Task2:

1. Introduction and Explanation of ACO in the TSP Variant

Ant Colony Optimization (ACO) is a bio-inspired optimization technique based on the foraging behavior of ants. In the context of the Traveling Salesman Problem (TSP), each "ant" constructs a potential solution by traversing cities. The decision of which city to visit next is influenced by two factors:

1.1 Initialization:

A pheromone matrix (τ_xy) is initialized with a value of 1 on all edges.
The best cost and best path are initialized to infinity and empty respectively.

1.2 Iterations

The algorithm runs for 50 fixed iterations.
In each iteration, a number of ant agents (1, 5, 10, or 20) are created.
Each ant starts at a random city, builds a full tour using Roulette Wheel Selection based on pheromone and heuristic values.
After all ants complete their tours:
Pheromone is updated: Better (shorter) tours contribute more pheromone.
Pheromone evaporation ($\rho = 0.1$) is applied.
The global best solution is updated if a better one is found.

1.3 Construction of Ant Tours

For each ant, choose a path to follow from the starting point to the endpoint using a probabilistic decision rule that is based on the pheromone levels on the edges.

Probability = $(\tau\_ij^\alpha * \eta\_ij^\beta) / \Sigma(\tau\_ik^\alpha * \eta\_ik^\beta)$
where:
- τ is the pheromone level
- η is the inverse of distance
- $\alpha = 1$, $\beta = 5$ in this implementation

1.4 Pheromone Update

After all ants have constructed their tours, the pheromone levels on each edge are updated.
More pheromone is deposited on shorter (better) tours.
Additionally, pheromone evaporation is applied (controlled by parameter rho) to prevent stagnation and encourage exploration.

1.5 Termination

The algorithm terminates after 50 iterations.
The best tour and its corresponding distance are returned as the final solution.

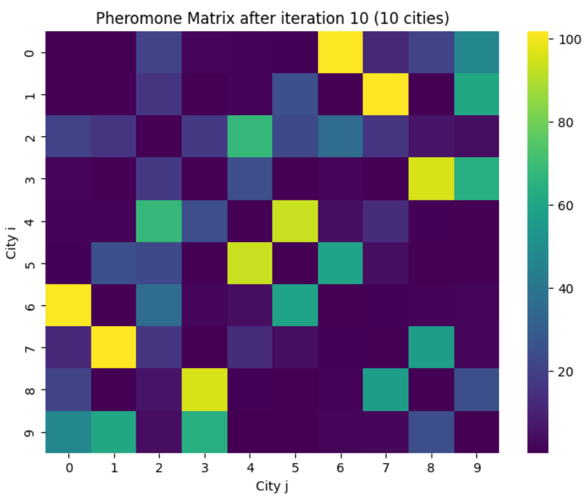
2. Implementation and Output Process

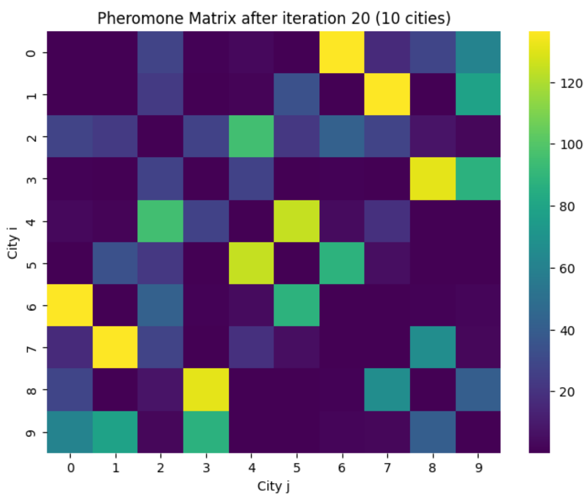
A distance matrix is generated only once per city size (10 and 20 cities).
Ant colony is created for each ant configuration: 1, 5, 10, and 20 ants.
For each run:
- Ants build complete tours.
- Pheromone is updated after each iteration.
- Best cost and average cost are recorded.

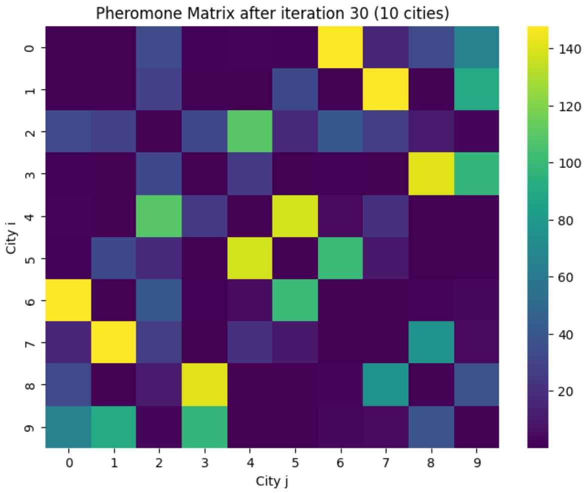
pheromone matrix every 10 iterations

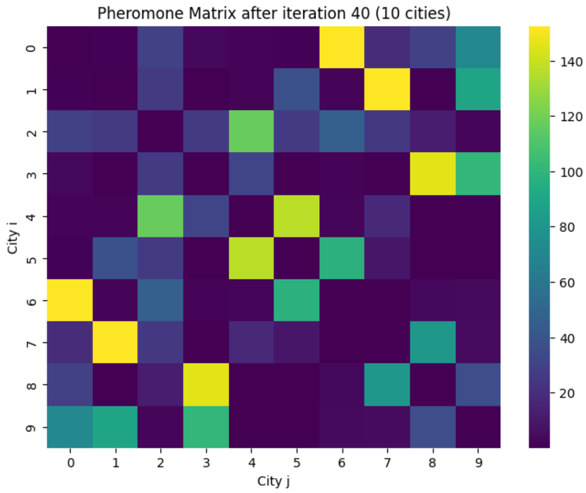
```
Iteration 10/50, Best cost: 95
Iteration 20/50, Best cost: 95
Iteration 30/50, Best cost: 95
Iteration 40/50, Best cost: 95
Iteration 50/50, Best cost: 95
```

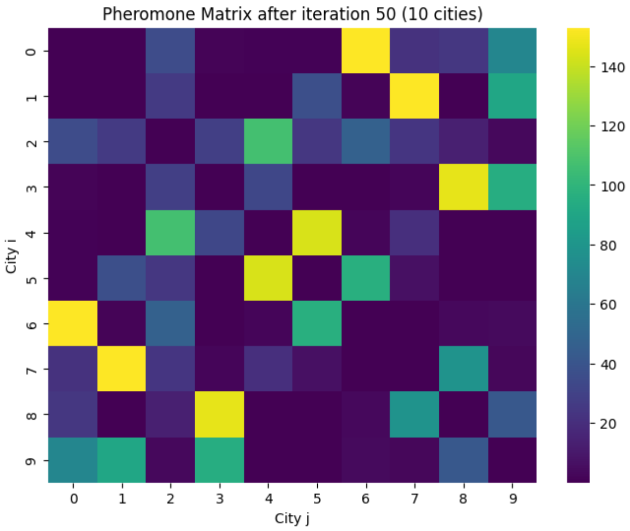
Visualizing pheromone matrix snapshots for 10 cities:



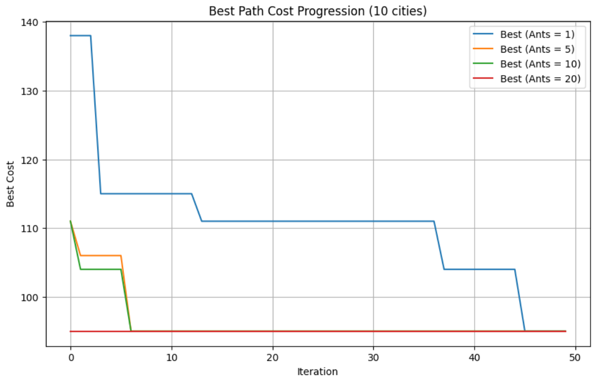


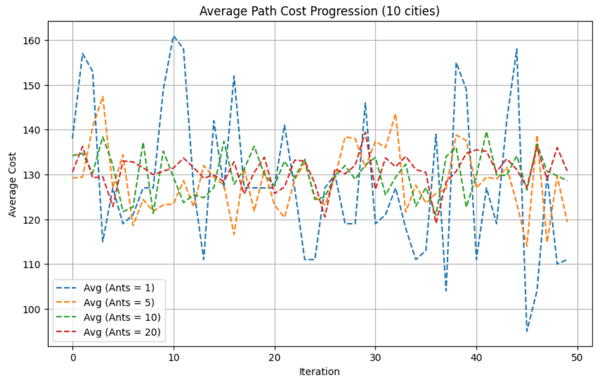

Pheromone Matrix after iteration 30 (10 cities)
City i
City j
0
1
2
3
4
5
6
7
8
9
20
40
60
80
100
120
140

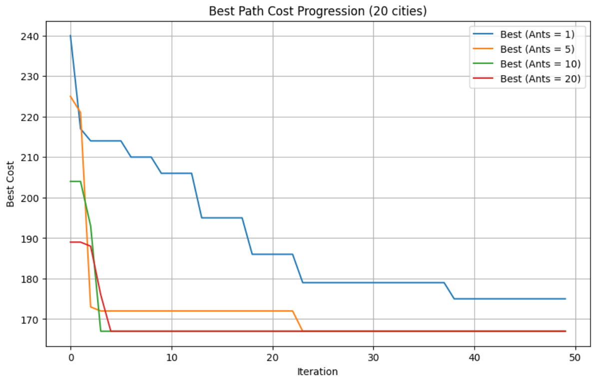

Pheromone Matrix after iteration 40 (10 cities)
City i
City j
0
1
2
3
4
5
6
7
8
9
20
40
60
80
100
120
140

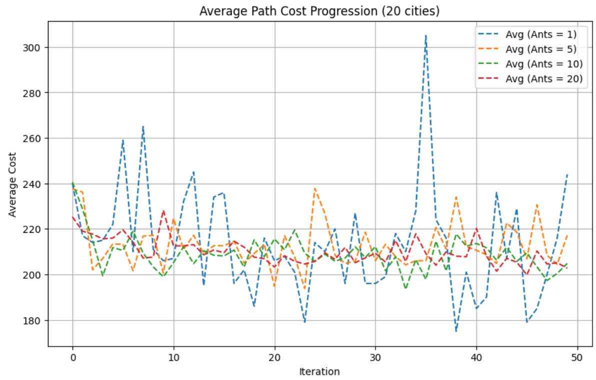

Pheromone Matrix after iteration 50 (10 cities)
City i
City j
0
1
2
3
4
5
6
7
8
9
20
40
60
80
100
120
140

Plotting Best Cost and Average Cost Per Iteration









3. Distance Matrix Generation

The distance between each pair of cities is randomly generated between 3 and 50 (inclusive). A symmetric matrix is used to represent the city-to-city distance.

```
Distance matrix for 10 cities:
[[ 0 42 50 22 37 28 11 28 17  9]
 [42  0 26 41 20 14 27  4 43  6]
 [50 26  0  8 14 20 48 42 26 19]
 [22 41  8  0  6 43 21 45  3  4]
 [37 20 14  6  0  9 25 22 31 31]
 [28 14 20 43  9  0 20 25 22 16]
 [11 27 48 21 25 20  0 42 28 38]
 [28  4 42 45 22 25 42  0 11 11]
 [17 43 26  3 31 22 28 11  0  7]
 [ 9  6 19  4 31 16 38 11  7  0]]
```

```
Distance matrix for 20 cities:
[[ 0 10 22 25 31  9 10 32 15 33 48 15 27 26 19 32 18 26 30 27]
 [10  0 25  5 30 25 50 20 46 36  8 24  7 49 32 37  6  4 15 12]
 [22 25  0  7 27 23 13 26 38 33 40 30 12  8  3 24 50 29 29 23]
 [25  5  7  0 38  9 41 44 30 31  5  3 41 29 20  3 49 39 36  8]
 [31 30 27 38  0 25 23 39 15 46 17 32 19 47  8  3 11 44 41 12]
 [ 9 25 23  9 25  0 45 29 20 48  6 33 24 20  8 28 31 21 44 45]
 [10 50 13 41 23 45  0 12 18 29 49 40 41 34 12 49 33 46  6 19]
 [32 20 26 44 39 29 12  0 35 26 19 27 32 19 42 41 22 39 32 45]
 [15 46 38 30 15 20 18 35  0 37 15 21 25 43 32 34 37 46  6 20]
 [33 36 33 31 46 48 29 26 37  0 43 33 39 37 43 38 18 38 22 25]
 [48  8 40  5 17  6 49 19 15 43  0  4 40 39 48 19 40 19 10  8]
 [15 24 30  3 32 33 40 27 21 33  4  0 47 39 29  5 10 15 14  4]
 [27  7 12 41 19 24 41 32 25 39 40 47  0  8  9 50 12 18 40  8]
 [26 49  8 29 47 20 34 19 43 37 39 39  8  0 18 22 15 44 28 32]
 [19 32  3 20  8  8 12 42 32 43 48 29  9 18  0  9 40 12 15 28]
 [32 37 24  3  3 28 49 41 34 38 19  5 50 22  9  0 13  9  9 18]
 [18  6 50 49 11 31 33 22 37 18 40 10 12 15 40 13  0 34 33 10]
 [26  4 29 39 44 21 46 39 46 38 19 15 18 44 12  9 34  0 37 37]
 [30 15 29 36 41 44  6 32  6 22 10 14 40 28 15  9 33 37  0 26]
 [27 12 23  8 12 45 19 45 20 25  8  4  8 32 28 18 10 37 26  0]]
```

4. Pheromone Map and Optimization Progress

For every 10 iterations, the current best path cost is recorded

```
Results for 10 cities:
Iteration 10/50, Best cost: 115
Iteration 20/50, Best cost: 111
Iteration 30/50, Best cost: 111
Iteration 40/50, Best cost: 104
Iteration 50/50, Best cost: 95
Ants: 1 -> Best path cost: 95
Iteration 10/50, Best cost: 95
Iteration 20/50, Best cost: 95
Iteration 30/50, Best cost: 95
Iteration 40/50, Best cost: 95
Iteration 50/50, Best cost: 95
Ants: 5 -> Best path cost: 95
Iteration 10/50, Best cost: 95
Iteration 20/50, Best cost: 95
Iteration 30/50, Best cost: 95
Iteration 40/50, Best cost: 95
Iteration 50/50, Best cost: 95
Ants: 10 -> Best path cost: 95
Iteration 10/50, Best cost: 95
Iteration 20/50, Best cost: 95
Iteration 30/50, Best cost: 95
Iteration 40/50, Best cost: 95
Iteration 50/50, Best cost: 95
Ants: 20 -> Best path cost: 95
```

```
Results for 20 cities:
Iteration 10/50, Best cost: 206
Iteration 20/50, Best cost: 186
Iteration 30/50, Best cost: 179
Iteration 40/50, Best cost: 175
Iteration 50/50, Best cost: 175
Ants: 1 -> Best path cost: 175
Iteration 10/50, Best cost: 172
Iteration 20/50, Best cost: 172
Iteration 30/50, Best cost: 167
Iteration 40/50, Best cost: 167
Iteration 50/50, Best cost: 167
Ants: 5 -> Best path cost: 167
Iteration 10/50, Best cost: 167
Iteration 20/50, Best cost: 167
Iteration 30/50, Best cost: 167
Iteration 40/50, Best cost: 167
Iteration 50/50, Best cost: 167
Ants: 10 -> Best path cost: 167
Iteration 10/50, Best cost: 167
Iteration 20/50, Best cost: 167
Iteration 30/50, Best cost: 167
Iteration 40/50, Best cost: 167
Iteration 50/50, Best cost: 167
Ants: 20 -> Best path cost: 167
```

5. Conclusion

- 10 Cities: The algorithm gradually improved the cost across iterations, with more ants generally yielding better and faster convergence.
- 20 Cities: The solution space is larger, making it harder to converge to the best path, especially with fewer ants.
- More ants help explore the space better, but also increase computational effort.

6. Comparison Between 10 and 20 Cities

- The same number of ants produced slower and less optimal results for 20 cities compared to 10.
- A larger city set requires more iteration or more agents to achieve similarly good performance.