CSCI-4511W: Introduction to Artificial Intelligence
Final Project Report
The N-Queens Problem

Chork Hieng

April 27, 2023

Abstract

The N-Queens problem is a classic problem in computer science and mathematics. It's a puzzle game that asks how to place N chess queens on an NxN chessboard so that no two queens threaten each other. In other words, no two queens can occupy the same row, column, or diagonal. The problem is NP-complete, meaning that there is no known polynomial time solution for all values of N, and the most efficient known algorithms have exponential time complexity. The problem has applications in fields such as artificial intelligence, computer vision, and cryptography.

Project Description

In this project, We will attempt to solve N-Queens problems with different approaches. The N-Queens problem was introduced in 1850 by **Carl Gauss** [7]. This problem is an interesting puzzle that is significant practice to find Constraint Satisfaction Problem. It's represented as a puzzle that is an NxN chessboard with N queens such that one queen can't attack other queens [7]. A queen can move vertically, horizontally, or diagonally on the grids without attacking one another. This problem can be easy for small chessboard like 3x3 or 4x4, but it becomes more difficult as the number of queens within the problem becomes larger [4]. The effort to solve larger N-Queens problem is growing exponentially, which needs expensive computation to solve [4]. This project will explore more capability of how to solve this problem using different approaches and introduce solution to larger N-Queens problem. Because solving this problem can be computationally expensive, there will be varieties of algorithms present to attempt finding the solution, which is effective to do comparison and conclusion of the project.

Introduction

The N-Queens problem is a well-known problem in computer science, which aims to place N queens on an NxN chessboard in such a way that no two queens can attack each other [2]. A queen can move vertically, horizontally, or diagonally on the grids without attacking one another. This problem can be solvable by human for small chessboard like 3x3 or 4x4, but it becomes more difficult as the number of queens within the problem becomes larger [4]. The N-Queens problem was first proposed in 1848 by the chess player **Max Bezzel** and has since been studied by researchers in the field of computer science and mathematics [3]. The N-queens problem has a rich history, and many mathematicians and computer scientists have contributed to its development. After the problem was first proposed by **Max Bezzel** in 1848, in 1850, **Franz Nauck** published the first solution to the problem for size of 8x8 grids. Since then, many researchers have attempted to solve the problem for various values of N, which is the size of the grids. In the early days, researchers used brute force and backtracking algorithms to solve the problem. However, with the advent of heuristics, researchers have developed more efficient solutions for the N-Queens problem [8].

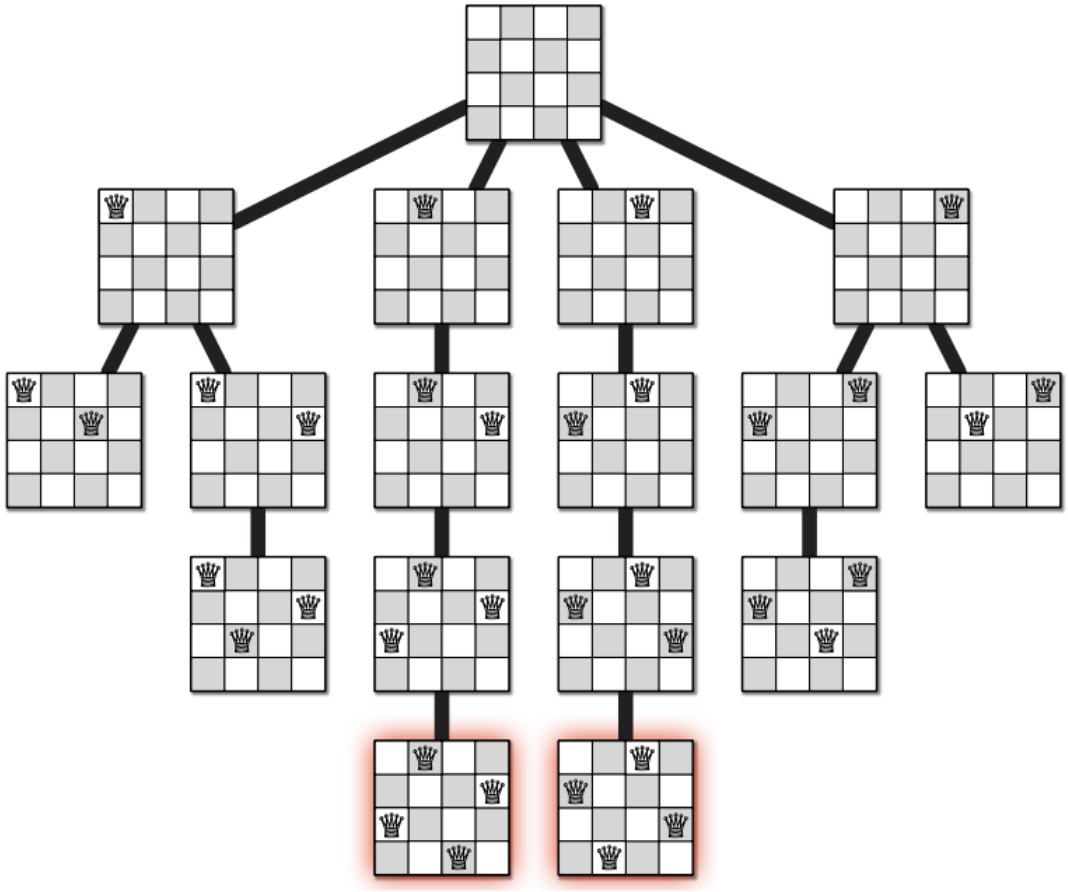


Figure 1: The illustration of problem with size of 4x4 grids.

Background Work

An early approach to solve the N-Queens problems was the brute force approach, which gave the possibility to solve an 8x8 grids in about six minutes on computer algorithms [1]. The algorithm starts by placing a queen in the first column of the first row, then places a queen in the second column of the second row, and so on until all queens are placed on the board without attacking each other. Brute force search involves trying all possible arrangements of queens on the board and checking if any two queens are attacking each other. This method is guaranteed to find a solution if one exists, but the time complexity grows exponentially as the size of the board becomes larger. This was reasonable at the time when approach was proposed to solve the N-Queens problem, but later researches has provided better performance and computational cost.

Regarding to later research, backtracking was introduced as a more efficient way to solve the N-queens problem [10]. Backtracking is a general algorithmic technique that involves trying out different solutions and backtracking when the solution does not meet the required conditions. The backtracking approach can be applied to the N-queens problem by placing queens on the board row by row and checking if the placement of each queen is safe. If a queen cannot be placed in a row without attacking another queen, the algorithm backtracks to the previous row and tries a different position for the previous queen. For example, if a queen in the second row cannot be placed in any of the columns without attacking the queen in the first row, the algorithm backtracks to the first row and tries a different position for that queen [10]. This process continues until a solution is found or all possibilities are exhausted. Backtracking approach uses recursion, so it's considered as faster, comparing to brute force approach.

Additionally, the backtracking method is also an algorithmic technique that incrementally builds a solution to a problem and abandons a solution as soon as it determines that the solution cannot be completed [10]. The backtracking algorithm used for the N-Queens problem works by placing a queen in a column of the first row and then recursively attempting to place queens in subsequent rows such that they do not attack each other. If a queen cannot be placed in a particular row without attacking another queen, the algorithm backtracks and tries a different position for the queen in the previous row [10].

In addition to backtracking approach, there are also other algorithms that can be used to solve the N-Queens problem. One such algorithm is the depth-first backtracking algorithm, which is a type of brute-force search. The depth-first backtracking algorithm explores all possible solutions to the problem by recursively constructing partial solutions and backtracking when a dead end is reached [10]. Although this algorithm is guaranteed to find a solution if one exists, it can be very inefficient for large values of N due to the large number of partial solutions that must be explored. This can be computationally expensive in terms cost of time to find a solution.

However, backtracking is not enough for the research in trying to find technique for solving N-Queens problem. It is still considered slow for the very large problems. In later research, heuristic value approach was proposed as another way to improve technique in solving N-Queens problem [8]. Solving the N-Queens problem using heuristic approach typically involve

assigning a heuristic value to each board configuration, which represents how appropriate that configuration is in terms of how many queens are attacking each other. The goal is then to find the board configuration with the lowest heuristic value, which represents the best solution to the problem.

One common heuristic approach is the "minimum conflicts" algorithm. In this approach, the method starts with a random initial board configuration and then repeatedly move a single queen to a new row within its column, choosing the row that results in the lowest heuristic value [8]. This process is repeated until a solution is found or a maximum number of iterations is reached.

The heuristic value used in the minimum conflicts algorithm is based on the number of queens that are attacking each other, where a lower value indicates a better configuration. Specifically, for each queen on the board, we count the number of other queens in the same row, column, and diagonal as that queen, and add up these counts to get the total number of conflicts [8]. The heuristic value for the board is then the sum of the number of conflicts for each queen on the board.

Another method was proposed, which uses a parallel genetic algorithm that involves dividing the problem into smaller sub-problems and solving them simultaneously using multiple processors. The algorithm works by generating a population of potential solutions and evaluating their fitness based on how many queens are threatened by one another [9]. The fittest solutions are then selected for reproduction, where their genetic material is combined to create new potential solutions. The experiments were conducted to compare the performance of this approach to other existing methods for solving the N-Queens problem. The results showed that the parallel genetic algorithm outperformed other methods in terms of both solution quality and execution time [9].

In addition to the proposed genetic algorithm, improvement was also made, which uses an advanced mutation operator that can generate solutions to the N-queens problem with high accuracy. The searchers compared the performance of their algorithm to other genetic algorithms that have been proposed for this problem. The experiment demonstrates the effectiveness of genetic algorithms in solving the N-queen problem and shows the potential benefits of using advanced mutation operators in these algorithms [5].

Recent research discusses a parallel genetic algorithm for solving the N-Queens problem using message passing interface-compute unified device architecture, a parallel genetic algorithm that can efficiently solve this problem by distributing the workload across multiple processors. The experiment shows that this proposed algorithm can produce high-quality solutions to the N-Queens problem and is efficient for solving other challenging combinatorial optimization problems [6].

Approaches Used in this Experiment

There are four approaches used in this experiment, and the graph used to analyze the data only contains the two most efficient approaches. This experiment compares and contrasts four different approaches for solving the N-Queens problem by comparing the run time in

finding solutions.

Brute Force Approach

Brute force is a very simple way to explore all the possible solutions to the problem, and it's guaranteed to find the solution if the solution exists. This approach is considered slow but is a very good way to start with such a small problem.

Backtracking Approach

Backtracking is a faster method, compared to brute force, and it's considered as an improvement of computation from brute force approach. Backtracking uses recursive method, which backtracks to the previous level if there is no possible solution in current level. The process keeps backtracking until the solution is found, enabling faster solution-finding for moderately complex problems.

Heuristic Value Approach

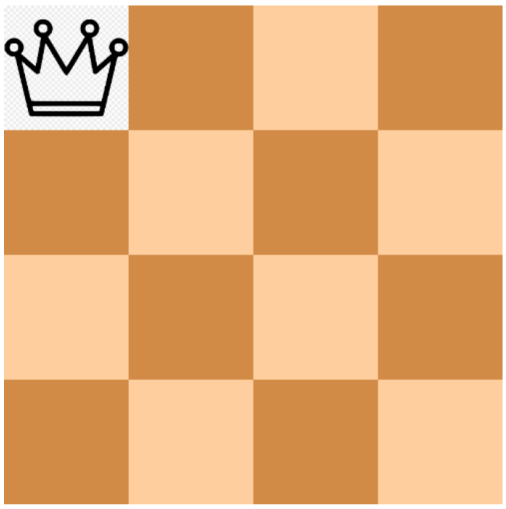
Heuristic value based method is considered one of the most efficient approaches to solve N-Queens problem, and it can find the solution in faster time, compared to other algorithms. This method works by having an estimate value for the solution and choose the first solution found. The solution is considered near-optimal, which is an efficient approach to larger sizes of the problem. The heuristic method incorporates domain-specific knowledge, allowing it to prune the search space and reduce the number of possibilities it needs to explore.

Genetic Algorithm

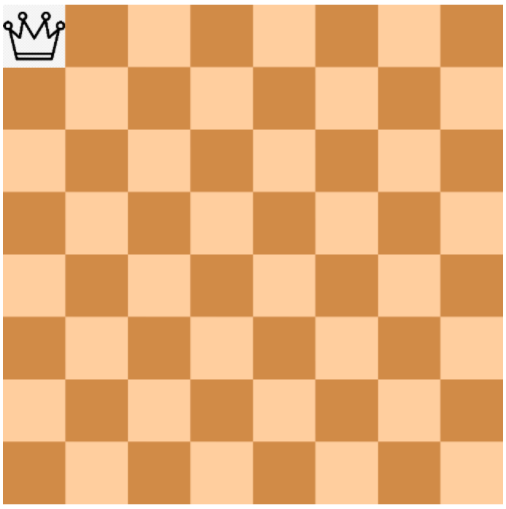
The genetic algorithm is a fascinating approach rooted in the principles of natural selection and evolution. It generates different sets of solutions (populations) and uses evolutionary operators such as selection, crossover, and mutation to evolve the solutions over multiple generations. The algorithm iteratively refines the solutions, ultimately converging on the best one. Although this approach is slower than the heuristic method when dealing with a large number of queens, it still offers valuable insights and can provide globally optimal solutions in some cases.

Experimental Design

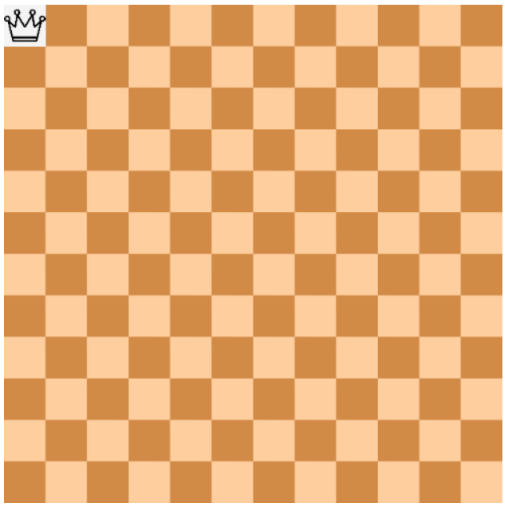
The aim of this study is to compare the effectiveness of four different algorithms in solving N-Queens problems. Specifically, we will investigate the performance of brute force, backtracking, heuristic value approaches, and genetic algorithm. The study will involve running these four algorithms on various N-Queens problems of different sizes, ranging from small to large. To conduct this study, we will first select a set of N-Queens problems of varying sizes. The sizes will be chosen based on the computational complexity of the problem, with smaller sizes serving as a baseline and larger sizes testing the limits of each algorithm. The performance of each algorithm will be measured in terms of the time it takes to find a solution, as well as the quality of the solution found. We will use standard performance metrics, such as run time and accuracy, to assess the effectiveness of each algorithm. Additionally, we will examine the relationship between problem size and algorithm performance. To ensure the reliability and validity of the results, we will run each algorithm on each problem multiple times and take the average performance as the final result. We will also use statistical analysis to determine whether there are significant differences in performance between the three algorithms. The time comparison of each algorithm is based on the implementation from a GitHub repository (`https://github.com/Chorkhieng/N-Queens-Problems-Algorithms/tree/main`). This link is part of the experiment for solving N-Queens but only the graphic picture is used to illustrate the various sizes of the board (`https://chess.akdev.ir/`).



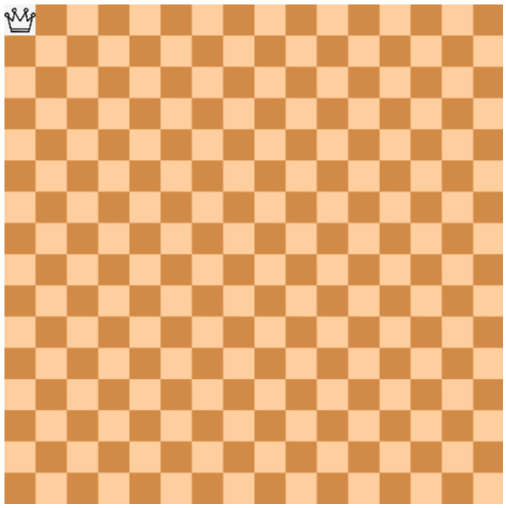
(a) A problem with 4 queens (4x4)



(b) A problem with 8 queens (8x8)



(c) A problem with 12 queens (12x12)



(d) A problem with 16 queens (16x16)

Figure 2: Levels of difficulty increase as the size increases.

Results of the Experiment

Results Using Brute Force Approach (Time Measured in Seconds):

Size of the Board			
4 by 4	8 by 8	12 by 12	16 by 16
4.53E-05	0.045	>100	>100
4.39E-05	0.092	>100	>100
6.87E-05	0.033	>100	>100
4.88E-05	0.035	>100	>100
4.46E-05	0.035	>100	>100
7.53E-05	0.032	>100	>100
4.44E-05	0.034	>100	>100
4.42E-05	0.042	>100	>100
6.08E-05	0.157	>100	>100
6.83E-05	0.138	>100	>100
6.80E-05	0.279	>100	>100
4.44E-05	0.101	>100	>100
6.22E-05	0.204	>100	>100

Table 1: Trials 1-13

Size of the Board			
4 by 4	8 by 8	12 by 12	16 by 16
4.43E-05	0.046	>100	>100
4.59E-05	0.095	>100	>100
6.08E-05	0.081	>100	>100
4.09E-05	0.06	>100	>100
4.49E-05	0.045	>100	>100
7.03E-05	0.047	>100	>100
4.45E-05	0.048	>100	>100
4.34E-05	0.046	>100	>100
5.01E-05	0.161	>100	>100
6.03E-05	0.053	>100	>100
6.96E-05	0.045	>100	>100
4.89E-05	0.053	>100	>100
7.01E-05	0.069	>100	>100

Table 2: Trials 14-26

Results Using Backtracking Approach (Time Measured in Seconds):

Size of the Board			
4 by 4	8 by 8	12 by 12	16 by 16
7.30E-05	0.012	6.823	>100
3.39E-05	0.011	8.407	>100
3.36E-05	0.012	7.185	>100
1.02E-04	0.007	6.134	>100
5.77E-05	0.010	7.712	>100
3.43E-05	0.013	7.597	>100
3.43E-05	0.062	6.471	>100
3.44E-05	0.012	7.444	>100
3.41E-05	0.007	6.128	>100
6.56E-05	0.018	8.885	>100
5.44E-05	0.007	7.814	>100
3.39E-05	0.011	7.842	>100
4.67E-05	0.011	6.235	>100

Table 3: Trials 1-13

Size of the Board			
4 by 4	8 by 8	12 by 12	16 by 16
1.07E-04	0.009	6.613	>100
3.41E-05	0.010	5.954	>100
3.31E-05	0.015	6.046	>100
3.46E-05	0.011	5.785	>100
9.66E-05	0.011	6.804	>100
3.86E-05	0.014	5.965	>100
8.82E-05	0.011	6.427	>100
3.55E-05	0.007	6.541	>100
4.96E-05	0.011	7.161	>100
8.97E-05	0.013	9.312	>100
3.70E-05	0.007	7.876	>100
3.41E-05	0.007	6.995	>100
7.61E-05	0.07	6.894	>100

Table 4: Trials 14-26

Results Using Heuristic Value Approach (Time Measured in Seconds):

Size of the Board			
4 by 4	8 by 8	12 by 12	16 by 16
0.0048	0.088	2.095	1.764
0.0045	0.082	0.597	2.381
0.0068	0.089	0.558	1.855
0.0208	0.099	0.548	2.846
0.0073	0.106	0.887	1.500
0.0074	0.088	0.716	3.064
0.0066	0.084	1.789	1.480
0.0101	0.104	0.645	1.577
0.0046	0.079	0.535	2.541
0.0043	0.106	0.546	2.327
0.0069	0.087	0.557	1.525
0.0077	0.083	0.577	1.597
0.0303	0.099	0.648	1.692

Table 5: Trials 1-13

Size of the Board			
4 by 4	8 by 8	12 by 12	16 by 16
0.0078	0.084	0.528	2.747
0.0047	0.084	0.669	1.493
0.0047	0.087	0.782	1.726
0.0048	0.085	0.601	1.624
0.0042	0.084	0.537	1.833
0.0047	0.079	0.646	1.666
0.0043	0.085	0.544	1.554
0.0046	0.101	0.517	1.493
0.0058	0.085	1.941	1.571
0.0045	0.078	0.588	1.818
0.0047	0.085	0.601	1.667
0.0043	0.104	0.377	1.560
0.0045	0.084	0.418	1.493

Table 6: Trials 14-26

Results Using Genetic Algorithm (Time Measured in Seconds):

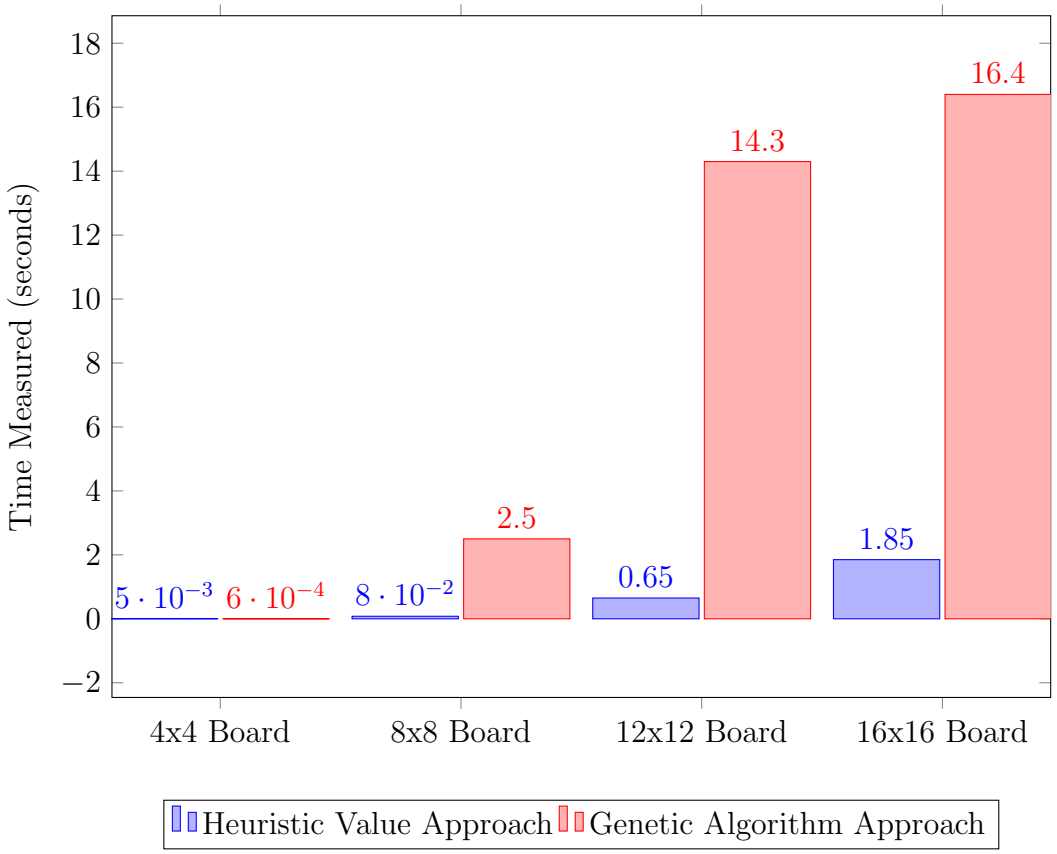
Size of the Board			
4 by 4	8 by 8	12 by 12	16 by 16
4.38E-04	3.384	11.316	10.633
4.96E-04	2.529	11.950	9.976
6.76E-04	2.875	10.949	10.816
5.3E-04	3.272	12.276	17.012
7.15E-04	2.247	14.441	15.520
4.27E-04	2.961	17.638	10.756
5.05E-04	2.863	1.534	17.634
4.19E-04	3.163	7.241	12.346
4.88E-04	0.0012	6.256	15.817
6.01E-04	0.139	9.536	14.071
6.46E-04	2.978	12.015	13.611
8.32E-04	0.082	6.293	21.314
6.76E-04	3.048	9.472	12.014

Table 7: Trials 1-13

Size of the Board			
4 by 4	8 by 8	12 by 12	16 by 16
5.58E-04	0.11	9.443	12.786
5.73E-04	3.319	16.524	19.860
9.27E-04	0.059	12.608	17.752
9.13E-04	2.845	13.852	10.211
6.45E-04	2.976	6.562	10.043
5.7E-04	0.007	16.575	11.914
8.34E-04	2.920	16.382	13.389
5.88E-04	3.203	8.473	16.967
5.83E-04	0.451	6.219	12.952
6.28E-04	0.11	6.417	9.518
5.45E-04	0.016	5.656	10.552
5.72E-04	4.195	7.339	9.637
5.89E-04	3.117	5.608	10.393

Table 8: Trials 14-26

Size of the Board (NxN) vs. Time Measured (seconds):



Data Analysis

In this experiment, we compared the performance of four different approaches to solve the N-Queens problem: brute force, backtracking, heuristic value, and genetic algorithms. We measured the run time and number of solutions found for each approach across a set of randomly generated input instances with varying values of N. This data analysis showed that the heuristic value approach is the fastest among the four approaches, with the lowest mean run time and standard deviation. The brute force and backtracking algorithms approaches had the highest run time and exhibited exponential growth as N increased. Backtracking approach had a lower run time than the brute force approach but still grew significantly as N increased. The heuristic value approach uses a heuristic function to guide the search towards promising solutions and avoid exploring unpromising parts of the search space, resulting in fastest run time. Overall, the results indicate that the heuristic value approach is the most efficient approach for solving the N-Queens problem in terms of time complexity.

Conclusion

In conclusion, the N-Queens problem is a challenging problem that has inspired many researchers to develop new algorithms and techniques. There have been many approaches improved over time, and the choice of those approaches depends on the size of the board and the desired level of accuracy. As until now, many researchers still continue to study and find better methods to solve the N-Queens problem, and it remains an important problem in the field of computer science and mathematics. The problem can be solved by brute force, backtracking, heuristic-based algorithms, and many other methods, and each method has its own advantages and limitations over the size of the problem. Although the brute force and backtracking approaches were effective in the early days, they become computationally expensive for larger problems. Heuristic-based algorithms, such as the minimum conflicts algorithm, are more efficient and can be used to find a solution in a shorter amount of time. Other algorithms, such as the depth-first backtracking algorithm, are guaranteed to find a solution if one exists, but they can be very inefficient for large values of N. The problem continues to be an active area of research, with new algorithms and techniques being developed to improve its solution.

References

[1] Alejandro Arteaga, Ulises Orozco-Rosas, Oscar Montiel, and Oscar Castillo. Evaluation and comparison of brute-force search and constrained optimization algorithms to solve the n-queens problem. volume 1050. Springer, 2022.

[2] Jason Bell and Brett Stevens. A survey of known results and research areas for n-queens. *Discrete Mathematics*, 309(1):1–31, 2009.

[3] Sabine Glock, Duarte Munhá Correia, and Benny Sudakov. The n-queens completion problem. *Research in Mathematics Sciences*, 9(1):41, 2022.

[4] E. J. Hoffman et al. Construction for the solutions of the m queens problem. *Mathematics Magazine*, pages:66–72, 1969.

[5] Vaishali Jain and Jyoti Prasad. Solving n-queen problem using genetic algorithm by advance mutation operator. *International Journal of Electrical and Computer Engineering*, 8(6):4519–4523, 2018.

[6] Chen Jianli, Chen Zhikui, Wang Yuxin, and Guozheng He. Parallel genetic algorithm for n-queens problem based on message passing interface-compute unified device architecture. *Computational Intelligence*, 36(4):1621–1637, 2020.

[7] Craig Letavec and John Ruggiero. The n-Queens Problem. University of Dayton.

[8] Ivana Martinjak and Marin Golub. Comparison of heuristic algorithms for the n-queen problem. In *2007 29th International Conference on Information Technology Interfaces*, pages 759–764. IEEE, 2007.

[9] Monire Taheri Sarvetamin, Amid Khatibi, and Mohammad Hadi Zahedi. A new approach to solve n-queen problem with parallel genetic algorithm. *Journal of Advances in Computer Engineering and Technology*, 4(2):69–78, 2018.

[10] Ryuhei Uehara. *Backtracking*. Springer, Singapore, 2019.