 2, 19, 6, 3, 15, 8, 11, 9, 10, 14, 4, 17, 16, 5, 12, 0], Length: 548
Shortest Path So Far: [13, 2, 18, 4, 14, 10, 1, 7, 0, 11, 8, 19, 6, 3, 15, 12, 16, 5, 17, 9, 13], Length: 564
Shortest Path So Far: [8, 2, 13, 18, 4, 14, 10, 9, 1, 7, 0, 11, 15, 16, 17, 19, 6, 3, 12, 5, 8], Length: 542
```


Tour 4/9