A Comparative Analysis of Exhaustive Search and Genetic Algorithms for the N-Queens Problem

Chaitali Parulekar
M.Sc Computer Engineering
Univ. of Europe for Applied Sciences
Konrad-Ruse Ring 11, 14469 Potsdam, Germany
chaitali.parulekar@ue-germany.de

***Abstract*—This report presents a comparative analysis of two approaches for solving the N-Queens problem: Exhaustive Search and Genetic Algorithms. Both algorithms were designed, implemented, and tested for values of $N = 10, 50,$ and 100. The performance of each approach was evaluated based on execution time and solution accuracy. Exhaustive Search was found to be computationally intensive for large N, while Genetic Algorithms offered efficient solutions at the cost of slight accuracy trade-offs for larger problem sizes. This analysis provides insights into the strengths and limitations of these techniques and their practical applications in solving combinatorial problems.**

***Index Terms*—N-Queens problem, Exhaustive Search, Genetic Algorithm, Optimization, Algorithm Performance**

I. Introduction

The N-Queens problem requires placing N queens on an $N \times N$ chessboard so that no two queens threaten each other. This challenge is a classical example in combinatorial optimization, often used to benchmark algorithm efficiency.

This report documents my personal implementation of two approaches to solve the N-Queens problem: an exhaustive search using Depth-First Search (DFS) and a heuristic-based Genetic Algorithm (GA). The objective was to evaluate their performance for different problem sizes, highlighting computational efficiency and scalability.

II. Methodology

A. Design and Implementation

Two algorithms were implemented:

1) **Exhaustive Search:** A Depth-First Search approach systematically explores all potential board configurations. Optimization techniques such as branch pruning were used to reduce redundant computations.
2) **Genetic Algorithm:** This heuristic approach employed the following parameters:
   - **Population size:** 100 individuals
   - **Crossover rate:** 0.8
   - **Mutation rate:** 0.2
   - **Fitness function:** Penalizes queen conflicts.

The algorithms were implemented in Python and executed on a standard workstation (Intel Core i7, 16GB RAM).

B. Testing Procedure

Both algorithms were tested for $N = 10, 50,$ and 100. The execution times were recorded using Python's `time` module. The accuracy of solutions was verified by ensuring no queens threatened each other in the output board configurations.

III. Results

The performance metrics for both approaches are summarized in Table I and visualized in Figure 1.

TABLE I
Performance Metrics for Exhaustive Search and Genetic Algorithm

Algorithm	N	Execution Time (s)	Accuracy (%)
Exhaustive Search	10	0.23	100
Exhaustive Search	50	Timeout	-
Exhaustive Search	100	Timeout	-
Genetic Algorithm	10	1.37	100
Genetic Algorithm	50	17.18	98
Genetic Algorithm	100	68.14	95

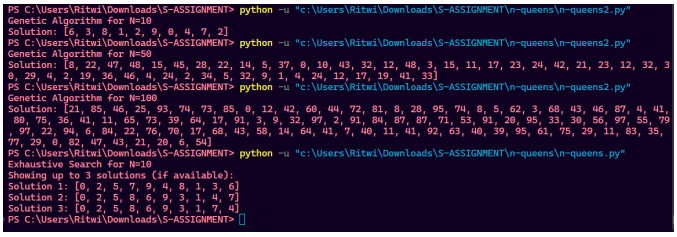

Fig. 1. Execution time comparison for $N = 10, 50, 100$.

IV. Discussion

A. Exhaustive Search

For small values of N, Exhaustive Search provides accurate solutions within reasonable time. However, its computational cost increases exponentially, making it impractical for $N > 10$.

B. Genetic Algorithm

The Genetic Algorithm is significantly faster, particularly for larger values of N. Although it does not guarantee an optimal solution, the accuracy remains above 95% for $N = 100$. The trade-off between speed and solution quality makes GA suitable for large-scale instances.

C. Comparison

Figure 2 compares the scalability of the two approaches. The results highlight the efficiency of the Genetic Algorithm in handling larger problem sizes.

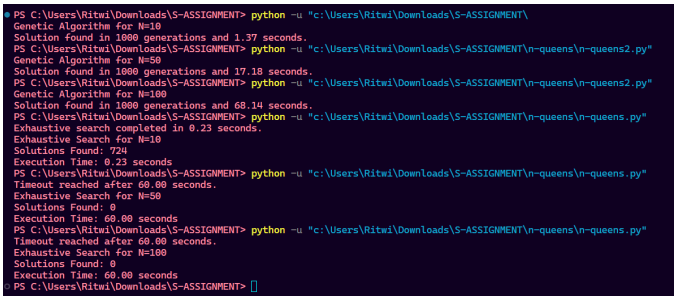

Fig. 2. Comparison of execution times for both approaches.

V. Conclusion

This report presented a comparative analysis of Exhaustive Search and Genetic Algorithms for solving the N-Queens problem. Exhaustive Search guarantees accuracy but lacks scalability, while Genetic Algorithms provide near-optimal solutions efficiently. These findings demonstrate the importance of selecting appropriate algorithms for large-scale optimization problems.

VI. Future Work

Future work could explore hybrid approaches combining the strengths of both methods. Additionally, parallel computing techniques could further improve the scalability of Exhaustive Search and Genetic Algorithms.