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:

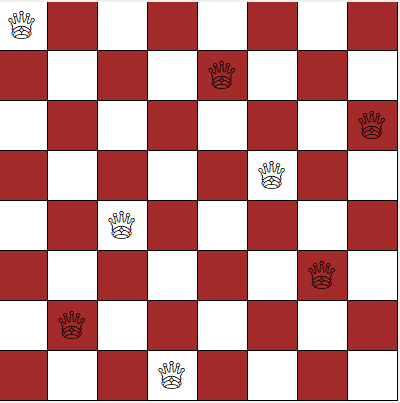
Board size (N): 8  
OR

Number of queens: 8

Output:

Solution found: Yes

The chessboard is an 8x8 grid.



- Similar applications in the market:

Parallel memory storage schemes.

[VLSI](https://www.sciencedirect.com/topics/mathematics/very-large-scale-integration) 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.

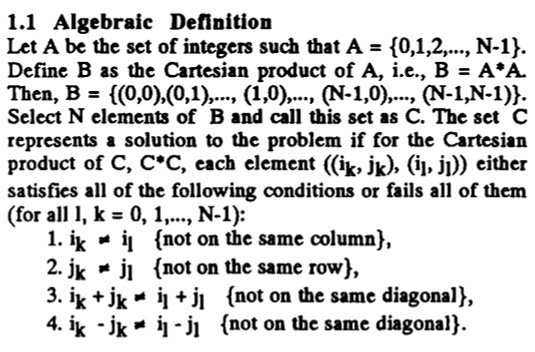
-Solution Visualization: Display the final solution on the chessboard, highlighting the positions of the queens.

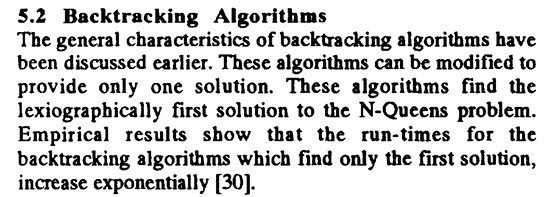
-Multiple Solutions: Provide the option to find and display multiple valid solutions if they exist.

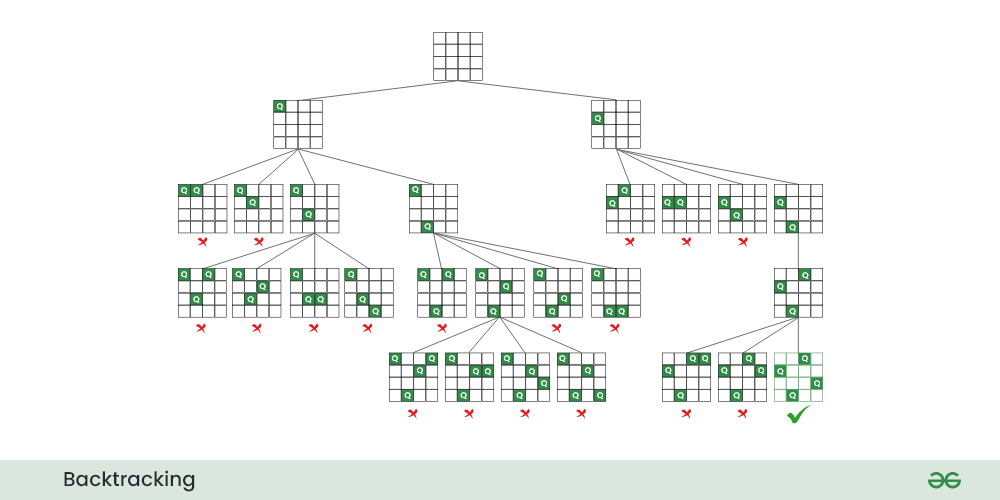
-Performance Metrics: Calculate and display metrics such as the time taken to find a solution and the number of backtracks performed.

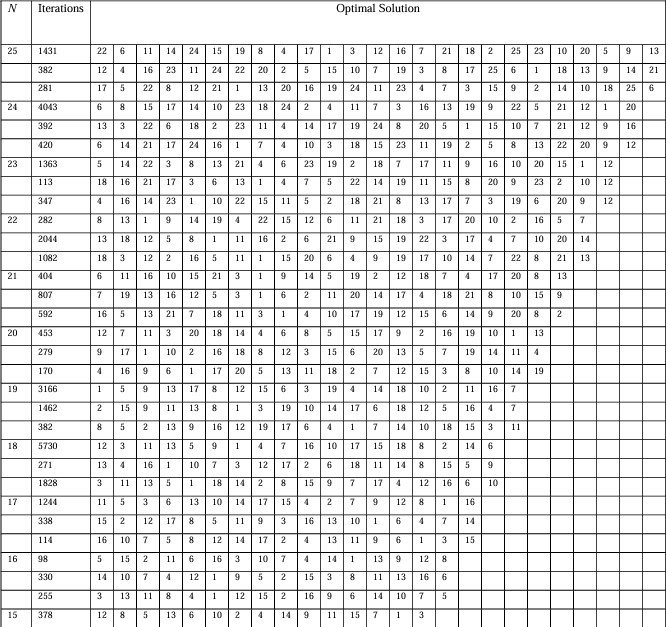
-A Literature Review of Academic publications:

<https://betterprogramming.pub/backtracking-n-queen-problem-and-sudoku-31974988bcb0>  
  
chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://arxiv.org/ftp/arxiv/papers/1802/1802.02006.pdf#:~:text=In%201850%20Franz%20Nauck%20gave,problem%20is%20O(n!).





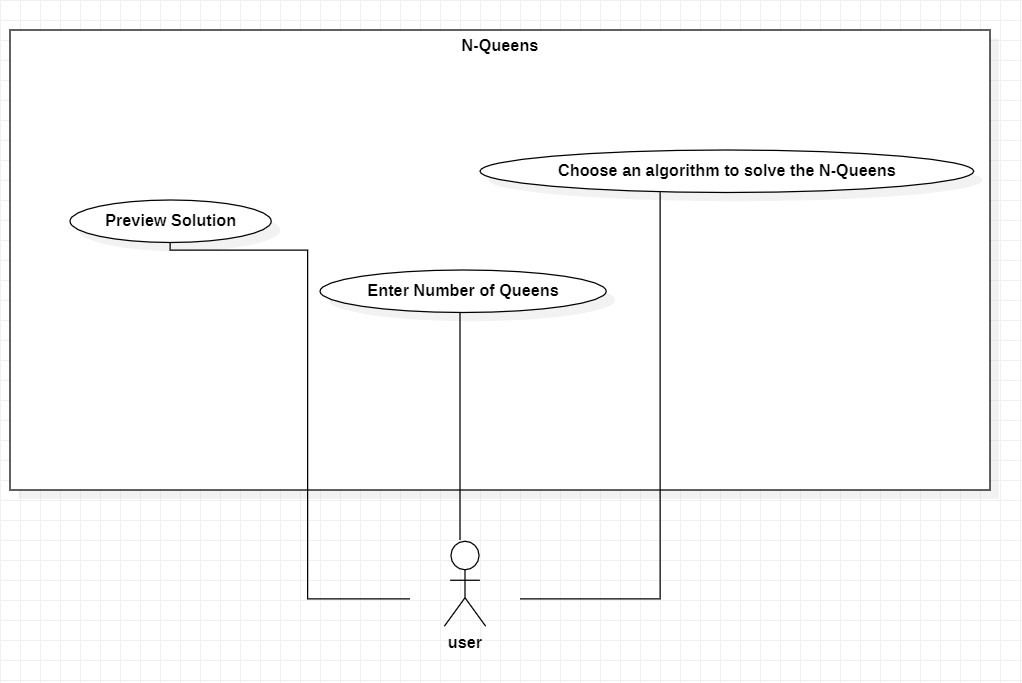




Proposed Solution:

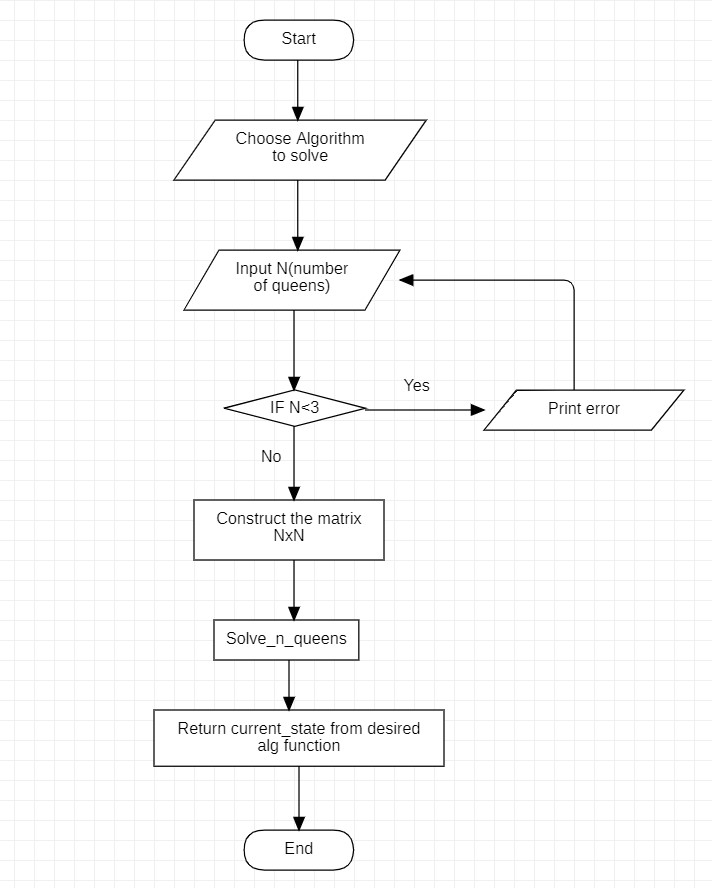
-Functionalities/Features (users’ perspective):

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.

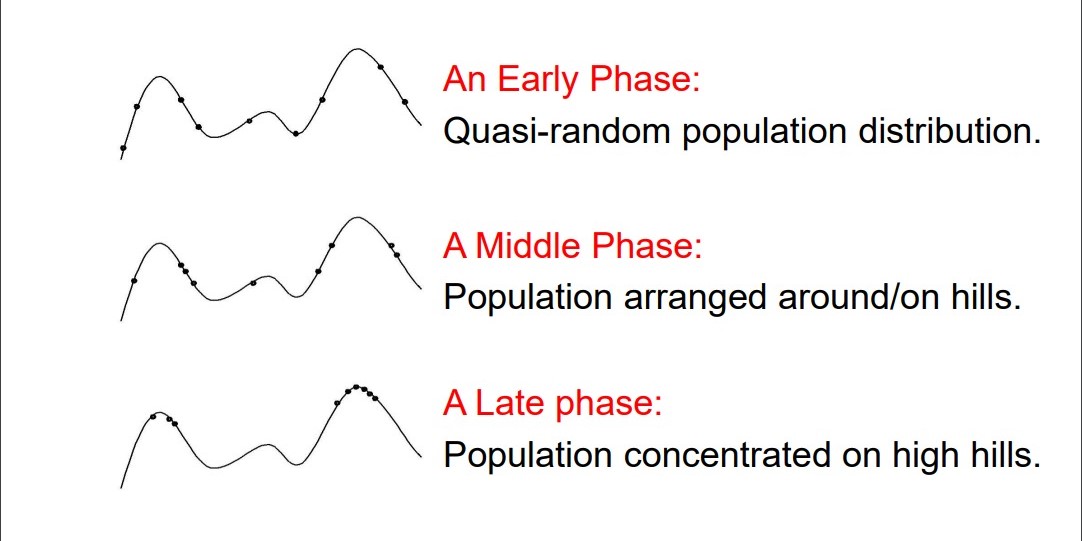
Genetic Algorithm a population of individual chromosomes is selected randomly.

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.

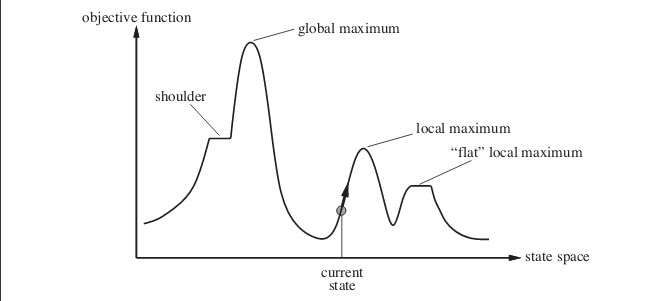
Mutation: Mutation is very important in genetic algorithm for not to stuck the process in local optimum.



-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!).
* 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.

1. 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).

1. 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!).

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 can handle problems with complex constraints and multiple solutions. It allows for easy adaptation to different variations of the problem, such as finding all solutions or finding a single solution.

Disadvantages:

1. Exponential Time Complexity: The N Queens problem has an exponential number of possible configurations to evaluate, making it computationally expensive for large values of N. The time complexity of the backtracking algorithm is typically O(N!), which grows rapidly with increasing N.
2. Space Complexity: Backtracking requires maintaining a stack or recursion stack to keep track of the decision path. This can lead to high space complexity, especially for problems with deep search trees or large problem instances.
3. Implementation Complexity: Implementing the backtracking algorithm correctly and efficiently can be challenging. It involves managing the search state, constraints, and backtracking steps. The complexity of the algorithm increases as the problem constraints become more intricate.

**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.

- Why did the algorithm behave in such a way? What might be the future modifications you’d like to try when solving this

problem?

Development platform tools:

VS Code, PyCharm

Programming Language: Python

Python Libraries: random, tkinter, heapq