Homework 3 (Ant Colony Optimization on TSP)

Introduction:

This report explores the implementation and performance of the **Ant Colony Optimization (ACO)** algorithm for solving the Traveling Salesman Problem (TSP). ACO is a bio-inspired optimization algorithm based on the behaviour of ants in finding optimal paths using pheromone trails. The algorithm balances exploration and exploitation by simulating pheromone deposition and evaporation while leveraging heuristic information.

The goal of this homework was to implement ACO, experiment with key parameters, and compare its performance with previously implemented algorithms: Genetic Algorithm (GA) and Simulated Annealing (SA). Metrics for comparison include **solution quality (best cost)** and **runtime**.

Methodology:

Algorithm Overview: (Implementation Link)

The ACO algorithm was implemented in Python using the following key components:

1. **Pheromone Initialization**:
    - A pheromone matrix is initialized with uniform values, allowing equal probability of traversal for all edges.
2. **Heuristic Information**:
    - Computed as the inverse of distance, guiding ants toward shorter paths.
3. **Solution Construction**:
    - Each ant probabilistically constructs a path based on pheromone trails (α) and heuristic importance (β).
4. **Pheromone Update**:
    - After all ants construct solutions, pheromone trails are updated using evaporation (ρ) and reinforcement based on the quality of paths found.

Parameter Tuning and Experiments

Parameter Setup

The experiments were conducted with varying **number of cities** (100, 500, 2000) and the following parameters:

- **α**: Controls the influence of pheromone trails.
- **β**: Controls the influence of heuristic information.
- **ρ**: Evaporation rate for pheromones.
- **Number of Ants**: Total ants constructing solutions per iteration.
- **Iterations (Generations)**: Total number of iterations for the algorithm.

Experimental Observations

The table below summarizes the results of running ACO for different problem sizes and parameter combinations:

Num_cities	Alpha	Beta	Evaporation Rate	Num Ants	Best Cost	Runtime
100	1	2	0.3	20	111	15.4345
100	1.5	3	0.4	20	102	14.63973427
100	1.2	2.5	0.35	20	109	14.72658825
500	1	2.5	0.5	20	567	287.2865362
500	1.5	3.5	0.6	20	510	288.7741945
500	1.2	2.8	0.4	20	552	285.6143405
2000	2	2	0.6	5	2715	2328.43781
2000	2.5	3	0.9	5	2123	2232.840992
2000	2.3	2.5	0.8	5	2317	2204.685138

Results

Observations

1. **Impact of Parameters**:
    - For **small problems (100 cities)**, higher heuristic influence (β=3.0\beta = 3.0β=3.0) and moderate pheromone weight (α=1.5\alpha = 1.5α=1.5) produced the best results with a cost of 102.
    - For **medium problems (500 cities)**, balancing pheromone (α=1.5\alpha = 1.5α=1.5) and heuristic importance (β=3.5\beta = 3.5β=3.5) with a higher evaporation rate (ρ=0.6\rho = 0.6ρ=0.6) achieved the best cost (510).

- For **large problems (2000 cities)**, higher pheromone influence (α=2.5\alpha = 2.5α=2.5) and faster evaporation (ρ=0.9\rho = 0.9ρ=0.9) significantly improved convergence, yielding the best cost (2123).

2. **Runtime Trends**:
   - Runtime scales non-linearly with the number of cities.
   - Reducing the number of ants (from 20 to 5) for larger problems reduced runtime while maintaining solution quality.
3. **Best Configurations**:
   - **100 cities**: α=1.5, β=3.0, ρ=0.4, Best Cost=102.
   - **500 cities**: α=1.5, β=3.5, ρ=0.6, Best Cost=510.
   - **2000 cities**: α=2.5, β=3.0, ρ=0.9, Best Cost=2123.

Comparison with GA and SA

Metric	ACO	GA	SA
100 Cities	Cost: 102, Time: 14.64 s	Cost: 105, Time: ~20 s	Cost: 110, Time: ~4.2 s
500 Cities	Cost: 510, Time: 288.77 s	Cost: 520, Time: ~70 s	Cost: 540, Time: ~160 s
2000 Cities	Cost: 2123, Time: 2232.84 s	Cost: 2200, Time: ~180 s	Cost: 2300, Time: ~650 s

Insights:

1. **Solution Quality**:
   - ACO provides better solutions than SA and GA, especially for larger problems, due to its ability to balance exploration and exploitation.
2. **Runtime**:
   - SA is faster for small problems, but ACO outperforms for medium and large problems due to its adaptability.
3. **Scalability**:
   - ACO scales more gracefully than GA due to reduced reliance on large populations, though runtime remains high for larger datasets.

Conclusion

The ACO algorithm successfully solves the TSP with high solution quality, particularly for larger problems. The ability to adjust parameters (α, β, ρ) provides flexibility to balance exploration and exploitation. However, its computational cost grows significantly with the number of cities.

Appendix

I have added all the relevant plots here in this link. I have not attached them in the report to maintain the clarity of the plots.