Project 2: Solving N-queens problem using hill-climbing and its variants

Team Members
Monesa Thoguluva Janardhanan - 801167556
Chandan Kumar Reddy Mannem - 801165621
Aparajitha Sriram - 801169526
Nourelislam Zouar - 801153922

Introduction

When faced with a problem with an unknown or unknown state space, some algorithms beyond the classical search algorithms could be used. The problem at hand is to solve the N-queens problem. For this we choose the hill-climbing search algorithm and its variants.

The N-Queens problem

The N-queens problem consists of N queens placed on an N x N chessboard such that no two queens attack each other. Two queens are said to attack each other if they are on the same row or on the same column or on the same diagonal. There is no one single solution for this type of problem. Given below is one possible solution for an 8 – Queens problem.

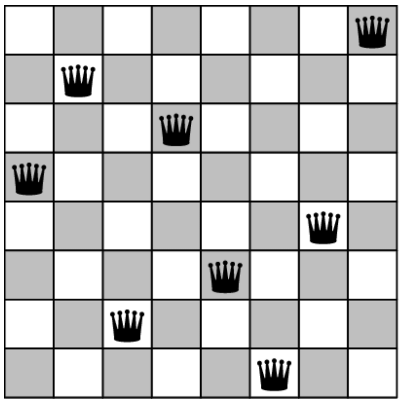

Figure 1 [1]

This type of problem is a typical example of a problem suitable to be solved by using a backtracking approach with an initial state of an empty board. Each queen is placed on each column calculating as to which square has no threat on that column. If there occurs a situation such that there are no squares in a particular column with zero threats, the algorithm should backtrack to the previous step and try a different square for placing the queen. The algorithm would stop when all the queens are placed such that no two queens attach one another.

However, here we use the hill-climbing search algorithm to find a solution for the N-Queens problem.

Hill Climbing Search Algorithm

The Hill-Climbing search is a heuristic based search algorithm. Problems that have a large set of inputs and an unexplored state space can be solved by using the hill-climbing search. Also, this search is typically suited for mathematical optimization problems where an initial state of a problem needs to be optimized step-by-step to reach the solution. This search, in its simplest form, finds the immediate neighbours of the current state and chooses the state which best optimizes the value of the current state. This neighbour state is then assigned as the current state and the search is performed again, till no neighbours with better values are found, which terminates the algorithm. However, the classic hill-climbing search is not optimal and has only 14% success rate. The algorithm could get stuck in a state that is a **local minimum or local maximum**, where none of its neighbours have a better heuristic than the current state.

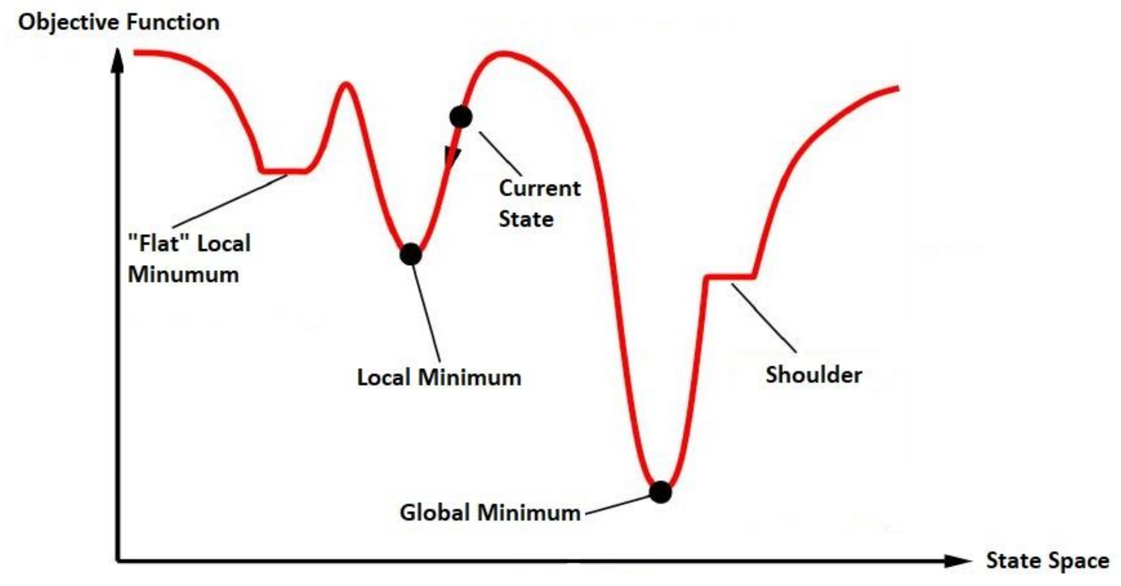

Figure 2 The state space landscape of a hill-climbing search algorithm while trying to find the global minimum.

The classic hill-climbing search could also get stuck in a **shoulder** or a **flat local minimum**, since all the neighbours of the current state might have the same heuristic as itself. Hence, there are some variants of this search algorithm which can be used to overcome these issues.

- Steepest-Ascent Hill-Climbing search
- Hill-Climbing search with sideways move
- Hill-Climbing search with random-restart
- Stochastic Hill-Climbing
- First-Choice Hill-Climbing

Steepest-Ascent Hill-Climbing search:

This search algorithm first examines all the neighbours and selects the state that is closest to the solution state as the next state.

Hill-Climbing search with sideways move:

When the search algorithm gets stuck on a flat local minimum where the best successor has the same heuristic value as the current state, this algorithm allows a sideways move with a hope that the assumed flat local minimum is really a shoulder. However, if it is really a flat local minimum, an infinite loop will occur. Hence, a limit could be assigned on the number of consecutive sideways moves, after which the search is terminated. Allowing sideways move could increase the success rate from 14% to 94%.

Hill-Climbing search with random-restart:

This solves the problem of the hill-climbing search not being complete. This restarts the algorithm from a random initial state when the algorithm hits a local minimum. The algorithm repeatedly restarts the execution till it finds the global minimum and terminates only after that. This can be used with and without the sideways move. Hence, this algorithm is complete with a probability approaching 1 as this will eventually generate the goal state as the initial state.

Stochastic Hill-Climbing:

This algorithm chooses at random from among the uphill or downhill moves. The probability of selection depends on the steepness of the slope and this can have advantages in certain type of landscapes. However, unlike the random-restart algorithm, this is not a complete algorithm.

First-Choice Hill-Climbing:

This implements stochastic hill climbing by generating successors randomly till one is generated with a better heuristic than the current state. This is a good strategy if there are countless number of successors. However, this is not a complete algorithm either.

Problem Formulation

Our goal is to implement the process of solving the N-Queens problem by using the hill-climbing search along with its two variants – sideways move and random-restart (with and without the sideways move). We have implemented these algorithms in Java with a character matrix for maintaining the state of the chessboard. The heuristics for each square in each state of the N-Queens problem will be calculated based on the number of threats prevalent between the queens, in the horizontal, vertical and diagonal directions.

Program Structure

The program has a class **N_QueensAlgorithm**.

Variables (Public):

- **board –** A character matrix that maintains the state of the chessboard.
- **queensArray –** An integer array that maintains the position of the queens on the board.
- **boardCounter –** Maintains the number of times the state has been changed.
- **randomRestartCounter –** Maintains the number of times the algorithm restarted in the random restart version of hill-climbing search.

Functions:

- **shuffleboard ()**
  Input parameters:
  numberOfQueens: Number of queens required by the user.
  algorithm: The algorithm chosen by the user, either Hill Climbing or Random-Restart hill-climbing search.
  Return type: int
  Output: returns 1 for success and 0 for failure
  Purpose:
  - It is responsible for generating the random initial state.
  - Based on the 4 algorithms, it calls the HillClimbing algorithm and returns the success or failure.

- **printQueensBoard ()**
  Input parameters:
  numberOfQueens: Number of queens required by the user.
  Return type: void
  Output: Prints the current board
  Purpose:
  - It displays the positions of the various queens on the current board.

- **hillClimbing ()**
  Input parameters:
  numberOfQueens: Number of queens required by the user.
  presentHeuristic: Heuristic of the current board.
  Return type: int
  Output: Returns the heuristic value of the best successor of the current board.
  Purpose:
  - This is the method that has the core logic of the hill-climbing search on the N-Queens problem.
  - It iterates through every column and places the queen in the square having the best heuristic.

- **calulateHeuristicForBoard ()**
  Input parameters:
  numberOfQueens: Number of queens required by the user.
  Return type: int
  Output: Returns the heuristics of the current state of the board.
  Purpose:
  - It calculates the heuristic of the current state of the board.
  - The heuristic is the total of the number of threats among the queens in the horizontal, vertical and diagonal directions.

- **main()**
  Input parameters:

args: user input from the command line

Return type: void

Purpose:

- Gets the user input for the number of queens and the number of trials.
- Instantiates the class and calls all the corresponding methods for the four algorithms.
- It displays the numberOfTrails, the rate of success, the number of solutions found, the rate of failure and the average number of steps when the algorithm succeeds or fails.

Observations:

The results from the three variants of Hill-Climbing Search algorithm have been summarized below for the 8-Queens problem.

Algorithm	Number of Trials	Rate of Success	Rate of Failure	Avg. no. of steps during success	Avg. no. of steps during failure	No. of solutions found
Hill-Climbing search	100	2%	98%	19	18	2
	500	3.40%	96.60%	14	18	17
Hill-climbing search with sideways move	100	29%	71%	31	120	29
	500	29.20%	70.80%	32	121	146
Hill-Climbing with Random-Restart without sideways move	100	100%	0%	14	0	100
	500	100%	0%	15	0	500
Hill-Climbing with Random-Restart with sideways move	100	100%	0%	30	0	100
	500	100%	0%	33	0	500

Algorithm	Number of Trails	Average number of random restarts required
Hill-Climbing with Random-Restart without sideways move	100	1620
	500	6975
Hill-Climbing with Random-Restart with sideways move	100	130
	500	703

The below are the results for the 4-Queens problem.

Algorithm	Number of Trials	Rate of Success	Rate of Failure	Avg. no. of steps during success	Avg. no. of steps during failure	No. of solutions found
Hill-Climbing search	100	20%	80%	6	6	20
	500	20%	80%	6	6	100
Hill-climbing search with sideways move	100	57%	43%	12	24	57
	500	66%	34%	12	25	330
Hill-Climbing with Random-Restart without sideways move	100	100%	0%	5	0	100
	500	100%	0%	5	0	500
Hill-Climbing with Random-Restart with sideways move	100	100%	0%	12	0	100
	500	100%	0%	12	0	500

Algorithm	Number of Trials	Average number of random restarts required
Hill-Climbing with Random-Restart without sideways move	100	182
	500	810
Hill-Climbing with Random-Restart with sideways move	100	27
	500	134

From our results, we found that the hill-climbing search performed better when it used the random-restart mechanism. The success rate increased to a full 100% in both the 8-Queens and 4-Queens problem.

Conclusion

Thus, by using the Hill-Climbing search algorithm, we have been able to implement the N-Queens problem and find a solution for it. We have used four algorithms which are the Hill-Climbing Search and the Random-Restart algorithm, both with and without the sideways move.

Sample Output

The search sequences and the output of the various values from four random initial configurations for each of the algorithms in an 8-Queens problem is included in the .zip file. We have included only one trial of the random-restart algorithm as it repeatedly restarts a number of times from a random initial state till it hits the goal state.

References

[1] “A visual representation of a solution for the 8-queens problem.” [Online]. Available: https://www.researchgate.net/figure/A-visual-representation-of-a-solution-for-the-8-queens-problem-left-and-the-variables_fig1_333815714

[2] “Introduction to Hill Climbing | Artificial Intelligence.” [Online]. Available: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/