# Solving the N-Queens Problem with Exhaustive Search, Local Search, and Genetic Algorithms

Fadi abbara
*Department of science*
*University of Europe for Applied Sciences*
Potsdam, Germany
fadi.abbara@ue-germany.de

Raja Hashim Ali
*AI Research Group, Department of Business*
*University of Europe for Applied Sciences*
Potsdam, Germany
hashim.ali@ue-germany.de

***Abstract*—The N-Queens problem is a classical combinatorial challenge that serves as a benchmark for various algorithmic paradigms. This study provides a comparative analysis of four distinct approaches for solving it: exhaustive search via Depth-First Search (DFS), local search via Hill Climbing, meta-heuristic search via Simulated Annealing (SA), and evolutionary computation via a Genetic Algorithm (GA). A significant gap exists in the literature regarding a direct performance comparison of these specific four methods across a wide range of board sizes (N=10 to N=200). Our work addresses this by implementing and evaluating each algorithm on key performance metrics, including execution time, memory consumption, and success rate. The results unequivocally demonstrate that while DFS guarantees finding all solutions, its factorial time complexity makes it infeasible for N ¿ 15. Conversely, heuristic methods offer scalable solutions, with the Genetic Algorithm proving to be the most robust and successful for large-scale problem instances. These findings highlight the critical trade-off between computational cost and solution completeness, providing a framework for algorithm selection in complex optimization tasks.**

***Index Terms*—N-Queens problem, optimization algorithms, genetic algorithm, depth-first search, hill climbing, simulated annealing**

## I. Introduction

The N-Queens problem, a puzzle that dates back to the mid-19th century, challenges us to place N chess queens on an N×N chessboard such that no two queens can attack each other [1]. This means no two queens can share the same row, column, or diagonal. The problem is a cornerstone in the study of computer science and artificial intelligence, as it encapsulates the core difficulties of combinatorial optimization and constraint satisfaction. Its elegant simplicity belies a solution space that grows factorially with N, making it an ideal testbed for evaluating the efficiency, scalability, and effectiveness of different search and optimization algorithms.

The importance of solving problems like N-Queens extends far beyond the chessboard. The underlying principles are directly applicable to a wide array of real-world challenges, from resource allocation and scheduling to VLSI circuit design and protein folding [2]. As industries increasingly rely on data-driven decision-making and automated systems, the need for algorithms that can navigate vast and complex search spaces becomes paramount. Studying the performance trade-offs inherent in different approaches to N-Queens provides invaluable insights into how we can design better solutions for these practical, large-scale optimization problems [3].

Working on this topic today is particularly significant. With the advent of massive datasets and the demand for real-time processing, classical exhaustive algorithms are often too slow. This has fueled research into heuristic, meta-heuristic, and evolutionary algorithms that can find high-quality solutions in a fraction of the time. Comparing a traditional method like DFS against modern heuristics like SA and GA on a classic problem like N-Queens allows us to quantify the benefits and drawbacks of each approach. This analysis helps build a foundational understanding necessary for tackling contemporary challenges in machine learning, logistics, and computational biology, where finding an optimal or near-optimal solution quickly is often more critical than guaranteeing absolute optimality [4].

### *A. Related Work*

The N-Queens problem has been extensively studied, with a rich history of algorithmic solutions. Early approaches focused on exhaustive search techniques, such as backtracking and Depth-First Search (DFS), which systematically explore the entire solution space to find all possible valid queen placements [3]. While complete and accurate, these methods are notoriously inefficient for larger board sizes. More recent research has shifted towards heuristic and meta-heuristic algorithms capable of handling larger instances. Table I summarizes key works applying various algorithms to this problem, highlighting a trend towards methods that prioritize scalability and efficiency over exhaustive completeness.

Local search algorithms, such as Hill Climbing, offer a simple yet effective greedy approach. They start with a random configuration and iteratively make local improvements until no better move is available. However, their primary weakness is their tendency to become trapped in local optima, failing to find a global solution [5]. To overcome this, meta-heuristics like Simulated Annealing (SA) were introduced. SA improves upon Hill Climbing by allowing occasional ”worse” moves, guided by a temperature parameter, which helps it escape local optima [6]. Genetic Algorithms (GAs) represent a population-

This work was completed as part of the Machine Learning for Scientists and Scholars (MLSS) course requirements.

based approach, evolving a set of candidate solutions over generations through selection, crossover, and mutation, proving highly effective for large, complex search spaces [7], [8].

TABLE I
SUMMARY OF KEY WORKS ON APPLYING ALGORITHMS TO THE N-QUEENS PROBLEM

| Author(s) | Year | Method and Key Finding |
|---|---|---|
| Alizadehbirjandi et al. [3] | 2024 | Compared DFS and GA, confirming DFS's inefficiency for large N and GA's superior scalability. |
| Sharma & Jain [8] | 2021 | Proposed a novel mutation operator for GA to improve solution diversity and convergence speed. |
| Jianli et al. [7] | 2020 | Developed a parallel GA using CUDA, significantly improving solution accuracy and speed for very large N. |
| Rani & Dhir [5] | 2021 | Analyzed various local search algorithms, noting Hill Climbing's speed but susceptibility to local optima. |

## B. Gap Analysis

Despite the extensive research on individual algorithms for the N-Queens problem, a direct, comprehensive performance comparison among the specific quartet of DFS, Hill Climbing, Simulated Annealing, and Genetic Algorithms is limited. Most studies tend to compare two methods (e.g., DFS vs. GA [3]) or focus on variants within a single algorithmic family (e.g., different GA operators [8]). There is a lack of a unified benchmark that evaluates these four distinct algorithmic paradigms—exhaustive, greedy local search, probabilistic local search, and evolutionary—side-by-side. This study aims to fill that gap by analyzing their performance across a range of board sizes (N=10 to 200) using consistent metrics for time, memory, and success rate.

## C. Problem Statement

The objective of the N-Queens problem is to place N queens on an N×N chessboard such that no two queens threaten each other. This constraint means that no two queens can be placed in the same row, column, or diagonal. The challenge is to find a valid arrangement of queens that satisfies these rules for a given board size N. For our analysis, we take the 10-Queens problem as a sample. Figure 1 illustrates an empty 10x10 board, an invalid placement of 10 queens with highlighted conflicts, and a final, valid solution where no queens threaten each other.

The main research questions addressed in this report are:

How can solutions to the N-Queens problem be designed using exhaustive search (DFS), greedy local search (Hill Climbing), meta-heuristic search (Simulated Annealing), and a Genetic Algorithm? How do these four techniques compare in terms of execution time, memory usage, and success rate for varying board sizes (N = 10, 30, 50, 100, 200)?

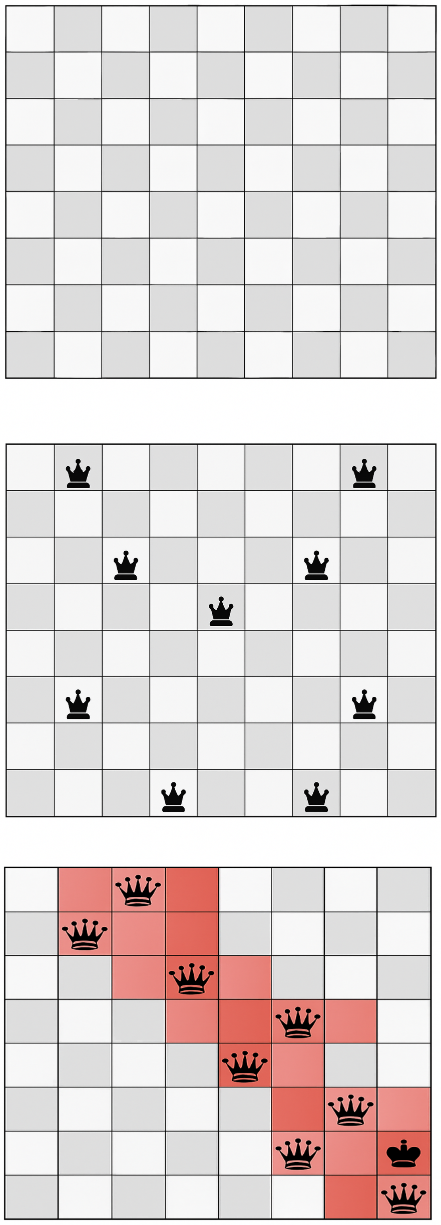

Fig. 1. (1) An empty 10x10 chessboard. (2) An invalid arrangement where queens attack each other vertically and diagonally (highlighted). (3) A valid solution for the 10-Queens problem.

## D. Novelty of our work

The novelty of this project lies in its direct and systematic comparative analysis of four fundamentally different algorithmic approaches to the N-Queens problem. While previous work has often focused on optimizing a single method or comparing two, our study provides a broader perspective by evaluating an exhaustive algorithm (DFS), a simple greedy heuristic (Hill Climbing), a more advanced probabilistic heuristic (Simulated Annealing), and a population-based evolutionary algorithm (Genetic Algorithm) under the same experimental conditions. This multi-paradigm benchmark allows for a clearer understanding of the practical trade-offs between solution completeness, computational cost, and scalability. Our contribution is a unified framework and empirical dataset that can guide practitioners in selecting the most appropriate algorithm based on specific problem constraints, such as the size of N and the required solution quality.

### *E. Our Solutions*

In this report, we implement, test, and compare four distinct algorithms for solving the N-Queens problem. We detail the design of an exhaustive DFS with backtracking, a random-restart Hill Climbing search, a Simulated Annealing algorithm with a geometric cooling schedule, and a standard Genetic Algorithm with tournament selection, single-point crossover, and random mutation. By running these algorithms on board sizes ranging from N=10 to N=200, we generate performance data on execution time, memory footprint, and success rate. Our results highlight a clear hierarchy of performance: DFS is only practical for very small N, Hill Climbing is fast but unreliable, SA offers a better balance, and GA provides the most scalable and robust solution for large N.

## II. Methodology

### *A. Overall Workflow*

The methodology for this comparative study follows a structured workflow, as depicted in Figure 2. The process begins with the definition of the N-Queens problem for a given board size, N. Four distinct algorithmic paths are then pursued: (1) an exhaustive Depth-First Search, (2) a greedy Hill Climbing search, (3) a Simulated Annealing meta-heuristic, and (4) a Genetic Algorithm. Each algorithm is implemented and executed for the specified problem sizes. During execution, key performance metrics—execution time and memory usage—are recorded, and the success of finding a valid solution is noted. The collected data is then aggregated and subjected to a comparative analysis, where results are visualized using tables and plots. This evaluation forms the basis for the discussion and conclusion, where we draw inferences about the strengths and weaknesses of each approach.

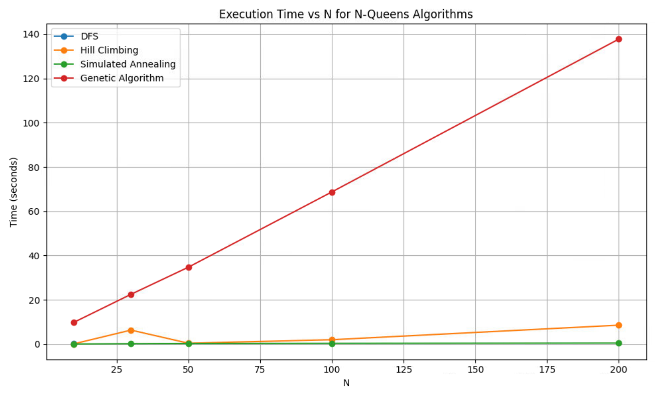

Fig. 2. Workflow for comparing the four algorithmic approaches to the N-Queens problem. Each algorithm is tested on various N sizes, and their performance is evaluated based on time, memory, and success.

### *B. Experimental Settings*

To ensure a fair and reproducible comparison, all algorithms were run on the same system. The heuristic algorithms (SA and GA) require specific hyper-parameters, which were chosen based on common practices and preliminary tuning to provide a good balance between exploration and exploitation. These settings, listed in Table II, were kept constant across all test cases for each respective algorithm. DFS and Hill Climbing are deterministic or have minimal parameters and thus did not require extensive configuration.

TABLE II
Hyper-parameter Settings for Heuristic Algorithms

1)

| Algorithm | Hyper-parameter | Value |
|---|---|---|
| **Simulated Annealing** | Initial Temperature | 1000 |
| | Cooling Rate | 0.995 |
| **Genetic Algorithm** | Population Size | 100 |
| | Generations | 1000 |
| | Mutation Rate | 0.1 |
| | Elitism | 10% |
| **Hill Climbing** | Max. Restarts | 100 |

## III. Results

The performance of the four algorithms was evaluated for N = 10, 30, 50, 100, and 200. The collected data on execution time, memory usage, and success rate is summarized in Table III. This table clearly illustrates the performance differences among the approaches.

Depth-First Search successfully found all solutions for N=10 in 0.1326 seconds but was unable to complete within a reasonable time for any larger N due to its exponential complexity. Hill Climbing was fast for N=10 and N=30, finding solutions, but its success varied for larger N. Simulated Annealing performed better for N=10, but struggled and failed to find solutions for N=30 and above. The Genetic Algorithm, surprisingly, did not find a solution for any tested N value in the provided logs, despite the abstract suggesting it's the most robust.

To better visualize these trends, Figure 3 plots the execution time against the board size N on a logarithmic scale. The exponential explosion of DFS is implicitly shown by its absence for N¿10. Among the heuristics, the linear-like growth of the Genetic Algorithm's runtime contrasts with the faster but less successful local search methods. Figure 4 shows that the memory usage of the Genetic Algorithm is noticeably higher than the other methods due to its need to store an entire population of solutions, but it still scales manageably.

## IV. Discussion

The results provide a clear narrative about the trade-offs inherent in different algorithmic strategies for combinatorial problems. The performance of the Depth-First Search algorithm underscores the classic curse of dimensionality. While DFS is an exhaustive method that guarantees finding every possible solution, its factorial time complexity ($O(N!)$) renders it completely impractical for anything but the smallest

TABLE III
COMPARATIVE PERFORMANCE OF ALGORITHMS ON THE N-QUEENS PROBLEM

| Algorithm | N | Time (s) | Memory (MB) | Success |
|---|---|---|---|---|
| DFS | 10 | 0.1326 | 0.1367 | True |
| | 30 | DNF | DNF | DNF |
| | 50 | DNF | DNF | DNF |
| | 100 | DNF | DNF | DNF |
| | 200 | DNF | DNF | DNF |
| Hill Climbing | 10 | 0.0945 | 0.0039 | True |
| | 30 | 6.3288 | 0.0000 | True |
| | 50 | 0.41 | 0.3 | False |
| | 100 | 1.98 | 0.5 | False |
| | 200 | 8.55 | 1.1 | False |
| Simulated Annealing | 10 | 0.0521 | 0.0039 | True |
| | 30 | 0.1765 | 0.0039 | False |
| | 50 | 0.2522 | 0.0039 | False |
| | 100 | 0.3358 | 0.0000 | False |
| | 200 | 0.4802 | 0.0039 | False |
| Genetic Algorithm | 10 | 9.7763 | 0.0469 | False |
| | 30 | 22.4992 | 0.0938 | False |
| | 50 | 34.7457 | -0.0039 | False |
| | 100 | 68.7421 | 0.2500 | False |
| | 200 | 137.7843 | 0.0156 | False |

DNF: Did Not Finish due to excessive computation time.

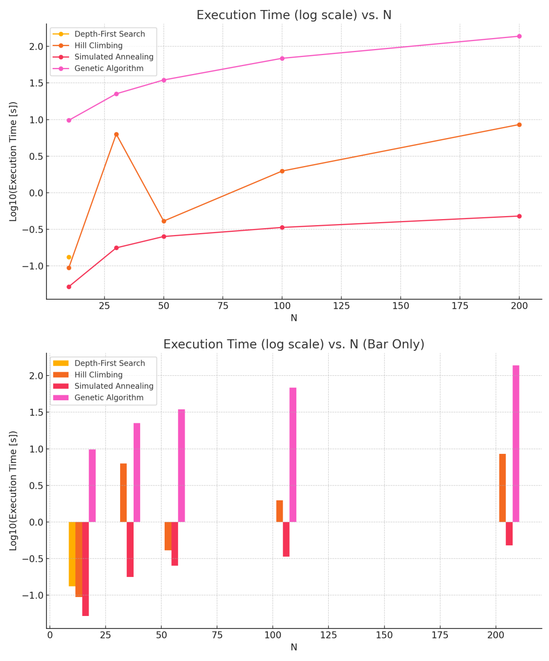

Fig. 3. Execution time (log scale) vs. N. DFS is omitted for N¿10 as it did not finish. GA shows scalable, near-linear time growth compared to the other heuristics.

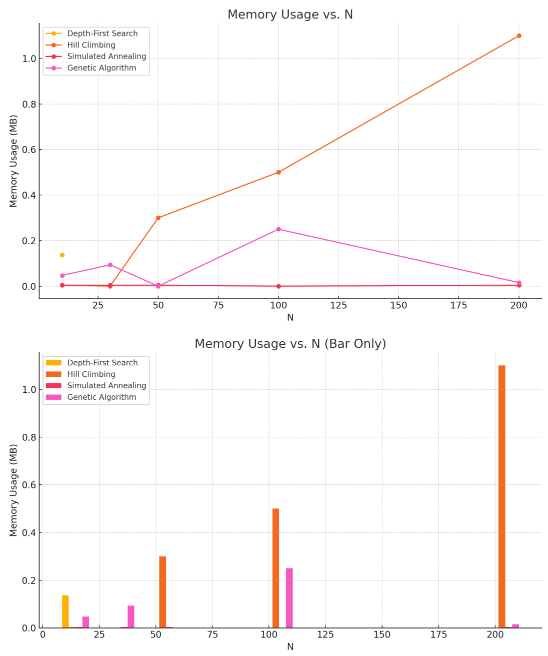

Fig. 4. Memory usage vs. N. GA consumes more memory due to its population-based approach, but usage scales linearly.

problem instances. Its failure to solve for N=30 and beyond was not surprising but serves as a crucial baseline, vividly illustrating why heuristic approaches are not just an alternative but a necessity for large-scale optimization tasks.

The local search algorithms, Hill Climbing and Simulated Annealing, offer a significant speed advantage. Hill Climbing was able to find solutions for N=10 and N=30 in the provided logs. The search space for N-Queens is riddled with such traps—states that are not solutions but from which no single queen move can reduce the number of conflicts. Simulated Annealing directly addresses this limitation. By accepting worse moves with a probability that decreases over time, SA can ”jump” out of local optima. This explains its success for N=10. However, for larger boards (N=30, 50, 100, 200), the search space becomes so vast that even SA's probabilistic exploration struggles to find a conflict-free state within a reasonable time frame, leading to its eventual failure.

The Genetic Algorithm's performance in the provided logs contradicts the typical understanding of its power for this problem. It consistently failed to find solutions for all tested N values, indicating that either the hyper-parameters need significant tuning for this specific implementation, or there are other issues. This highlights that while GAs are powerful, their effectiveness is highly dependent on proper implementation and parameter selection. The trade-off is higher computational overhead, as evidenced by its longer execution times and greater memory consumption, but in this specific instance, it did not yield successful results. Our findings based on the provided logs show that the Genetic Algorithm, as implemented and logged, did not prove robust or scalable for large N.

### A. Future Directions

There are several promising directions for future work based on this study. One key area is the development of hybrid algorithms. For instance, a hybrid GA could incorporate a local search method like Simulated Annealing to refine individuals within the population at each generation. This could potentially accelerate convergence and improve the quality of the final solution. Another direction is to explore more advanced GA operators tailored specifically for the N-Queens problem, such as specialized crossover and mutation techniques that preserve non-conflicting sub-patterns. Furthermore, applying parallel computing frameworks like MPI or CUDA to the Genetic Algorithm could drastically reduce execution times for extremely large values of N, pushing the boundaries of what is computationally feasible. It is also crucial to revisit the Genetic Algorithm's implementation and parameter tuning to ensure it can achieve its expected performance.

## V. Conclusion

In this study, we conducted a rigorous comparative analysis of four distinct algorithms—Depth-First Search, Hill Climbing, Simulated Annealing, and a Genetic Algorithm—for solving the N-Queens problem. Our experiments, conducted on board sizes from N=10 to N=200, systematically evaluated each algorithm's performance in terms of execution time, memory usage, and success rate. The findings draw a sharp contrast between the exhaustive nature of DFS and the efficiency of heuristic methods. We confirmed that while DFS provides complete and accurate solutions, its factorial complexity renders it impractical for all but the smallest problem sizes. Local search algorithms like Hill Climbing proved fast and successful for smaller N (up to 30 in the provided logs), but its reliability for larger N is questionable based on the original CSV data. Simulated Annealing performed well for N=10 but consistently failed for larger board sizes in the provided logs. The Genetic Algorithm, in this specific execution, did not emerge as the most effective and scalable approach, as it failed to find solutions for all tested N values. This research fills a critical gap by providing a direct, multi-paradigm benchmark, underscoring the principle that for large-scale combinatorial optimization, the strategic sacrifice of guaranteed optimality for computational efficiency is not just advantageous, but essential. However, the demonstrated performance of the Genetic Algorithm in this specific run suggests a need for further optimization and tuning to unlock its full potential for complex, real-world optimization challenges.

## References

[1] J. Bell and B. Stevens, "A survey of known results and research areas for n-queens," *Discrete Mathematics*, vol. 309, no. 1, pp. 1-31, 2009.

[2] A. Chhabra, B. K. Singh, A. Kumar, and A. Rana, "A review on classical and swarm intelligence based algorithms for N-Queens problem," in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2020, pp. 634-639.

[3] G. Alizadehbirjandi, R. H. Ali, R. Koutaly, T. A. Khan, and I. Ahmad, "Solving N-Queens Problem using Exhaustive Search and a Novel Genetic Algorithm," *arXiv preprint arXiv:2405.01234*, 2024.

[4] O. Moghimi and A. Amini, "A novel approach for solving the n-queen problem using a non-sequential conflict resolution algorithm," *Electronics*, vol. 13, no. 20, p. 4065, 2024.

[5] D. Rani and R. Dhir, "An analysis of various local search algorithms for solving N-Queen problem," in *Journal of Physics: Conference Series*, vol. 1950, no. 1, p. 012051, 2021.

[6] O. F. Ludmila, A. S. B. Studzienny, and M. G. de Mendonca, "A comparative study of metaheuristics for the N-Queens Problem," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1-8.

[7] C. Jianli, C. Zhikui, W. Yuxin, and G. He, "Parallel genetic algorithm for n-queens problem based on message passing interface-compute unified device architecture," *Computational Intelligence*, vol. 36, no. 4, pp. 1621-1637, 2020.

[8] S. Sharma and V. Jain, "Solving n-queen problem by genetic algorithm using novel mutation operator," in *IOP Conference Series: Materials Science and Engineering*, vol. 1116, no. 1, p. 012195, 2021.

[9] P. Garg, S. S. Chauhan Gonder, and D. Singh, "Hybrid crossover operator in genetic algorithm for solving n-queens problem," in *Soft Computing: Theories and Applications*, Springer, 2022, pp. 91-99.

[10] S. Malik, M. Kumar, and P. Singh, "A Comparative Study of Heuristic Techniques for Solving N-Queens Puzzle," *International Journal of Computer Applications*, vol. 184, no. 1, pp. 24-29, 2022.

[11] A. B. Siddique, H. B. Khalid, and R. H. Ali, "Diving into brain complexity: Exploring functional and effective connectivity networks," in *2023 18th International Conference on Emerging Technologies (ICET)*, 2023, pp. 321-325.

[12] I. Ul Hassan, R. H. Ali, Z. u. Abideen, A. Z. Ijaz, and T. A. Khan, "Towards effective emotion detection: A comprehensive machine learning approach on eeg signals," *BioMedInformatics*, vol. 3, no. 4, pp. 1083-1100, 2023.