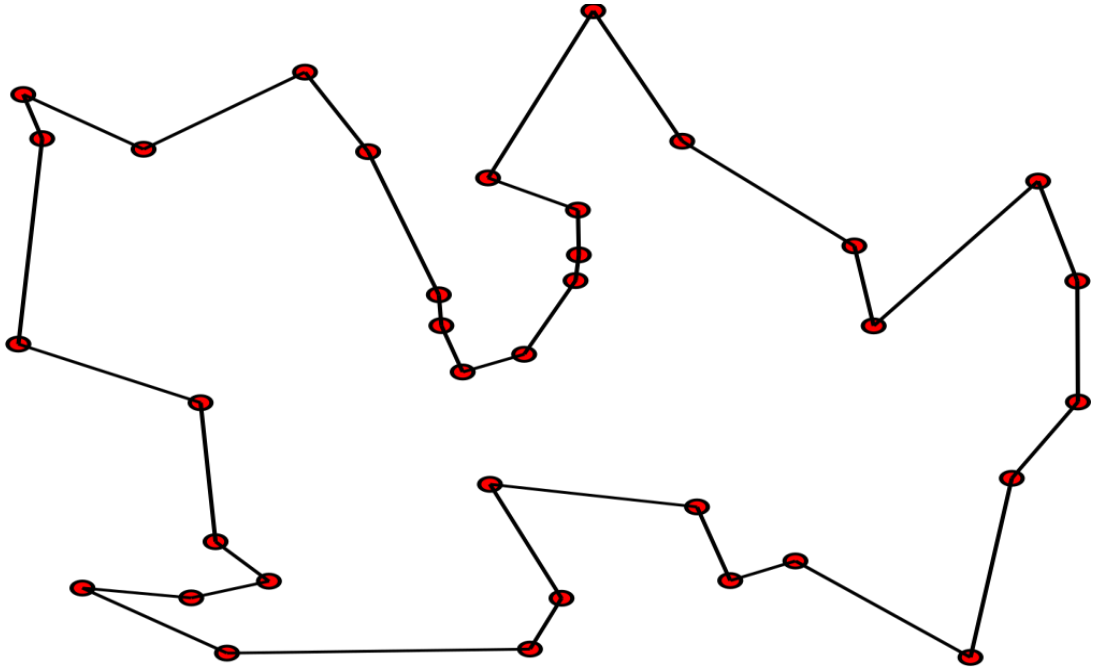
Solving Travelling Salesman Problem with Hill Climbing & Simulated annealing

Problem Statement

- Optimization problem where a salesperson needs to find the shortest possible route that visits a set of cities exactly once and returns to the starting city.
- Given a set of cities and the distances between them, what is the shortest possible route that visits each city exactly once and returns to the starting point.
  - Suppose a salesman needs to visit 4 cities:
    $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$
  - The goal is to find the route with the minimum total distance.



Implementation Details of Hill Climbing for TSP

Node Representation

- The cities are represented as random (x, y) coordinates in a 2D space.
- Generate num_cities = 10 randomly using NumPy.

Heuristic Function

- The Euclidean Distance heuristic is used

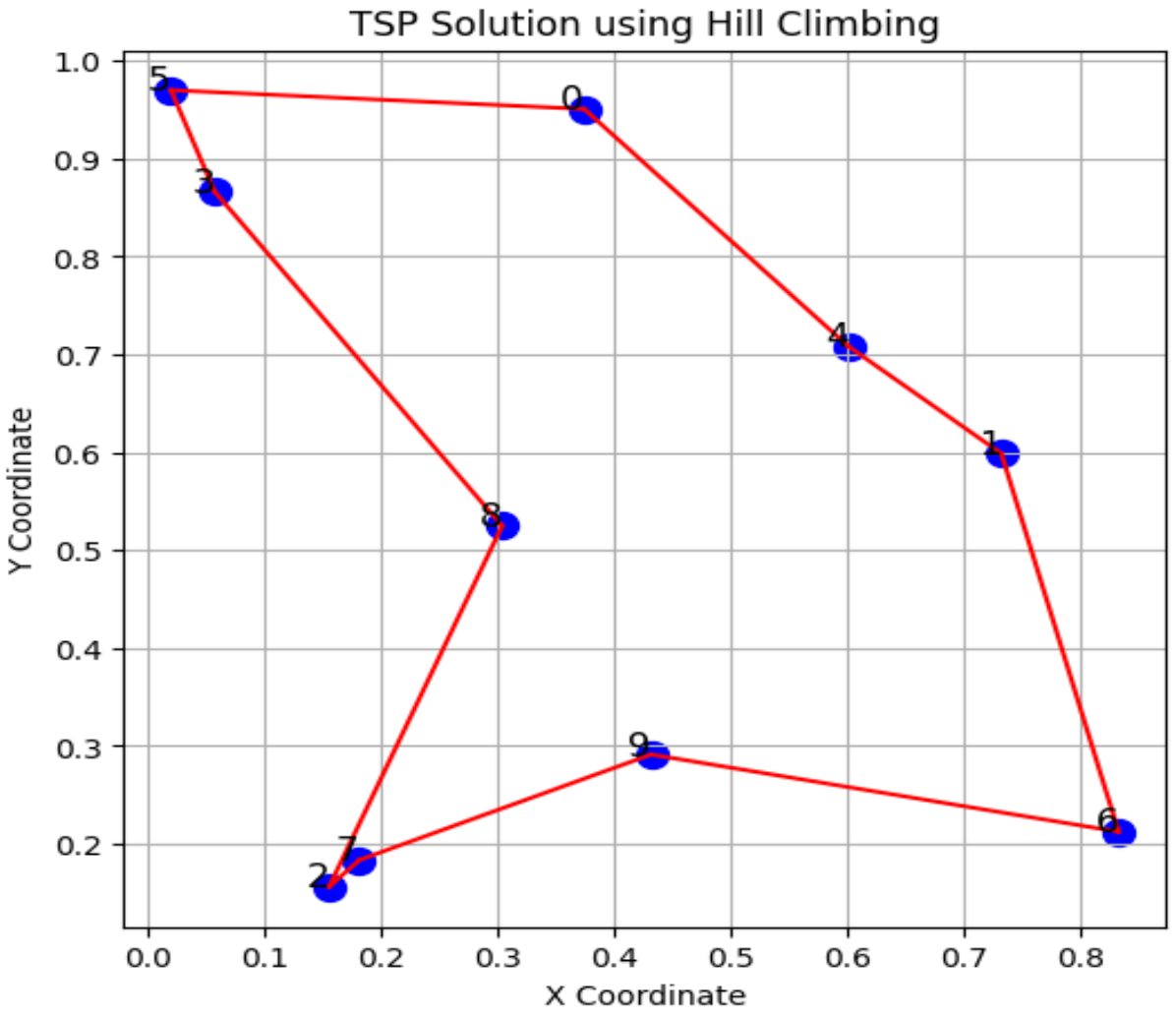
$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Branching (Expanding Nodes)

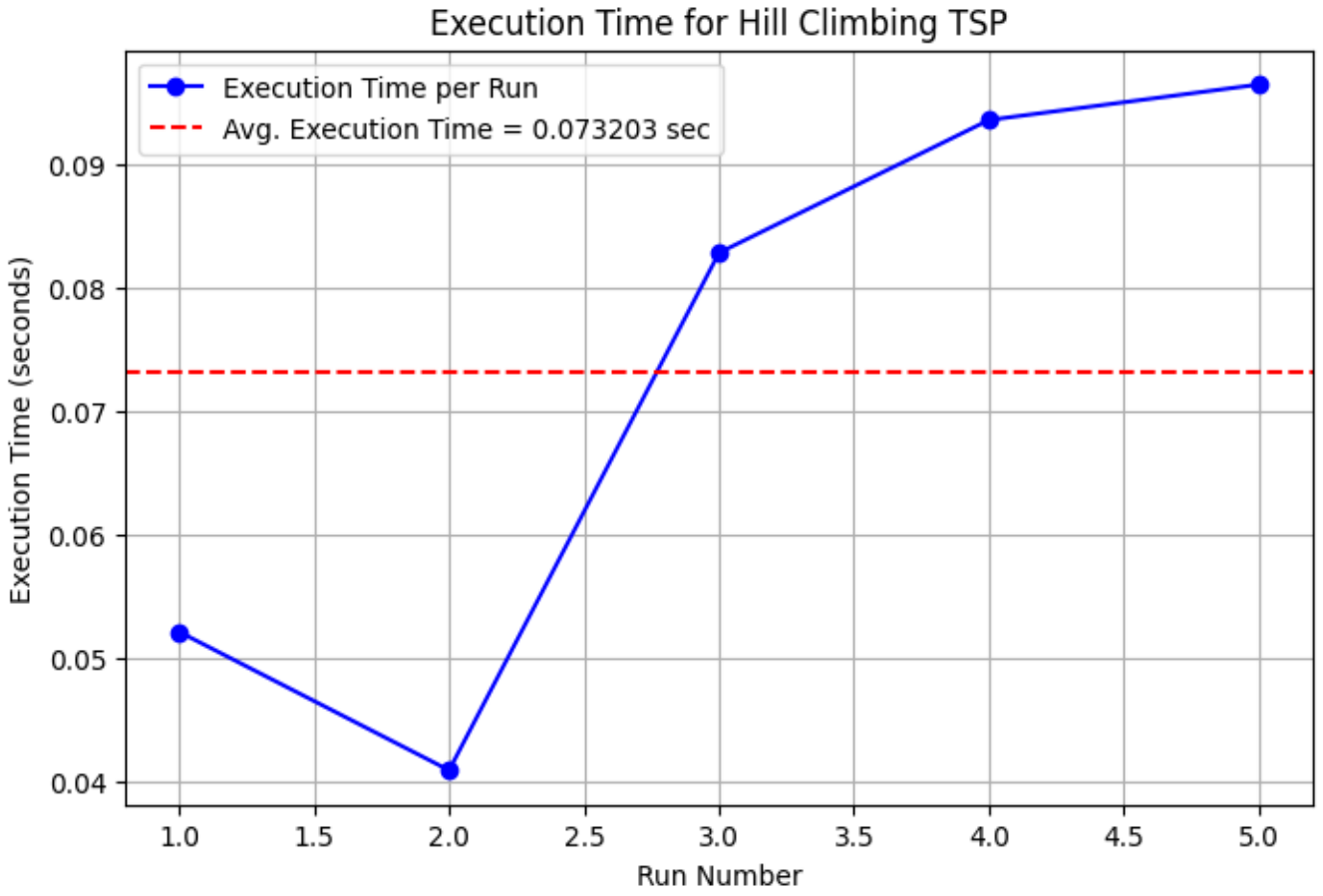
- The algorithm starts with a random route, which is a shuffled list of city indices
- A neighboring solution is generated by swapping two randomly chosen cities
- If the new route has a shorter distance, it replaces the current solution.

Performance Analysis

Visualization



Average Execution Time



Best route found: [1, 6, 9, 7, 2, 8, 3, 5, 0, 4]

Total distance of best route: 2.9031

Average execution time over 5 runs: 0.073203 seconds

Implementation Details of Simulated Annealing for TSP

Node Representation

- The cities are represented as random (x, y) coordinates in a 2D space.
- Generate num_cities = 10 randomly using NumPy.

Heuristic Function

- The Euclidean Distance heuristic is used
- This function computes the straight-line distance between two cities.

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Branching (Expanding Nodes)

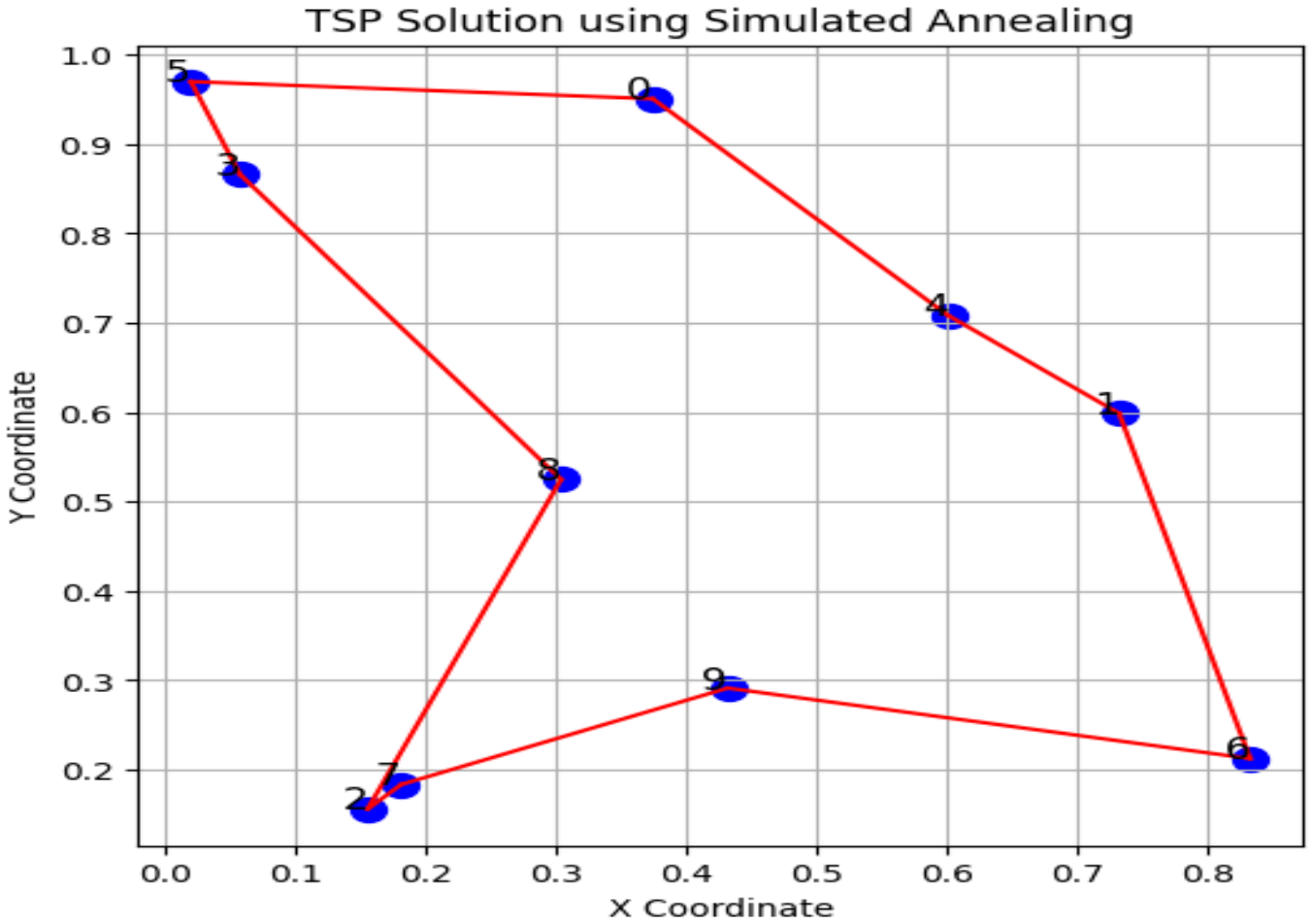
- The algorithm starts with a random route, which is a shuffled list of city indices
- A neighboring solution is generated by swapping two randomly chosen cities
- If the new route has a shorter distance, it replaces the current solution.

Simulated Annealing Algorithm

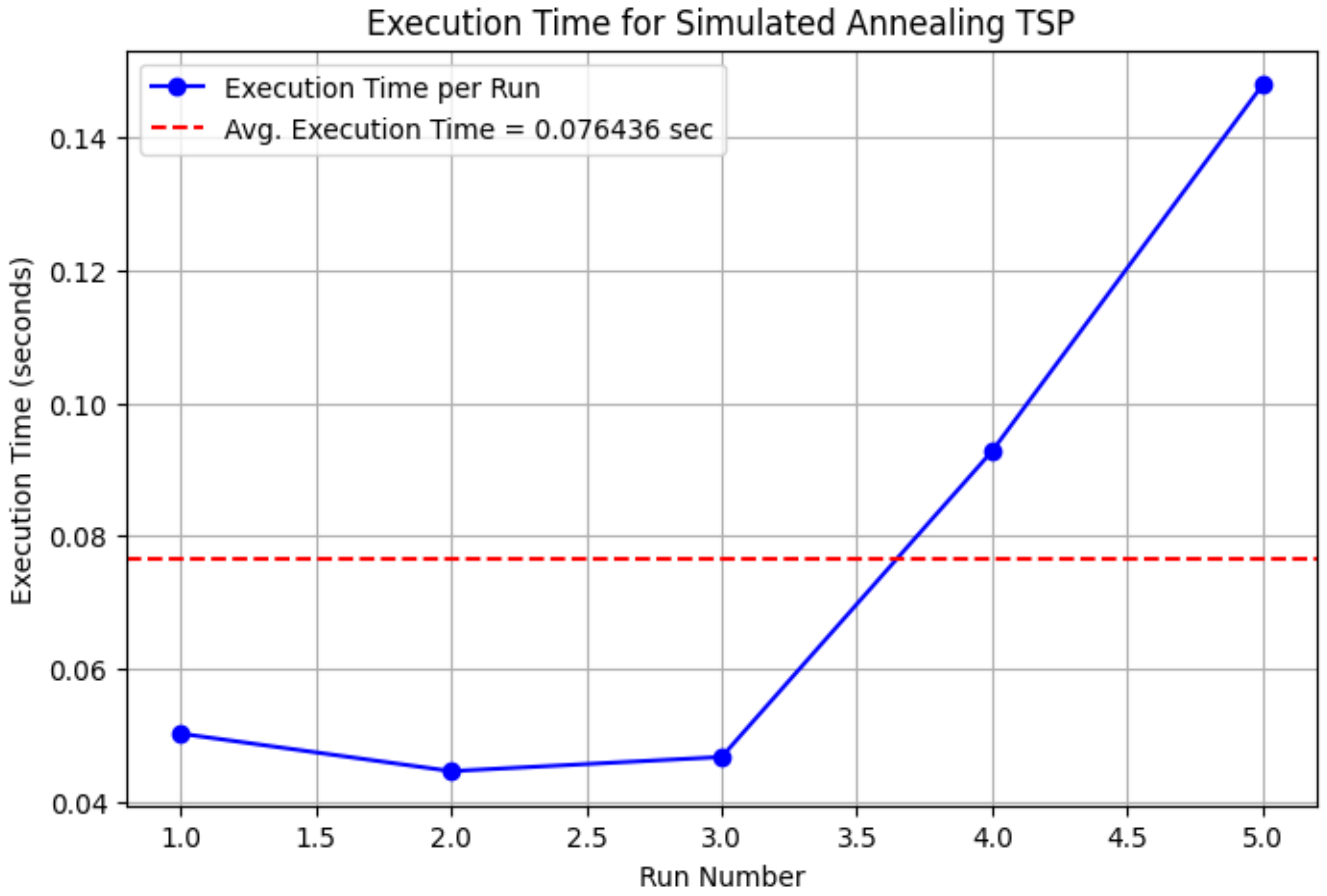
- max_iterations: Number of iterations for optimization.
- initial_temp: Initial temperature for the annealing process.
- cooling_rate: Determines how the temperature decreases.
- Random initial solution: A random tour is created by shuffling city indices.
- Track the best solution: The best route found is stored separately.
- Set initial temperature: High temperatures allow more diverse exploration.
- **Randomly swap two cities** to generate a neighboring solution.
- Always accept a better solution (delta < 0).
- If the new solution is **better than the best recorded**, update the best route.
- Gradually **reduce the temperature**, limiting the probability of accepting worse solutions over time.

Performance Analysis

Visualization



Average Execution Time



Best route found: [2, 8, 3, 5, 0, 4, 1, 6, 9, 7]

Total distance of best route: 2.9031

Average execution time over 5 runs: 0.076436 seconds