Advanced Algorithm Lab Assignment

Implementation of Heuristic Algorithms for Real-World Optimization Problems

1 Problem Statement

Case Study: Logistics Route Optimization for Last-Mile Delivery

A logistics company operating in an urban area must deliver parcels from its central depot to multiple customer locations every morning and return to the depot before the next dispatch cycle. The goal is to minimize the total travel distance while visiting each customer exactly once.

2 Dataset

Use the following dataset of 20 delivery points represented by (x, y) coordinates in kilometers. These points can be visualized on a 2D grid or mapped approximately to real-world regions.

Location ID	X-Coordinate	Y-Coordinate
1	5	8
2	12	4
3	8	14
4	16	10
5	3	9
6	11	17
7	14	2
8	9	6
9	7	3
10	18	12
11	2	7
12	10	15
13	6	11
14	17	5
15	4	4
16	8	10
17	1	13
18	15	15
19	13	8
20	19	9

Distance Metric: Use Euclidean distance between any two points $A(x_1, y_1)$ and $B(x_2, y_2)$:

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

Depot: Location 1 (Start and End Point)

3 Tasks

Task 1: Greedy Heuristic Implementation

- Start from the depot.
- Repeatedly move to the nearest unvisited location until all are visited.
- Return to the depot.

Implementation Guidelines:

- Maintain a list of visited locations.
- Use a loop to select the nearest neighbor at each step.
- Compare route cost for different starting points.

Outputs:

- Final route (sequence of location IDs).
- Total route distance.
- Execution time.

Task 2: Hill Climbing Heuristic

- Begin with a random route (permutation of all locations).
- Generate neighboring routes by swapping two cities.
- Move to the neighbor only if it improves total distance.
- Repeat until no better route exists.

Implementation Guidelines:

- Define a distance function to compute total route cost.
- Implement stopping condition (no improvement or iteration limit).

Outputs:

- Best route found.
- Distance vs. iteration plot.
- Discussion on local optima.

Task 3: Simulated Annealing

- Start with a random route.
- Generate neighboring routes by swapping two locations.
- Accept worse solutions with probability $P = e^{-\Delta E/T}$, where ΔE is the increase in distance and T is temperature.
- Gradually reduce temperature: $T = \alpha T$.

Typical Parameters:

- Initial Temperature: 100
- Cooling Factor (α): 0.95
- Stop when $T < 0.001$ or iteration limit reached.

Outputs:

- Best route and total cost.
- Temperature vs. cost plot.
- Discussion on temperature schedule.

Task 4: Tabu Search

- Start with an initial random route.
- Define neighborhood function (swap or insertion).
- Maintain a Tabu List of recent moves.
- Select the best move not in the Tabu List.

Implementation Guidelines:

- Implement a queue-based Tabu List with fixed size (5–10).
- Track and update the best solution separately.

Outputs:

- Final route and cost.
- Cost vs. iteration plot.
- Discussion on effect of Tabu tenure.

4 Performance Evaluation

1. Apply all four algorithms to the same dataset.
2. Record performance metrics as shown below:

Algorithm	Best Distance	Time (s)	Iterations	Remarks
Greedy				
Hill Climbing				
Simulated Annealing				
Tabu Search				

Include route visualizations and performance graphs using Python's `matplotlib`.

5 Implementation Specifications

Parameter	Recommended Value / Remark
Programming Language	Python / C++
Distance Metric	Euclidean
Initial Temperature (SA)	100–1000
Cooling Factor (α)	0.90–0.99
Tabu Tenure	5–10
Iteration Limit	5000–10000
Visualization Tool	`matplotlib` or any plotting library

6 Deliverables

Each student must submit:

1. Source code files for all four algorithms.
2. A concise report as a single PDF containing:
   - Problem formulation and description.
   - Implementation summary.
   - Experimental results and graphs.
   - Comparative analysis and discussion.
   - Observations and conclusions.
3. Screenshots of outputs and visualizations.