Simulated Annealing for TSP and Jigsaw Puzzle Reconstruction

Gajipara Nikunj, Parmar Divyraj
M.Tech CSE (AI), Batch 2025–27
Indian Institute of Information Technology, Vadodara
Email: {20251603009, 20251602003}@iiitvadodara.ac.in

***Abstract*—This experiment applies the Simulated Annealing (SA) algorithm to two optimization problems—the Traveling Salesman Problem (TSP) and a Jigsaw Puzzle reconstruction task. In Part A, SA finds a near-optimal tour visiting 20 Rajasthan tourist locations with minimal total travel cost. In Part B, SA reassembles a scrambled 4×4 Lena image by minimizing adjacency mismatch energy. Both experiments demonstrate SA's ability to escape local minima and produce high-quality solutions for combinatorial optimization.**

***Index Terms*—Simulated Annealing, Traveling Salesman Problem, Jigsaw Puzzle, Combinatorial Optimization**

I. Objective

- Apply Simulated Annealing to optimize combinatorial problems.
- Implement SA for two tasks: TSP and Jigsaw reconstruction.
- Analyze convergence, cost reduction, and final outputs.
- Evaluate performance based on total cost and adjacency energy.

II. Problem Definition

Part A — Traveling Salesman Problem (TSP)

Given 20 tourist locations in Rajasthan, find the shortest possible route that visits each city exactly once and returns to the starting city.

Part B — Jigsaw Puzzle Reconstruction

Given a scrambled Lena image divided into 4×4 blocks, find the block permutation that minimizes the sum of adjacency mismatches between neighboring tiles.

III. Approach

GitHub Repository: https://github.com/NikunjGajipara27/Lab-Assignment

A. Simulated Annealing Overview

Simulated Annealing (SA) is a probabilistic optimization technique inspired by the annealing process in metallurgy. It uses a temperature parameter T that gradually decreases, controlling the probability of accepting worse solutions to escape local minima.

B. General Algorithm

```
Initialize state s, temperature T
Repeat until stopping condition:
    Generate neighbor s'
    Compute E = cost(s') - cost(s)
    If E < 0:
        Accept s'
    Else accept with probability exp(-E / T)
    Decrease temperature T ← T
Return best state found
```

IV. Implementation Details

A. Part A — TSP Implementation

- **State:** A permutation of 20 city indices.
- **Neighbor:** Random 2-swap of city order.
- **Cost:** Total round-trip distance (Haversine formula).
- **Cooling Schedule:** $T \leftarrow 0.995T$ per iteration.
- **Acceptance:** Metropolis criterion $e^{-\Delta E/T}$.

B. Part B — Jigsaw Puzzle Implementation

- **State:** A permutation of 16 block indices.
- **Neighbor:** Swap of two blocks.
- **Cost (Energy):** Sum of L_1 pixel differences across adjacent block borders.
- **Cooling:** Geometric schedule with $\alpha = 0.997$.
- **Termination:** When $T < 1$ or after 10^6 iterations.

V. Experimental Setup

Table I
Experimental Setup Parameters

Programming Language	Python 3.10
Algorithm	Simulated Annealing (SA)
Dataset A	20 Rajasthan tourist locations
Dataset B	Scrambled Lena image (4×4 grid)
Cooling Rate	0.995 (TSP), 0.997 (Jigsaw)
Initial Temperature	8000
Stopping Criteria	$T < 1$ or max iterations
Output	Best tour / Reconstructed image

Figure 1. Example Jigsaw Input / Intermediate Image.

VI. Results and Analysis

Part A — TSP Results

The SA algorithm produced a best tour of 20 Rajasthan cities with total distance:

$$\text{Best Cost} \approx 3021.16 \text{ km}.$$

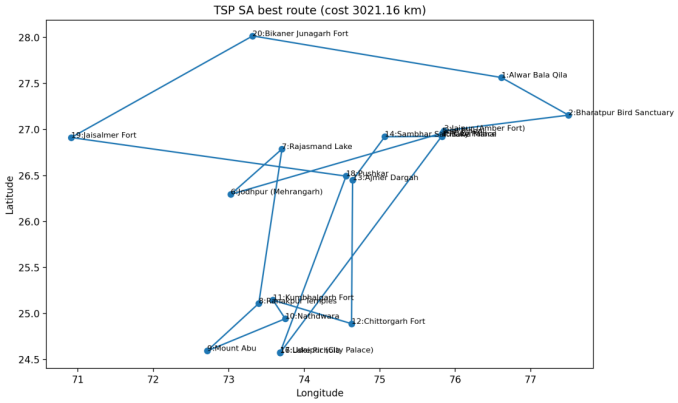

Figure 2. TSP SA best route (cost 3021.16 km).

Output Files:

- `tsp_demo_route.png` — final best route.
- `tsp_demo_history.png` — cost convergence plot.
- `tsp_demo_tour.csv` — ordered tour with distances.

Tour Order:

```
19, 15, 0, 1, 3, 12, 17, 13, 16, 18,
8, 9, 5, 14, 2, 7, 6, 4, 11, 10
```

Part B — Jigsaw Puzzle Results

The final assembled image achieved the lowest adjacency energy of:

$$\text{Final Energy} = 56588.$$

Output Files:

- `jigsaw_final_tuned_unscrambled_energy_56588.png` — final reconstructed image.
- `jigsaw_final_tuned_best_perm.txt` — best block permutation.
- `jigsaw_final_tuned_runs.csv` — energy and timing logs.

Best Permutation:

```
13 14 15 12 3 0 1 2 7 4 5 10 6 11 8 9
```

Observations

- SA rapidly decreases cost early, then stabilizes near optimal values.
- For TSP, SA produced a nearly optimal tour comparable to known heuristics.
- For Jigsaw, the final image achieved smooth adjacency continuity.
- Longer runs and slower cooling improve solution quality.

VII. Discussion

- SA effectively handles permutation-based optimization.
- Acceptance of worse moves helps escape local minima.
- Cooling rate critically affects convergence speed and final cost.
- SA can be extended to hybrid models combining local search.

VIII. Conclusion

Simulated Annealing successfully optimized both the TSP and the Jigsaw reconstruction problems. The algorithm demonstrated convergence towards high-quality solutions with smooth cost reduction. Further improvements may include adaptive cooling, hybrid SA-GA models, or parallel multi-start execution for robustness.

IX. Sample Command Execution

```
# For TSP
python LAB_4.py tsp --mode builtin --max_steps 500
  --out_prefix tsp_demo

# For Jigsaw Puzzle
python LAB_4.py jigsaw --input scrambled_lena.mat
  --max_steps 1000000 --restarts 3 --seed 42 \
  --initial_temp 8000 --cooling_rate 0.997 \
  --iter_per_temp 400 --out_prefix jigsaw_final_tu
```

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.

[3] AI Lab Manual, Lab Assignment 4 – Simulated Annealing for TSP and Jigsaw Puzzle, 2025.