Experiment 4
Non-deterministic Search and Simulated Annealing

***Abstract*—This report explores techniques like Non-Deterministic Search and Simulated Annealing to solve the Travelling Salesman Problem (TSP), which is a well-known difficult problem. Simulated Annealing works by making random moves, with the likelihood of a move being made based on the potential improvement it offers. This report applies the Simulated Annealing algorithm to find a solution for the TSP.**

I. INTRODUCTION

The Travelling Salesman Problem (TSP) is a key challenge in combinatorial optimization. It asks: "Given a set of cities and distances, what is the shortest route that visits each city once and returns to the start?" The TSP is crucial in computer science and operations research because it is NP-hard, meaning solving it efficiently for large inputs is extremely difficult, and no known algorithm can solve all cases quickly.

The Travelling Salesman Problem (TSP) can be visualized as an undirected weighted graph, where cities are the vertices and the paths between them are the edges, each with a distance or cost. The goal is to find the shortest Hamiltonian cycle, a route that visits every vertex exactly once and returns to the starting point, minimizing the total travel distance. TSP is usually modeled as a complete graph, where every pair of cities is connected by an edge. If no direct path exists between cities, an artificial edge with a large weight can be added to ensure the graph remains complete without affecting the optimal solution.

The Travelling Salesman Problem (TSP) has practical applications in fields like logistics, transportation, telecommunications, and circuit design. Despite its computational challenges, researchers continually develop new algorithms and heuristics to find efficient, near-optimal solutions for real-world problems, such as route planning, network optimization, and supply chain management.

II. NON-DETERMINISTIC SEARCH

Non-deterministic search is a problem-solving approach that explores multiple potential solutions simultaneously, allowing algorithms to make random moves at each step. This flexibility enables the discovery of various solutions, particularly in complex scenarios like the Travelling Salesman Problem, where exhaustive searches for optimal solutions can be time-consuming. Unlike deterministic algorithms, which yield consistent outputs, non-deterministic algorithms can exhibit different behaviors across executions, investigating multiple routes to uncover a range of outcomes. This variability enhances their effectiveness in optimization tasks, particularly when exact solutions are computationally expensive or unrealistic, facilitating innovative solutions that might not be found through traditional methods.

III. TRAVELLING SALESMAN PROBLEM

The Travelling Salesman Problem (TSP) is a well-known question in combinatorial optimization that asks: "Given a list of cities and the distances between each pair, what is the shortest route that visits each city exactly once and returns to the starting point?" As an NP-hard problem, the TSP is significant in both theoretical computer science and operations research, as it exemplifies the challenges of optimizing routes and logistics in various practical applications.

Mathematical Formulation:

$$\text{Minimize } Z = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}$$

Constraints:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \quad (1 \le i \le n)$$

$$\sum_{i=1}^{n} x_{ji} = 1 \quad \forall j \quad (1 \le j \le n)$$

- Z: Represents the total distance (or cost) of the tour.
- d_{ij}: The distance (or cost) between city i and city j.
- x_{ij}: A binary variable that indicates whether the route travels from city i to city j (1 if yes, 0 if no).

IV. SIMULATED ANNEALING

Simulated Annealing is a probabilistic optimization algorithm inspired by the annealing process in metallurgy, which allows materials to reach a low-energy crystalline state through controlled heating and cooling. By making random moves and accepting worse solutions with a certain probability—reflected in the change in evaluation value ΔE—the algorithm effectively balances exploration and exploitation in complex search spaces. As the algorithm progresses, the likelihood of accepting inferior moves decreases, mimicking the cooling process and enabling the search of both promising and sub-optimal regions, thereby facilitating the discovery of high-quality solutions in complex optimization problems across various fields, including operations research, engineering, and artificial intelligence.

Algorithm:

In this context, ΔE represents the energy difference, and we assess the probability of making a move based on this value within the conditional check. The variable T denotes the temperature, which starts at a high value and gradually decreases through a cooling function. Initially, all moves are equally likely due to this high temperature.

$current \leftarrow INITIAL - STATE$
$T \leftarrow$ some large value
for $t = 1$ to ∞ **do**
  if termination criteria **then**
    break
  end if
  $next \leftarrow$ a randomly selected successor of $current$
  $\Delta E \leftarrow VALUE(next) - VALUE(current)$
  if $\Delta E > 0$ **then**
    $current \leftarrow next$
  else
    $current \leftarrow next$ only with probability $1/(1 + e^{-\Delta E/T})$
  end if
  $T \leftarrow cooling - function(T)$
end for

V. PROBLEM STATEMENTS

1) Given a list of at least twenty important tourist locations in Rajasthan, and assuming the cost of travel between two locations is proportional to the distance between them, use the simulated annealing algorithm to plan a cost-effective tour. The goal is to find a tour that visits each location exactly once and returns to the starting point, minimizing the total travel cost.

Problem Definition:

- **Input:** A graph representing cities as nodes, with the distances between them represented as weighted edges.
- **Output:** A route that goes to each city only one time and comes back to the starting city, while keeping the total travel cost (or distance) as low as possible.

Start-up:

1) Identify 20 significant tourist locations in Rajasthan to serve as nodes in the graph.
2) Set the initial temperature T to a high value (e.g., $T = 1000$).
3) Define a minimum temperature T_{min} to terminate the annealing process (e.g., $T_{\text{min}} = 10$ or $T_{\text{min}} = 102$).
4) Establish a cooling schedule by reducing the temperature by a fraction α after each iteration.
5) Generate an initial tour by creating a random permutation of the selected cities.
6) Calculate the cost (distance) of the initial tour using the distances between the identified cities.

Simulated Annealing

1) Repeat the following steps until the temperature reaches T_{min}:
   a) **Generate Neighbor:** Create a neighboring tour by perturbing the current tour. This may include:
      - Swapping two cities in the tour.
      - Reversing a sequence of cities in the tour.
      - Shifting a subset of cities within the tour.
   b) **Calculate Cost:** Determine the total distance of the new tour.
   c) **Acceptance Probability:** Compute the acceptance probability using the formula:

$$\text{Acceptance Probability} = e^{-\frac{\Delta f}{T}},$$

where Δf is the difference in cost between the new tour and the current tour (new cost - current cost), and T is the current temperature.

   d) **Accept or Reject Neighbor:** Accept the new tour based on the acceptance probability. Always accept the new tour if it has a lower cost than the current tour.
   e) **Update Tour:** If the new tour is accepted, update the current tour to the new tour and adjust the current cost accordingly.
   f) **Cooling:** Decrease the temperature following the established cooling schedule.
2) Track the best tour and its corresponding distance found throughout the iterations.

Solution:

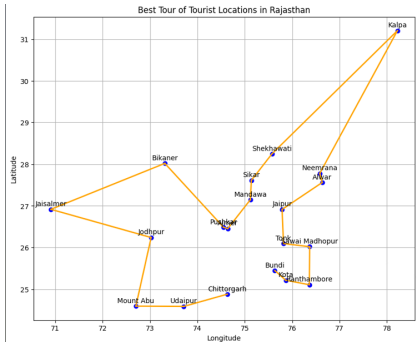

Figure 1: Best Tour Path.

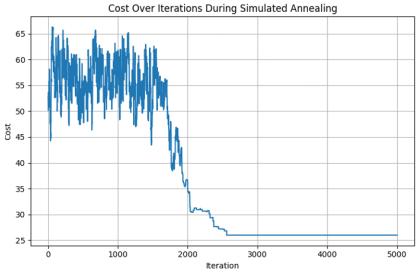

Figure 2: Cost vs Iteration

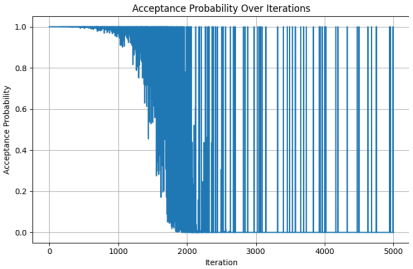

Figure 3: Acceptence Probability.

2) You need to solve a jigsaw puzzle represented as a 512x512 pixel image. The image is divided into smaller pieces and the objective is to rearrange these pieces to reconstruct the original image.

Problem Definition:

- **Input:** The input consists of a file named `scrambled_lena.mat`, which contains the scrambled image data represented as a matrix of pixel values. The image dimensions are 512x512 pixels, and it can be divided into smaller square pieces, such as 64 pieces of 128x128 pixels each. The initial configuration is represented by a random arrangement of the jigsaw pieces, serving as the starting state for the simulated annealing algorithm.
- **Output:** The desired output is the reconstructed image, which should reflect the solved configuration of the original Lena image. This reconstructed image will be properly arranged and can be saved or displayed as a 512x512 pixel image, representing the final state after the simulated annealing process has been completed.

Simulated Annealing in Puzzle Solving:

A. Initial State

The initial state consists of a random arrangement of the tiles. Each tile represents a 128×128 pixel block of the image. This scrambled configuration serves as the starting point for the algorithm.

B. Energy Calculation

The **energy function** quantifies how "disordered" the current arrangement of tiles is. This is computed based on the differences in pixel values between adjacent tiles:

- **Left-Right Energy:** Measures pixel differences along the vertical edges of tiles.
- **Up-Down Energy:** Measures pixel differences along the horizontal edges of tiles.

The total energy is the sum of energy contributions from all tiles, reflecting how well the tiles fit together.

C. Neighbor Generation

Neighboring Solutions: The algorithm generates neighboring configurations by swapping two tiles. This represents a small perturbation in the current arrangement of pixels. Each swap results in a new arrangement of the tiles, leading to a different configuration of the overall image.

D. Acceptance Probability

The algorithm calculates the change in energy (ΔE) after a swap. If the new configuration has lower energy (i.e., a better fit), it is accepted. If the new configuration has higher energy (worse fit), it may still be accepted with a certain probability, which decreases as the algorithm progresses and temperature lowers. This allows for exploration of less optimal configurations to escape local minima.

E. Cooling Schedule

The temperature T starts high, allowing for more random swaps and exploration of the solution space. Over time, the temperature decreases, reducing the likelihood of accepting worse configurations, and focusing more on refining the best found solution.

F. Final State

The process continues until the temperature reaches a predefined minimum (T_{min}), at which point the algorithm stops. The resulting arrangement of tiles represents the reconstructed image, ideally closely resembling the original.

Solution:

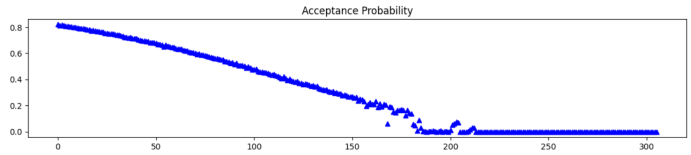

Figure 4: Acceptance Probability.

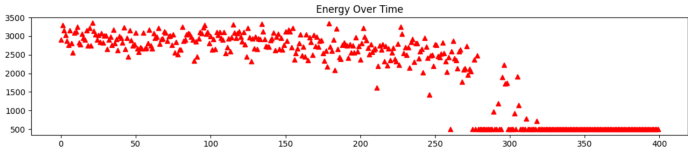

Figure 5: Energy Over Time.

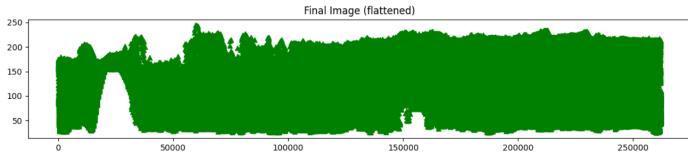

Figure 6: Image Flattened

Figure 7: Solved Puzzle