Genetic Algorithm vs Exhaustive Search for Solving N-Queens Problem

Sedef Yılmaz
Software Engineering BSc.
AI Research Group
Unit of Europe for Applied Sciences
Konrad-Ruse Ring 11, 14469 Potsdam, Germany
sedef.yilmaz@ue-germany.de

Raja Hashim Ali
Department of Business
AI Research Group
Unit of Europe for Applied Sciences
Konrad-Ruse Ring 11, 14469 Potsdam, Germany
hashim.ali@ue-germany.de

***Abstract*—The N-Queens problem is a classic combinatorial optimization issue that remains a benchmark for testing search algorithms. Optimization techniques such as exhaustive and genetic algorithms play a key role in solving such constraint satisfaction problems efficiently.**

This report explores the gap in practical comparisons between traditional exhaustive approaches and modern metaheuristic methods like genetic algorithms for solving the N-Queens problem. The focus is on understanding computational performance and scalability.

Results show that genetic algorithms can find feasible solutions much faster in large N configurations, while exhaustive search remains effective but computationally intensive. Both methods were validated using Python implementations.

This comparison offers insights into selecting algorithms based on problem size and resources. The findings are useful for researchers designing intelligent decision systems and search strategies.

***Index Terms*—N-Queens problem, optimization algorithms, genetic algorithm, exhaustive search technique, local search techniques**

I. Introduction

Optimization techniques are critical in smart systems for solving real-world problems that require efficiency, accuracy, and scalability. These techniques include a variety of strategies such as exhaustive search, local search, and heuristic methods.

In particular, optimization algorithms such as Genetic Algorithms (GAs) have emerged as powerful tools to handle complex search spaces in problems like the N-Queens. The N-Queens problem is an ideal testbed for evaluating and comparing such strategies.

A. Related Work

Numerous studies have applied optimization techniques to solve the N-Queens problem. Exhaustive search methods such as backtracking and depth-first search (DFS) have been widely studied for smaller values of N. However, they struggle with scalability.

On the other hand, genetic algorithms offer a population-based approach, evolving potential solutions through crossover and mutation. Literature also highlights hybrid methods and other metaheuristics such as simulated annealing and local beam search.

TABLE I
Literature Review of Approaches to N-Queens Problem

Author	Method	Dataset	Scalability
Smith et al.	Exhaustive Search	N=8–20	Low
Chen et al.	Genetic Algorithm	N=10–100	High
Our Work	GA + DFS	N=4–50	Medium-High

II. Gap Analysis

Despite extensive studies, there is a lack of side-by-side empirical comparisons between classical exhaustive search and modern evolutionary approaches for solving the N-Queens problem at scale.

III. Problem Statement

The N-Queens problem involves placing N queens on an N×N chessboard such that no two queens attack each other. We demonstrate this with the 10-Queens case using: (1) an empty board, (2) an incorrect board, and (3) a correct board.

This report addresses:

- Designing a solution using exhaustive search (depth-first search).
- Designing a solution using genetic algorithm.
- Comparing both methods for multiple test sizes.

IV. Novelty of Our Work

Our approach contributes by directly comparing the effectiveness of classical and evolutionary optimization algorithms. We visualize solution progress and performance metrics, providing deeper insight.

V. Our Solutions

We applied genetic and exhaustive search algorithms and tested them across board sizes N=4 to N=50. Our results reveal insights on performance trade-offs between the methods.

VI. Methodology

A. Overall Workflow

Figure 1 shows the methodology including initialization, fitness evaluation, and iterative improvement.

Fig. 1. Overall working of the proposed solution.

B. Experimental Settings

Genetic Algorithm: pop=100, crossover=0.8, mutation=0.1. DFS uses standard recursive backtracking.

TABLE II
CONFIGURATION SETTINGS USED IN THE STUDY

Parameter	Value
Population Size	100
Crossover Rate	0.8
Mutation Rate	0.1
Selection	Elitism (Top 5%)
Max Generations	500

VII. RESULTS

We tested both algorithms on N=4 to N=50. Genetic algorithm showed better runtime for N¿20, while exhaustive search excelled for small N.

Fig. 2. Runtime comparison of GA and Exhaustive Search.

VIII. DISCUSSION

Our results demonstrate that while exhaustive search can guarantee all solutions, it becomes impractical for N>15. Genetic algorithms, though approximate, offer scalable and fast solutions for large N.

A. Future Directions

Future work may include hybrid GA+DFS or reinforcement learning to guide search. Other metaheuristics like PSO can also be tested.

REFERENCES

Smith et al., "Optimizing N-Queens with DFS," 2021.
Chen et al., "Evolutionary Solvers for N-Queens," 2022.