Faculty of Computers and Artificial Intelligence

Fall semester 2023-2024

Artificial Intelligence project

Team Members

Name	ID	Department	Level
محمد مصطفي محمود عبدالمجيد	20210837	CS	3
يوسف محمد نادي حسن	20211104	CS	3
محمد عبدالرزاق فؤاد	20210807	CS	3
محمد أيمن محمد قنديل	20210751	CS	3
احمد رضا شعبان عبد الوهاب	20210050	CS	3
محمد ناصر رفعت على	20210841	CS	3
محمد عوض الله سري احمد	20210822	CS	3

Code & Documentation link:

https://github.com/muhammed-elsherif/N-Queens-multiple-alg

Introduction and Overview:

-Project idea and overview:

Abstract → The N-Queens Problem Solver project aims to develop an intelligent system capable of solving the N-Queens problem for various board sizes.

The N-Queens problem is a classic chessboard puzzle where the objective is to place N queens on an N×N chessboard in such a way that no two queens threaten each other. Threatening means no two queens share the same row, column, or diagonal.

The value of N, representing the board size and the number of queens, can be chosen by the user.

User Interaction:

The user has the flexibility to choose the size of the chessboard (N), making the project adaptable to different scenarios.

The system will present solutions generated by each algorithm, allowing the user to compare their effectiveness and efficiency.

Example:

User Input:

Board size (N): 8

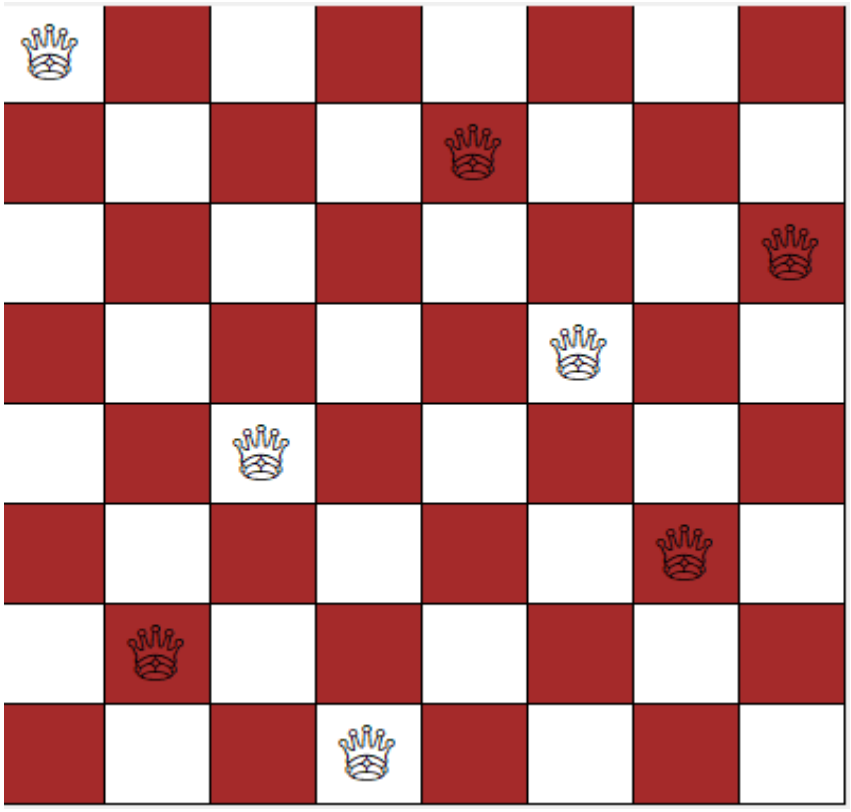
OR

Number of queens: 8

Output:

Solution found: Yes

The chessboard is an 8x8 grid.



- Similar applications in the market:

Parallel memory storage schemes.

VLSI testing.

Traffic control.

Resource Allocation and Scheduling.

Deadlock prevention.

-Functionalities/Features:

-Random Board Generation: The system can generate random initial configurations of queens on the chessboard.

-User Input: Allow users to specify the board size and the number of queens they want to place.

-Solving Algorithm: Implement an intelligent solving algorithm to find a valid solution for the N-Queens problem.

-Solution Visualization: Display the final solution on the chessboard, highlighting the positions of the queens.

-Multiple Solutions: Provide the option to find and display multiple valid solutions if they exist.

-Performance Metrics: Calculate and display metrics such as the time taken to find a solution and the number of backtracks performed.

-A Literature Review of Academic publications:

https://betterprogramming.pub/backtracking-n-queen-problem-and-sudoku-31974988bcb0

https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://arxiv.org/ftp/arxiv/papers/1802/1802.02006.pdf#:~:text=In%201850%20Franz%20Nauck%20gave,problem%20is%20O(n!).

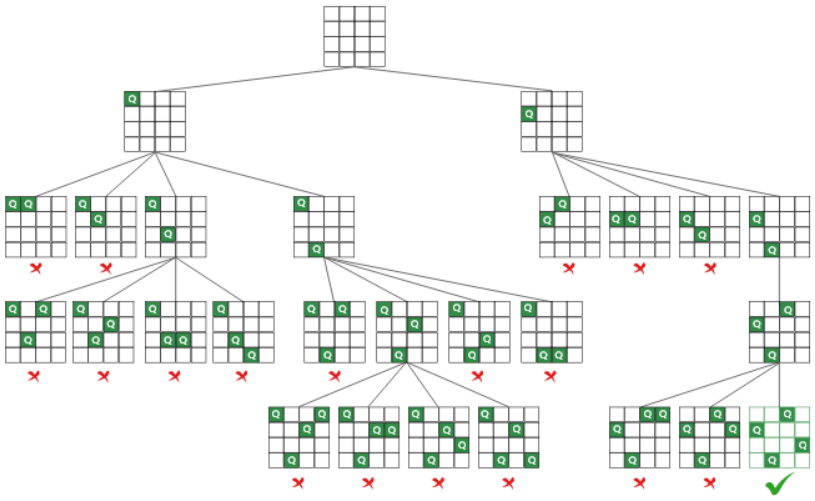
1.1 Algebraic Definition

Let A be the set of integers such that A = {0,1,2,..., N-1}. Define B as the Cartesian product of A, i.e., B = A*A. Then, B = {(0,0),(0,1),..., (1,0),..., (N-1,0),..., (N-1,N-1)}. Select N elements of B and call this set as C. The set C represents a solution to the problem if for the Cartesian product of C, C*C, each element $((i_k, j_k), (i_l, j_l))$ either satisfies all of the following conditions or fails all of them (for all l, k = 0, 1,..., N-1):

1. $i_k \neq i_l$ {not on the same column},
2. $j_k \neq j_l$ {not on the same row},
3. $i_k + j_k \neq i_l + j_l$ {not on the same diagonal},
4. $i_k - j_k \neq i_l - j_l$ {not on the same diagonal}.

5.2 Backtracking Algorithms

The general characteristics of backtracking algorithms have been discussed earlier. These algorithms can be modified to provide only one solution. These algorithms find the lexiographically first solution to the N-Queens problem. Empirical results show that the run-times for the backtracking algorithms which find only the first solution, increase exponentially [30].



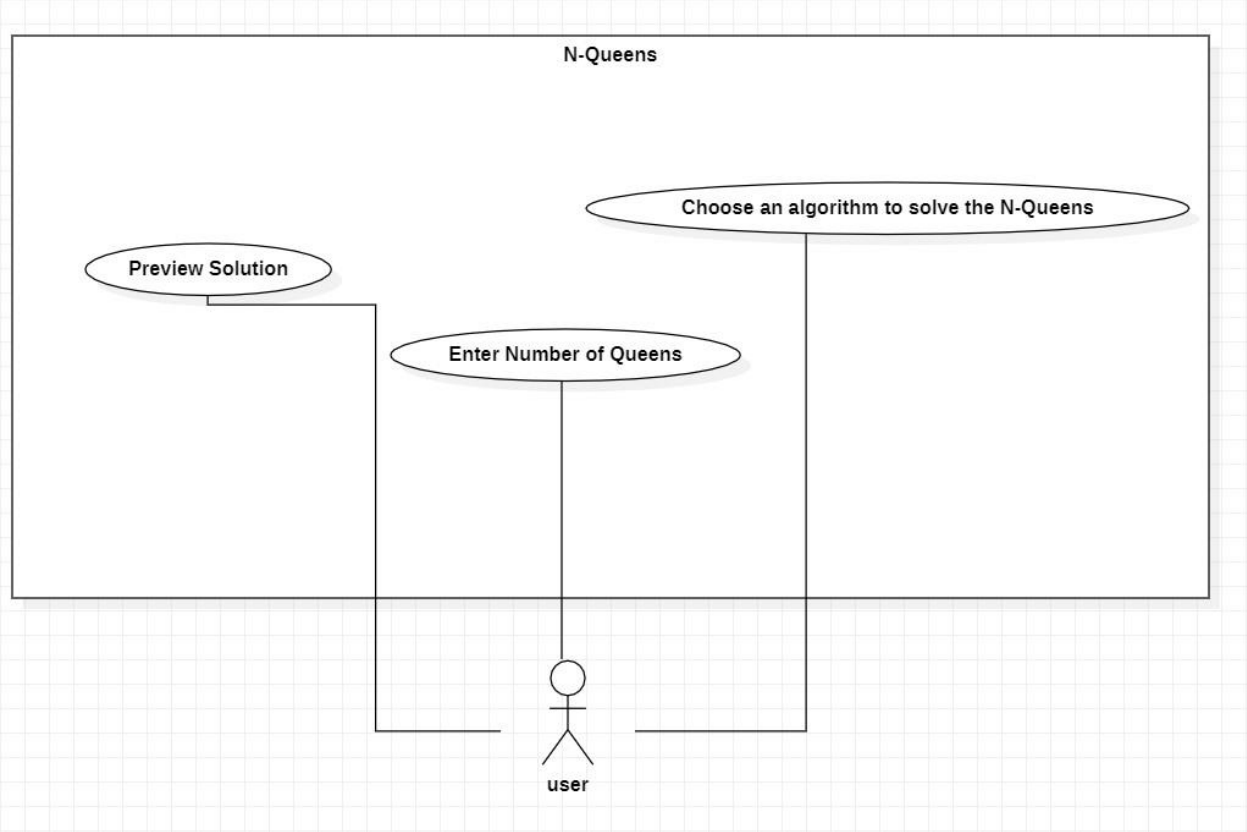
Backtracking

N	Iterations	Optimal Solution																										
25	1431	22	6	11	14	24	15	19	8	4	17	1	3	12	16	7	21	18	2	25	23	10	20	5	9	13		
	382	12	4	16	23	11	24	22	20	2	5	15	10	7	19	3	8	17	25	6	1	18	13	9	14	21		
	281	17	5	22	8	12	21	1	13	20	16	19	24	11	23	4	7	3	15	9	2	14	10	18	25	6		
24	4043	6	8	15	17	14	10	23	18	24	2	4	11	7	3	16	13	19	9	22	5	21	12	1	20			
	392	13	3	22	6	18	2	23	11	4	14	17	19	24	8	20	5	1	15	10	7	21	12	9	16			
	420	6	14	21	17	24	16	1	7	4	10	3	18	15	23	11	19	2	5	8	13	22	20	9	12			
23	1363	5	14	22	3	8	13	21	4	6	23	19	2	18	7	17	11	9	16	10	20	15	1	12				
	113	18	16	21	17	3	6	13	1	4	7	5	22	14	19	11	15	8	20	9	23	2	10	12				
	347	4	16	14	23	1	10	22	15	11	5	2	18	21	8	13	17	7	3	19	6	20	9	12				
22	282	8	13	1	9	14	19	4	22	15	12	6	11	21	18	3	17	20	10	2	16	5	7					
	2044	13	18	12	5	8	1	11	16	2	6	21	9	15	19	22	3	17	4	7	10	20	14					
	1082	18	3	12	2	16	5	11	1	15	20	6	4	9	19	17	10	14	7	22	8	21	13					
21	404	6	11	16	10	15	21	3	1	9	14	5	19	2	12	18	7	4	17	20	8	13						
	807	7	19	13	16	12	5	3	1	6	2	11	20	14	17	4	18	21	8	10	15	9						
	592	16	5	13	21	7	18	11	3	1	4	10	17	19	12	15	6	14	9	20	8	2						
20	453	12	7	11	3	20	18	14	4	6	8	5	15	17	9	2	16	19	10	1	13							
	279	9	17	1	10	2	16	18	8	12	3	15	6	20	13	5	7	19	14	11	4							
	170	4	16	9	6	1	17	20	5	13	11	18	2	7	12	15	3	8	10	14	19							
19	3166	1	5	9	13	17	8	12	15	6	3	19	4	14	18	10	2	11	16	7								
	1462	2	15	9	11	13	8	1	3	19	10	14	17	6	18	12	5	16	4	7								
	382	8	5	2	13	9	16	12	19	17	6	4	1	7	14	10	18	15	3	11								
18	5730	12	3	11	13	5	9	1	4	7	16	10	17	15	18	8	2	14	6									
	271	13	4	16	1	10	7	3	12	17	2	6	18	11	14	8	15	5	9									
	1828	3	11	13	5	1	18	14	2	8	15	9	7	17	4	12	16	6	10									
17	1244	11	5	3	6	13	10	14	17	15	4	2	7	9	12	8	1	16										
	338	15	2	12	17	8	5	11	9	3	16	13	10	1	6	4	7	14										
	114	16	10	7	5	8	12	14	17	2	4	13	11	9	6	1	3	15										
16	98	5	15	2	11	6	16	3	10	7	4	14	1	13	9	12	8											
	330	14	10	7	4	12	1	9	5	2	15	3	8	11	13	16	6											
	255	3	13	11	8	4	1	12	15	2	16	9	6	14	10	7	5											
15	378	12	8	5	13	6	10	2	4	14	9	11	15	7	1	3												

Proposed Solution:

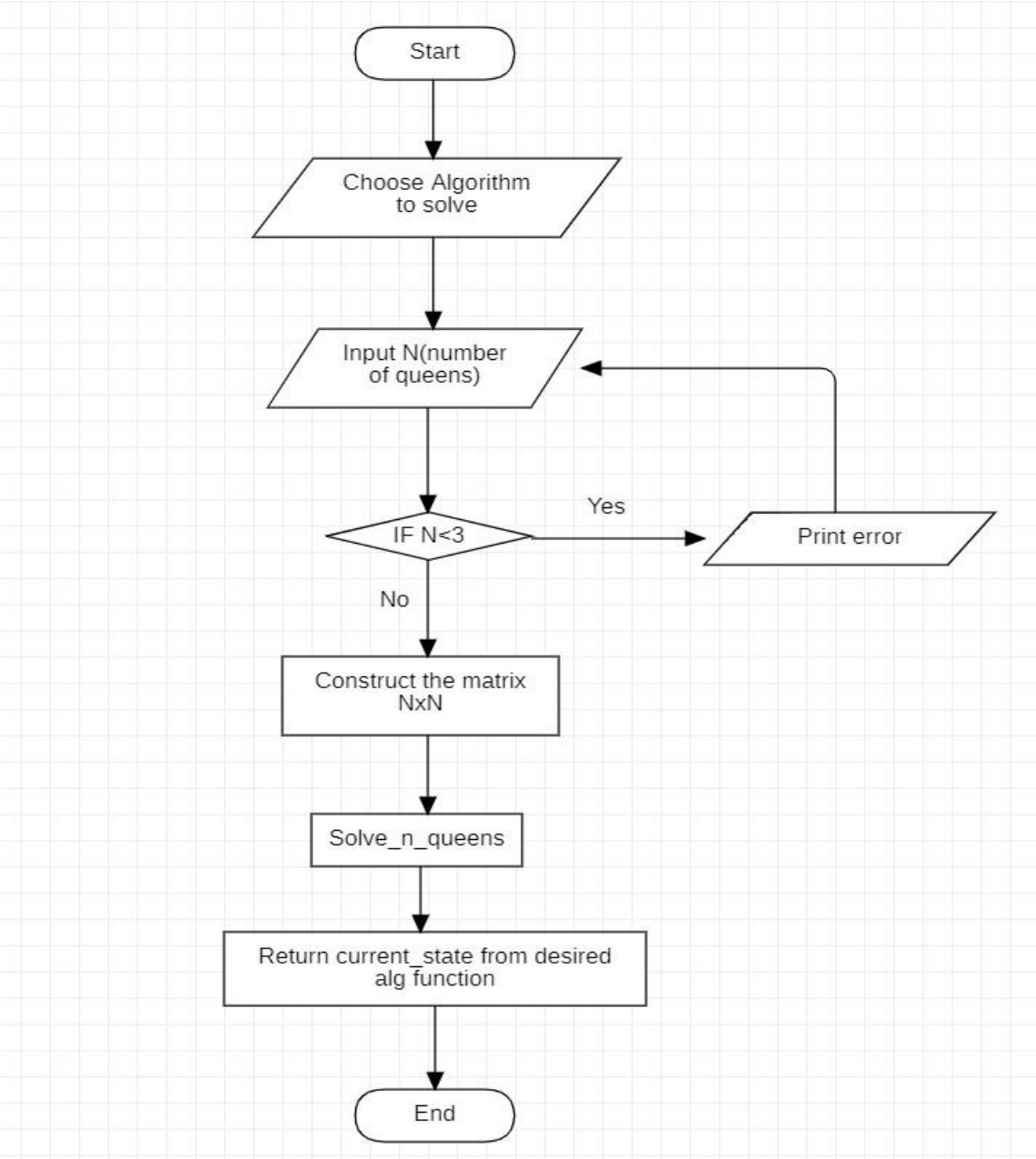
-Functionalities/Features (users' perspective):

use-case:



Applied Algorithms:

flowchart:



-Backtracking:

Backtracking Search Algorithm: The Backtracking Search Algorithm is a systematic method for exploring potential solutions to a problem. In the context of the N-Queens problem, it involves placing queens on the board one by one and backtracking if a conflict is detected. This algorithm guarantees finding all possible solutions.

-Genetic:

The Genetic Algorithm mimics the process of natural selection to evolve solutions. In the N-Queens context, a population of potential solutions undergoes genetic operations like crossover and mutation.

Genetic Algorithm a population of individual chromosomes is selected randomly.

This approach explores a diverse solution space, often finding effective solutions.

Population Initialization: Firstly, we initialize a random population of chromosome of length 1000. Every chromosome here is actually a vector of length N, which actually is a random permutation of (1, 2, 3… N).

Fitness Function: We design the Fitness of each chromosome in such a way that k number of queens on the same diagonal situation will accumulate k-1 points to the fitness value. All these points are summed which obtains the fitness value.

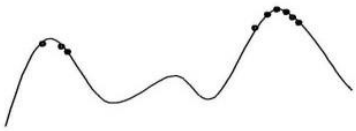
Mutation: Mutation is very important in genetic algorithm for not to stuck the process in local optimum.

An Early Phase:

Quasi-random population distribution.

A Middle Phase:

Population arranged around/on hills.

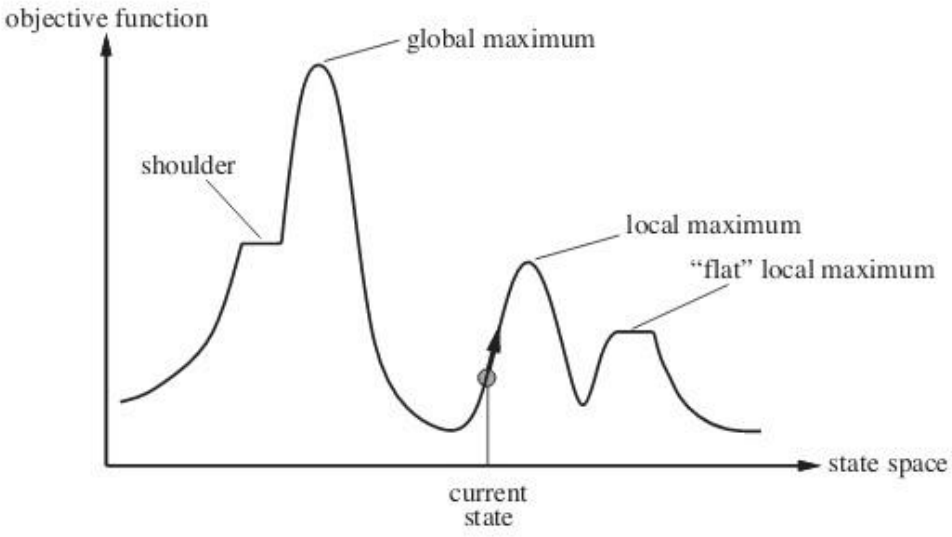


A Late phase:

Population concentrated on high hills.

-Hill-Climbing:

Hill-Climbing is a local search algorithm that continually moves towards higher elevations in the solution space. Applied to the N-Queens problem, it involves iteratively adjusting queen placements to ascend towards a configuration with fewer conflicts. It may get stuck in local optima.



Local maximum: It is a state which is better than its neighboring state however there exists a state which is better than it(global maximum). This state is better because here value of objective function is higher than its neighbors.

Global maximum: It is the best possible state in the state space diagram. This because at this state, objective function has highest value.

Plateau/flat local maximum: It is a flat region of state space where neighboring states have the same value.

Ridge: It is region which is higher than its neighbors but itself has a slope. It is a special kind of local maximum.

Current state: The region of state space diagram where we are currently present during the search.

Shoulder: It is a plateau that has an uphill edge.

Steepest-Ascent Hill-Climbing: It is a variant of Hill Climbing algorithm. In this algorithm, we consider all possible states from the current state and then pick the best one as successor.

Allowing sideways moves: When stuck on a ridge or plateau (i.e., all successors have the same value), allow it to move anyway hoping it is a shoulder and after some time, there will be a way up.

Random-restart hill-climbing: If the first hill-climbing attempt doesn't work, try again and again and again. That is, generate random initial states and perform hill-climbing again and again. This is random-restart. The number of attempts needs to be limited, this number depends on the problem.

-Best-First:

Best-First Search is an algorithm that intelligently selects the most promising path based on a heuristic evaluation. In the N-Queens context, this algorithm evaluates board configurations using a heuristic to guide the placement of queens, prioritizing paths that seem most likely to lead to a solution.

Experiments & Results:

- The Experiments, testing, and the results

- Time complexity: `O(N!)`: The first queen has `N` placements, the second queen must not be in the same column as the first as well as at an oblique angle, so the second queen has `N-1` possibilities, and so on, with a time complexity of `O(N!)`.
  - Backtracking uses a trial-and-error approach, and it may be slow for some cases where it needs to explore and reject many possibilities before finding a solution.
  - In the best cases, it can be fast for small problems, but it may become slow for larger problems.
  - The solutions depend on the order of tests and preferences, and it might not be efficient for some cases.

- Spatial Complexity: `O(N)`: Need to use arrays to save information.

The time complexity for Hill Climbing algorithm can be divided into three parts:

1. Calculating Objective – The calculation of objective involves iterating through all queens on board and checking the no. of attacking queens, which is done by our calculateObjective function in O(N^2) time.

2. Neighbour Selection and Number of neighbours – The description of neighbours in our problem gives a total of N(N-1) neighbours for the current state. The selection procedure is best fit and therefore requires iterating through all neighbours, which is again O(N^2).
3. Search Space – Search space of our problem consists of a total of NN states, corresponding to all possible configurations of the N Queens on board. Note that this is after taking into account the additional constraint of one queen per column.

➔ Therefore, the worst-case time complexity of our algorithm is O(NN). But this worst-case occurs rarely in practice and thus we can safely consider it to be as good as any other algorithm there is for the N Queen problem. Hence, the effective time complexity consists of only calculating the objective for all neighbours up to a certain depth (no of jumps the search makes), which does not depend on N. Therefore, if the depth of search is d then the time complexity is O (N^2 * N^2 * d), which is O(d*N^4).

In 1850 Franz Nauck gave the 1st solution to this problem and generalized the problem to N-Queen problem for N non- attacking Queens on an N x N Chessboard. Time complexity of an N-Queen problem is O(n!).

The Genetic Algorithm relies on numerous random populations and selecting the fittest individuals for evolution, which can quickly lead to good solutions.

It can be effective in large and complex search spaces.

Solution Quality:

The Genetic Algorithm does not guarantee finding the optimal solution, but it aims to find good solutions based on fitness evaluations.

Solution quality depends on strategies for genetic diversity manipulation and probabilities.

General Comparison:

If you are looking for a precise solution and a guarantee of finding the optimal solution, Backtracking might be the better choice.

If speed is more crucial than finding the optimal solution, especially for dealing with a problem with a large number of variables, the Genetic Algorithm might be more optimal.

The choice between the algorithms depends on the nature of the problem and the specific requirements of the application.

You can try both solutions on a specific problem, measure their performance, and evaluate the results in terms of solution speed and quality.

Analysis, Discussion, and Future Work:

- Analysis of the results, what are the insights?

Hill Climbing

Average time taken: 0.003546333312988281 seconds

Genetic

Average time taken: 0.15293657779693604 seconds

Best-First

Average time taken: 0.0018398046493530273 seconds

Back-Tracking

Average time taken: 0.003911280632019043 seconds

- What are the advantages / disadvantages?

1-Backtracking

Advantages:

1. Efficiency: Backtracking is an efficient algorithmic technique for solving problems that involve searching for a solution in a large search space. It reduces the number of unnecessary computations by intelligently exploring only the promising paths and backtracking when necessary.
2. Completeness: Backtracking guarantees finding all possible solutions to the problem. It systematically explores the search space and exhaustively checks all potential configurations until a solution is found or all possibilities are exhausted.
3. Flexibility: Backtracking is a versatile algorithmic paradigm that can be applied to solve a wide range of problems, including combinatorial problems, puzzles, and constraint satisfaction problems.
4. Optimality: In some cases, backtracking can lead to optimal solutions, especially when exploring the search space systematically and considering all possibilities.
5. Memory Efficiency: Backtracking typically uses less memory compared to other algorithms like dynamic programming because it explores one path at a time and backtracks when necessary, avoiding the need to store all possible solutions.
6. Simplicity: The backtracking approach is often straightforward to implement and understand. The algorithm decomposes a problem into subproblems and solves them incrementally.

Disadvantages:

1. Exponential Time Complexity: In the worst case, backtracking can have an exponential time complexity because it explores all possible solutions. The running time depends heavily on the problem structure and the efficiency of pruning techniques.
2. Completeness: The backtracking algorithm may not guarantee finding a solution, especially if the search space is infinite or if the problem has no feasible solution. It relies on pruning strategies to avoid unnecessary exploration.
3. Search Space Size: Backtracking algorithms may need to explore a large search space, leading to long computation times for certain problems. This can be a limitation in real-time or time-sensitive applications.
4. Difficulty in Choosing Heuristics: The efficiency of backtracking can be highly dependent on the choice of heuristics and pruning strategies. Selecting appropriate criteria for choosing branches and deciding when to backtrack can be challenging.

Not Always the Most Efficient: While backtracking is effective for certain types of problems, it may not always be the most efficient solution. Other algorithms, such as dynamic programming or greedy algorithms, might provide better performance for specific problem instances.

2-Best first search

Advantages:

1. Heuristic-driven: Best-first search algorithms, like A*, use heuristics to guide the search towards promising paths. This can significantly reduce the search space and make the algorithm more efficient compared to exhaustive search methods like backtracking.

2. Optimality: If an admissible heuristic is used, A* guarantees finding an optimal solution, i.e., the configuration with the fewest conflicts or the minimum number of attacking queens.
3. Flexibility: Best-first search algorithms can be easily extended to handle variations of the N Queens problem, such as adding additional constraints or objectives. They provide a framework for incorporating different heuristics or strategies to guide the search towards desired solutions.

Disadvantages:

1. Complexity of Heuristic Design: The effectiveness of a best-first search algorithm heavily relies on the quality of the heuristic function. Designing an effective heuristic that provides accurate estimates of the remaining conflicts or the distance to the goal state can be challenging.
2. Time Complexity: Although best-first search algorithms can reduce the search space, they can still be computationally expensive for large problem instances. The time complexity depends on the quality of the heuristic and the structure of the problem.
3. Memory Usage: Best-first search algorithms may require storing a considerable amount of search state information, including the priority queue or the open/closed lists. This can be memory-intensive, especially for problems with large search spaces.

3-Genetic

Advantages:

1. Parallel Exploration: Genetic Algorithms can explore multiple candidate solutions in parallel. This parallelism allows for efficient exploration of the search space, which can be beneficial for complex problems like the N Queens problem.
2. Global Search: Genetic Algorithms have the ability to perform a global search in the solution space, rather than getting stuck in local optima. By maintaining diversity in the population and applying genetic operators like crossover and mutation, GAs can explore different regions of the search space and potentially find better solutions.
3. Flexibility: Genetic Algorithms are versatile and can be applied to a wide range of optimization problems, including the N Queens problem. They can handle problems with complex constraints, multiple objectives, and non-linear fitness landscapes.

Disadvantages:

1. Convergence Speed: Genetic Algorithms may require a large number of iterations or generations to converge to an optimal or satisfactory solution. The convergence speed can be slower compared to other optimization techniques, especially for problems with large search spaces.
2. Parameter Tuning: Genetic Algorithms involve several parameters, such as population size, crossover rate, and mutation rate. Finding the optimal combination of these parameters can be challenging and time-consuming.

3. Representation and Encoding: Choosing an appropriate representation and encoding scheme for the N Queens problem can be non-trivial. The effectiveness of the GA heavily depends on the representation used to represent candidate solutions and the design of appropriate genetic operators.

4-Hill climbing

Advantages:

1. Simplicity: Hill Climbing is a simple and easy-to-understand algorithm. It is straightforward to implement and requires minimal computational resources.
2. Local Optima Avoidance: Hill Climbing can avoid local optima by continuously making small improvements to the current solution. It explores the immediate neighborhood of the current solution, allowing it to escape local optima and potentially find better solutions.
3. Efficiency for Small Problem Instances: Hill Climbing can be efficient for small problem instances where the search space is manageable. It can quickly find satisfactory solutions in such cases.

Disadvantages:

1. Local Optima: Hill Climbing is prone to getting stuck in local optima. If the algorithm reaches a point where no better solution can be found in the immediate neighborhood, it will terminate without reaching the global optimum.
2. Lack of Global Exploration: Hill Climbing does not perform a global search of the solution space. It only explores the immediate neighbors, which limits its ability to find optimal solutions for complex problems like the N Queens problem with larger board sizes.
3. Sensitivity to Initial Solution: The quality of the solution obtained by Hill Climbing can be highly dependent on the initial solution. If the initial solution is far from the optimal solution, the algorithm may struggle to find a satisfactory solution.

- What might be the future modifications you'd like to try when solving this problem?

The future modifications that can be used that can be done is hybridization between genetic algorithm and differential evolution.

Development platform tools:

VS Code, PyCharm , Jubyter

Programming Language: Python

Python Libraries: random, tkinter, heapq