8-Puzzle Problem Solving Using Simulated Annealing and Genetic Algorithms

Introduction

This project implements two different optimization algorithms, **Simulated Annealing** and **Genetic Algorithm** , to solve the classic ***8-Puzzle problem***. Both algorithms are heuristic-based techniques designed to efficiently search through the problem space, and this repository demonstrates how each can be applied to find a solution for the puzzle.

8-Puzzle Problem

The 8-Puzzle is a sliding puzzle consisting of a 3x3 grid with 8 numbered tiles and one empty space. The objective is to rearrange the tiles starting from a scrambled initial configuration to match the predefined goal state. The blank space is used to slide adjacent tiles, which eventually leads to the solution.

Goal State:

The goal state is represented as follows:

```
1  2  3
4  5  6
7  8  Blank
```

Problem Objective:

- Start with an arbitrary configuration of tiles (initial state).

- Move the tiles by sliding them into the blank space to reach the goal state.

- Minimize the number of moves or the cost to reach the goal.

Simulated Annealing Approach

Simulated Annealing (SA) is a probabilistic optimization algorithm inspired by the annealing process in metallurgy, where a material is heated and slowly cooled to reach a stable state. The algorithm attempts to find a solution by randomly exploring the search space and occasionally accepting worse states to escape local optima.

Key Components:

1. Initial State : The algorithm begins with a random configuration of the puzzle.
2. Objective Function : The Manhattan Distance is used as the heuristic to calculate how far each tile is from its goal position.
3. Temperature : Initially, a high temperature allows the acceptance of worse states. As the temperature decreases, the system becomes more selective, reducing the probability of accepting worse states.
4. Neighbor States : These are generated by making legal moves (sliding tiles) from the current configuration.
5. Acceptance Probability : If the neighbor state is better, it is always accepted. If it is worse, it is accepted with a probability determined by the temperature.

Cooling Schedule:

The temperature gradually decreases, simulating the cooling process. As the temperature lowers, the algorithm is less likely to accept worse solutions, converging to a stable state.

Genetic Algorithm Approach

Genetic Algorithm (GA) is a search heuristic that mimics the process of natural selection. It evolves a population of possible solutions by applying genetic operators such as selection, crossover, and mutation to achieve optimal results over successive generations.

Key Components:

1. Population : A group of random puzzle configurations (individuals) that evolve over time.
2. Fitness Function : The fitness of each individual is determined by the inverse of the Manhattan distance, with a higher fitness indicating a solution closer to the goal.
3. Selection : Individuals with higher fitness are more likely to be selected for reproduction.

4. Crossover : Two individuals (parents) are combined to produce offspring by sharing parts of their configurations.

5. Mutation : Occasionally, a random move is applied to a puzzle configuration to introduce variability and maintain diversity within the population.

6. Termination : The algorithm runs for a predefined number of generations or until a solution is found.

Evolution Process:

1. Initial Population : The algorithm starts with a population of random puzzle states.

2. Fitness Evaluation : Each individual's fitness is calculated using the Manhattan distance.

3. Selection and Reproduction : Fit individuals are selected, and offspring are created through crossover.

4. Mutation : Some offspring are mutated to maintain diversity.

5. Next Generation : The population evolves over several generations, and the algorithm seeks to improve fitness until a solution is reached.

Simulated Annealing Example:

Initial State :

```
1  2  3
4  Blank  6
7  5  8
```

- Heuristic (Manhattan Distance) : The distance each tile is from its goal position is calculated.
- Temperature : Starts high, allowing the algorithm to make random moves (even if worse).
- Cooling : As the temperature drops, the algorithm becomes more selective, refining its solution.

Eventually, the algorithm converges on the goal state after several iterations.

Genetic Algorithm Example:

- Initial Population : A set of random puzzle configurations is generated.
- Fitness Evaluation : The Manhattan distance is used to assess how close each configuration is to the goal.
- Crossover : Two configurations are combined to produce new offspring.
- Mutation : A random move is occasionally applied to keep the population diverse.

After several generations, the algorithm evolves a population that includes the goal state.

Comparison

Criterion	Simulated Annealing	Genetic Algorithm
Search Method	Single solution, probabilistic exploration	Population-based, evolutionary
Exploration	Uses temperature to control exploration	Crossover and mutation ensure exploration
Convergence	Gradually focuses on local refinements	Evolves over generations
Strength	Escapes local optima early with randomness	Parallel search over multiple solutions
Weakness	May take longer to converge	Risk of premature convergence