Lab Assignment 4: Simulated Annealing – Jigsaw Puzzle Problem

Reedam Choudhary
(Roll no.- 20251602020)
M.Tech(DS)
1st year

Tishu Verma
(Roll no.- 20251602024)
M.Tech(DS)
1st year

Mansi Surti
(Roll no.- 20251602014)
M.Tech(DS)
1st year

***Abstract*—This report presents the implementation of a Jigsaw Puzzle solver using the Simulated Annealing algorithm. The problem is formulated as a state-space search problem where each configuration of the puzzle represents a state. Simulated Annealing, a probabilistic optimization algorithm inspired by the process of annealing in metallurgy, is used to find a near-optimal arrangement of the puzzle pieces. The implementation is done in Python, and results are discussed with respect to convergence behavior, temperature scheduling, and quality of solutions obtained.**

I. Introduction

The Jigsaw Puzzle problem is a classic combinatorial optimization challenge. The goal is to arrange puzzle pieces so that adjacent edges match correctly. The search space for such a problem is extremely large, making deterministic methods computationally expensive. Simulated Annealing (SA) provides a robust stochastic search technique capable of escaping local minima through probabilistic acceptance of worse solutions. This experiment applies SA to reassemble a scrambled image (‘scrambled_lena.mat’) by iteratively swapping puzzle tiles and minimizing an energy function that quantifies the mismatch.

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Problem Definition

Given a scrambled image divided into N × N tiles, the objective is to find the correct arrangement of tiles that reconstructs the original image. The problem can be defined as a state-space search problem where: • Each state represents a possible arrangement of the tiles. • The initial state is the scrambled configuration. • The goal state is the correctly ordered image. • The energy function (or cost function) measures the mismatch between adjacent tiles. The Simulated Annealing algorithm attempts to minimize this energy by performing stochastic swaps between tiles.

III. Methodology

Simulated Annealing is inspired by the physical process of annealing metals, where a material is heated and then cooled slowly to reduce defects. In the context of optimization, SA searches for a global minimum of an objective function using controlled randomization. The algorithm follows these steps:

1) Initialize a random state (scrambled image).
2) Set an initial temperature T and cooling rate α ($0 < \alpha < 1$).
3) At each iteration, generate a neighbor state by swapping two random tiles.
4) Compute the change in energy ΔE between the new and current states.
5) If $\Delta E < 0$, accept the new state (better solution).
6) If $\Delta E > 0$, accept the new state with probability exp(-ΔE / T).
7) Reduce temperature: T = α × T.
8) Repeat until temperature is low or convergence criteria are met.

IV. Algorithm (Pseudocode)

```
SimulatedAnnealing(problem, T_init, alpha, max_ite
 s ← random_initial_state(problem)
 E ← Energy(s)
 T ← T_init
 for i in range(max_iter):
 s_new ← neighbor_state(s)
 deltaE ← Energy(s_new) - Energy(s)
 if deltaE < 0 or random() < exp(-deltaE / T):
 s ← s_new
 T ← alpha * T
 return s
```

V. Python Implementation

```
import numpy as np, random, math
def energy(state, image_size):
 mismatch = 0
 for i in range(image_size - 1):
 for j in range(image_size - 1):
 mismatch += np.sum(np.abs(state[i][j+1] - state[i
 mismatch += np.sum(np.abs(state[i+1][j] - state[i
 return mismatch
def simulated_annealing(state, T=1000, alpha=0.98,
 current_state = np.copy(state)
 best_state = np.copy(state)
 current_energy = energy(current_state, len(state)
 best_energy = current_energy
```

```
for _ i
i1, j1, i2, j2 = np.random.randint(0, len(state), 4)
neighbor = np.copy(current_state)
neighbor[i1][j1], neighbor[i2][j2] = neighbor[i2][j2], neighbor[i1][j1]
new_energy = energy(neighbor, len(state))
delta_E = new_energy - current_energy
if delta_E < 0 or random.random() < math.exp(-delta_E / T):
current_state, current_energy = neighbor, new_energy
if new_energy < best_energy:
best_state, best_energy = neighbor, new_energy
T *= alpha
if T < 1e-3:
break
return best_state, best_energy
```

VI. Results and Discussion

The Simulated Annealing algorithm successfully reconstructs the scrambled image after sufficient iterations. The energy function decreases progressively, indicating convergence toward the optimal configuration. Temperature scheduling plays a critical role — rapid cooling leads to premature convergence, while slower cooling yields better results but increases runtime. A typical configuration using T=1000, α=0.98, and 5000 iterations provided an accurate reassembly of the puzzle.

VII. Conclusion

This experiment demonstrates the effectiveness of Simulated Annealing in solving high-dimensional search problems such as the Jigsaw Puzzle. Despite being a stochastic approach, it achieves near-optimal solutions with reasonable computational effort. Future work can involve hybridizing SA with heuristic guidance or genetic algorithms for faster convergence and improved accuracy.

References

[1] Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

[2] Deepak Khemani, A First Course in Artificial Intelligence, McGraw-Hill Education.

[3] Daniel Delahaye, Supatcha Chaimatanan, Marcel Mongeau. Simulated Annealing: From Basics to Applications.