Project Report on N-Queen Problem Solving using Steepest-Ascent Hill Climbing and Genetic Algorithm

[Introduction]

In this project, we have implemented two algorithms to solve the N-Queen problem, specifically for 8 queens: the Steepest-Ascent Hill Climbing and the Genetic Algorithm. The N-Queen problem involves placing 8 queens on an 8x8 chessboard such that no two queens threaten each other. This problem is a classic example in computer science for studying algorithm efficiency and problem-solving techniques.

[Approach and Implementation]

1. Steepest-Ascent Hill Climbing Algorithm:

- **Concept:** This algorithm starts with a random state and iteratively moves to a neighboring state with a lower heuristic value, where the heuristic is the number of pairs of queens attacking each other.
- **Implementation:** We implemented this algorithm in Python, where we generated 100 random instances of the problem and applied the algorithm. The algorithm stops if it reaches a state where no neighboring state has a lower heuristic value.

2. Genetic Algorithm:

- **Concept:** This algorithm is a search heuristic that mimics the process of natural selection. It uses techniques such as crossover, mutation, and selection to evolve a set of solutions toward an optimal solution.
- **Implementation:** Our implementation involved a population of random states evolving over generations. We employed crossover and mutation operations to create new states and selected the best states based on the heuristic value.

[Analysis]

1. Steepest-Ascent Hill Climbing Algorithm:

- **Results:** The algorithm solved 21% of the instances.
- **Average Cost:** The average heuristic cost was 1.16.
- **Average Time:** The average running time was approximately 0.0035 seconds per instance.
- **Explanation:** The relatively low success rate is due to the algorithm's nature of getting stuck at local maxima. Since it only moves to better states, it fails to find a solution if it reaches a state where no immediate better state exists.

2. Genetic Algorithm:

- **Results:** The algorithm solved 56% of the instances.
- **Average Generations:** On average, it took about 493.89 generations per instance.

- **Average Time:** The average running time was approximately 0.545 seconds per instance.
- **Explanation:** The higher success rate in the genetic algorithm can be attributed to its ability to explore a wider search space and avoid getting stuck in local maxima, unlike the hill-climbing algorithm. The genetic operations of crossover and mutation introduce diversity in the solutions, thereby exploring more possible states.

[Findings and Conclusion]

- **Steepest-Ascent Hill Climbing:** While fast, it is limited by its tendency to get stuck in local maxima. It is efficient for simpler instances but struggles as complexity increases.
- **Genetic Algorithm:** Exhibits a higher success rate and explores a broader search space. However, it requires more computational time and resources.
- **Comparative Efficiency:** The genetic algorithm proved to be more efficient in solving the N-Queen problem than the Steepest-Ascent Hill Climbing algorithm.