

Solving N-Queen Problem Using Problem Solving Agent

Dhananjay Arora
Department of Computer Science
University of North Carolina
Charlotte NC, USA
darora2@uncc.edu

Abstract— This paper describes an implementation of a problem-solving agent in solving the n-queen problem. It addresses the way in which attacked cells and random heuristic is used to achieve the desired result which is demonstrated by a working code.

Keywords—*heuristic algorithm, n-queen problem*

I. INTRODUCTION

This paper demonstrates a way to solve the n-queen problem with the help of a problem-solving agent. The goal is to find a configuration of n queens not attacking each other by implementing the heuristic algorithm. Usually, an 8*8 board is taken and 8 queens are arranged in a manner so that no queen is attacking each other. This N-queen problem was invented in mid-1800s for people to solve it as a puzzle in their spare time but now is used as a good tool for discussing algorithms.

Some problems have a straightforward solution which can be achieved by just following some specific set of procedures. Since, the solution is only one and we must follow standard procedure every time. So, this problem-solving agent is hardly intelligent. On the other hand, some problems have more than one solution and all the alternative solutions need to be analyzed before solving the problem. Those kinds of problems require intelligent problem-solving agent. The set of objects among which we search for the solution is called the search space. For instance, search space here is the n-queen configurations.

II. BACKGROUND

A. Problem-solving agent

An Intelligent Agent is meant to maximize its performance measure. Problem-solving agent is a type of goal-based agent. General steps of problem-solving are:

- Goal formulation
- Problem formulation
- Search
- Execute

This problem in this paper has a discrete environment where more than one solution is possible for a problem.

B. Graph Search Method

This search method involves generating stepwise configuration.

- Initial State- No queens are placed on the board.

- Goal State- It depicts the target positions of N queens and none of the queens is attacking each other.
- Links- It depicts operators such as an addition of a queen.
- States- It corresponds to the configuration of N queens. For instance, a location in a map that is represented by nodes in the graph.

C. Backtracking Method

It is a method to solve this problem by following the below-mentioned steps:

- Place the queen in the top row. Note the diagonal and the column it occupies.
- Place next queen in the next row down, taking care of not placing it in the same column or diagonal. Keep track of preoccupied columns and diagonals and move on to the next row.
- If there is no position left in the next row due to the preoccupied column or diagonal, then backtracking starts. Now, go to the previous row and move the queen to the next available position in its row and start again with the previous step.

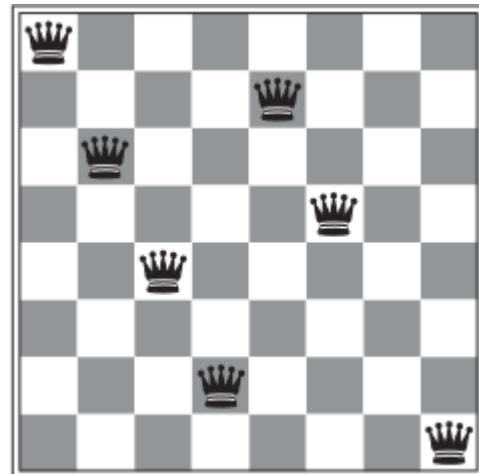


Figure1: 8-Queen Problem with two queens attacking

D. Heuristic Algorithm

A heuristic function ranks all the possible alternatives at any branching step in search algorithm based on available information. It tries to find the best route out of all possible routes. When $N > 8$, then the previous problem-solving techniques take a large amount of time. So, to solve the problem in a reasonable amount of time, the attacked cell and random heuristic algorithm is used to solve the N-Queen problem.

III. PROPOSITION

In this paper, inference has been taken from the below-mentioned Hill Climbing Approach and a solution is designed to solve the N Queen problem in Java accordingly.

Hill Climbing Heuristic method is used to solve the mathematical optimization problem in the field of Artificial Intelligence. When the input size is large, it tries to find a sufficiently good solution but that may not be optimal. It tries to provide the solution in a reasonable amount of time.

This hill-climbing takes the feedback from test procedure and this feedback is utilized by generator to decide next move in search space.

- *Simple Hill Climbing:* It examines the neighboring nodes one by one and selects the first neighboring node which optimizes the current cost as next node.
- *Steepest-Ascent Hill Climbing:* It first examines all the neighboring nodes and then selects the node closest to the solution state as next node.

Below is the algorithm for Steepest-Ascent Hill climbing:

Step 1: Evaluate the initial state. If it is goal state, then exit else make the current state as initial state

Step 2: Repeat these steps until a solution is found or current state does not change

- Let 'target' be a state such that any successor of the current state will be better than it.
- For each operator that applies to the current state:
 - apply the new operator and create a new state
 - evaluate the new state
 - if this state is goal state then quit, else compare with 'target'
 - if this state is better than 'target', set this state as 'target'
 - if target is better than current state set current state to Target

Step 3: Exit

- *Stochastic Hill Climbing:* It does not examine all the neighboring nodes before deciding which node to select. It just selects a neighboring node at random and decides (based on the amount of improvement in that neighbor) whether to move to that neighbor or to examine another.

IV. SOLUTION

Steepest ascent hill climbing algorithm solves this N queen problem only with 16% efficiency, otherwise it gets stuck at some point.

N queen problem is successfully solved with 100% efficiency by implementing the following algorithm, where heuristic being used is randomly computing the number of attacked cells and trying to reduce the number of attacks until it reaches zero.

Step 1: Initialize the N*N board by putting queens in a random position on the board such that there is only one queen in each row and make current state as initial state.

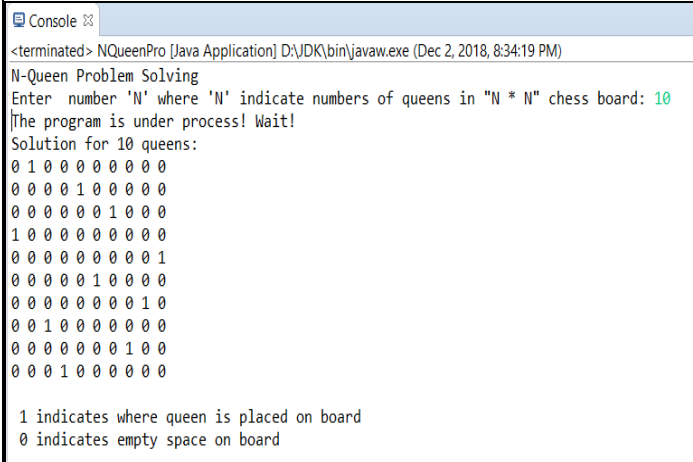
Step 2: Repeat these steps until a solution is found or number of attacks is zero in the current state.

- Evaluate the number of attacks in current state. Attack signifies how many queens are attacking each other horizontally, vertically or diagonally.
- If number of attacks are not zero, find a new state by adjusting the position of the queens on the board, such that the number of attacks decreases and then set the current state to new state.
- If no such next state is found, then set current state to initial state.

Step 3: Exit

The above algorithm is implemented in Java and it consists of below six methods:

- `printBoardinTerminal()`: Prints solution in console
- `evaluate()`: Calculates the number of attacks between queens
- `generateBoard()`: Generates new state from current state
- `findNextState()`: Calls `generateBoard()` method to generate a new state and if it has lesser number of attacks than the current state, then sets the current state to new state.
- `initialRandomBoard()`: Makes an initial state by putting one queen at a random position in each row.
- `SolveNQueen()`: This method evaluates the board again and again until it finds a new state in which the number of attacks is zero. To find a new state it calls method `findNextState()` and if it is unable to find any new state then it reinitializes the board by calling method `initialRandomBoard()`.



```
<terminated> NQueenPro [Java Application] D:\JDK\bin\javaw.exe (Dec 2, 2018, 8:34:19 PM)
N-Queen Problem Solving
Enter number 'N' where 'N' indicate numbers of queens in "N * N" chess board: 10
[The program is under process! Wait!]
Solution for 10 queens:
0 1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 0

1 indicates where queen is placed on board
0 indicates empty space on board
```

Figure 2: Solution to N queen problem, where N=10

V. CONCLUSION

The paper showed that the N queen problem can be successfully solved using the heuristic algorithm and the desired result is achieved in a realistic time frame, and the implemented Java code solves the N Queen problem with the help of the heuristic algorithm.

REFERENCES

- [1] I. Martinjak, and M. Golub, "Comparison of Heuristic Algorithms for the N-Queen Problem," IEEE Explore, July 2007, DOI: 10.1109/ITI.2007.4283867.
- [2] S. Russell and P. Norvig, "Artificial Intelligence-A Modern Approach," Third Edition, 2010, Pearson Education, Inc., pp. 64-107.
- [3] U.Rawat, Introduction to Hill Climbing | Artificial. (n.d.) Retrieved Nov 1, 2018 from <https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/>