TRAVELING SALESMAN PROBLEM USING CLASSICAL AND SOM BASED METHODS

GITHUB LINK: https://github.com/MORISON-K/TRAVELLING-SALESMAN-PROBLEM.git

Participants

NAME	REG NO.
Lunkuse Dorcus	24/U/06515/PS
Serena Robinah	24/U/11034/PS
Aturinzire Hargreave	24/U/22602/PS
Abinsinguza Morison.K	24/U/02594/PS
Biar Elijah Mabior	24/E/21430/PS

Problem Description

The Traveling Salesman Problem requires finding the shortest possible route that visits each city exactly once and returns to the starting city. Our implementation uses the following graph with 7 cities.

Graph Representation

```
adjacency_matrix = [
    [0, 12, 10, inf, inf, inf, 12],  # Node 1 (index 0)
    [12, 0, 8, 12, inf, inf, inf],   # Node 2 (index 1)
    [10, 8, 0, 11, 3, inf, 9],       # Node 3 (index 2)
    [inf, 12, 11, 0, 11, 10, inf],   # Node 4 (index 3)
    [inf, inf, 3, 11, 0, 6, 7],      # Node 5 (index 4)
    [inf, inf, inf, 10, 6, 0, 9],    # Node 6 (index 5)
    [12, inf, 9, inf, 7, 9, 0]       # Node 7 (index 6)
  ]
```

Dynamic Programming Approach

The Dynamic Programming implementation:

- Uses the Held-Karp algorithm to find the optimal solution

- Has time complexity $O(n^2 \cdot 2^n)$ and space complexity $O(n \cdot 2^n)$

- Guarantees finding the optimal solution

- Uses bit manipulation and memoization for efficiency

SOM Approach

The Self-Organizing Map implementation:

- Creates a neural network ring that adapts to the city positions

- Uses competitive learning with a neighborhood function

- Provides a heuristic solution that approximates the optimal tour

- Can scale to larger problems where exact methods become infeasible

Comparison of Methods

Aspect	Dynamic Programming	Self-Organizing Map
Optimality	Guarantees optimal solution	Approximates optimal solution
Time Complexity	$O(n^2 \times 2^n)$	$O(n^2 \times m)$ where m is iterations |
Scalability	Limited to the number of cities	Can handle hundreds of cities
Memory Usage	Exponential ($O(n \times 2^n)$)	Linear ($O(n)$)

Results

Dynamic Programming

- Finds the optimal tour with minimal total distance

SOM

- Approximates the TSP solution using neural network approach

- Results may vary due to the stochastic nature of the algorithm

Suggestions for Improvements or Extensions

Hybrid Approach: Use SOM to generate an initial solution and refine it using DP or another exact method.

Optimized SOM Parameters: Adjust learning rates, neuron count, and iterations to improve accuracy.

Metaheuristics (e.g., Genetic Algorithm, Simulated Annealing): These can balance efficiency and accuracy better than pure SOM.

Parallel Computing for DP: This could help mitigate its exponential complexity for larger instances.