Solving the N-Queens problem through Minimum Conflicts Algorithm with Comparison to Backtracking

Juri Ando, Joseph Gleason

CIS 4930 - Intro to A.I.

4-16-2019

I. Introduction

The N-Queens problem is a constraint satisfaction problem where there is a board of size n x n with n queens placed on the board. The objective of the search is to find a placement for all queens such that there is no queen threatening (able to attack) any other queen on the board based on the rules of chess. This means that for any given queen, there will be no other queen in the vertical, horizontal, or either diagonal directions. This type of problem is classified as a constraint satisfaction problem, which is any problem with a set of constraints on sets of variable/value combinations. In our case, the variables in the problem are the queens and the values are the possible positions on the board that the queens can be placed. With this problem definition, we will take a look at two main implementations to discover valid solutions.

II. Choice of Implementations

Since this problem is considered a constraint satisfaction problem, there are many different approaches to reach a valid solution. These different methods include, but are not limited to: Backtracking, Heuristic functions such as minimum remaining values, degree heuristics, least constraining value, forward checking, and minimum conflicts. The main implementation we chose to use is the minimum conflicts algorithm. For our implementation,

we chose to use minimum conflicts and backtracking as comparison of performance. Using minimum conflicts will theoretically give us the best results when compared to backtracking which theoretically should have exponential growth as the number of queens increases.

III. Algorithm Description

Backtracking is the basic uniformed algorithm for CSPs. This implementation can be represented as depth-first search with single-variable assignments. Since the branching factor of this algorithm is b = (n-l)d, there will be n!dn leaves which leads to O(n!) time complexity. As the number of queens increases, the time will grow exponentially very quickly which is terrible for performance.

Minimum Conflicts is a much more efficient algorithm with the following implementation. We start with a random assignment of all variables in the problem. For n queens, this means that each queen is assigned a random, unique position on the board. From here we calculate all of the conflicts for each of the queens and store these results. We then choose a random queen that is conflicted and find a new position for that queen that will minimize the conflicts of that queen. We keep on selecting conflicted queens until there are no more conflicts on any of the queens. When there are no more conflicts on any of the queens, this means that all constraints have been satisfied and we can return the positions of the queens.

IV. Improvements to minimum conflicts

Although minimum conflicts is the most efficient of the algorithms for CSPs, the basic minimum conflicts algorithm has some critical issues. The main issue in consideration is the fact that the search can get stuck in local minimums. This is clearly an issue due to the fact that a

true solution will never be reached. Due to this fact, we need to implement some way of escaping these multiple instances of local minimums. Our implementation was to incorporate backtracking within the minimum conflicts algorithm. This way, if we can see we are visiting the same assignments over and over again, we can go back and try to minimize the conflicts from a different angle. There are other implementations to escape a local minimum such as starting the entire search over but the implementation of backtracking proves to be most efficient.

The other main improvement on the algorithm we can make is the ability to use the n-queens problem definition to our advantage. The first way we can use this is in a greedy placement of the initial queens. One fact about all solutions is that there will be exactly one queen per column (or row). This means we can start with one queen in every column and randomize the row placement in each row. We can also use this fact of one queen per column to help our reduction of conflicts. Instead of looking at the entire board for the best position, we can now only look at the column the queen is currently in. By implementing these two improvements, we have effectively eliminated the row variable in the search for a solution.

V. Results

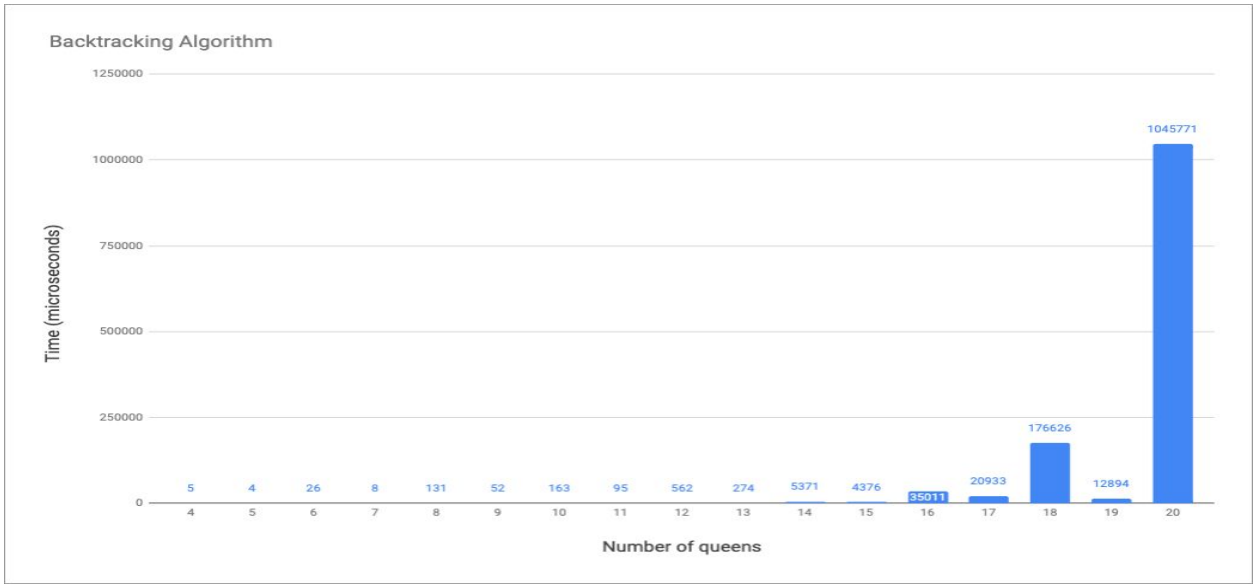

Figure 1) Performance measure of time on backtracking algorithm for n = 4-20

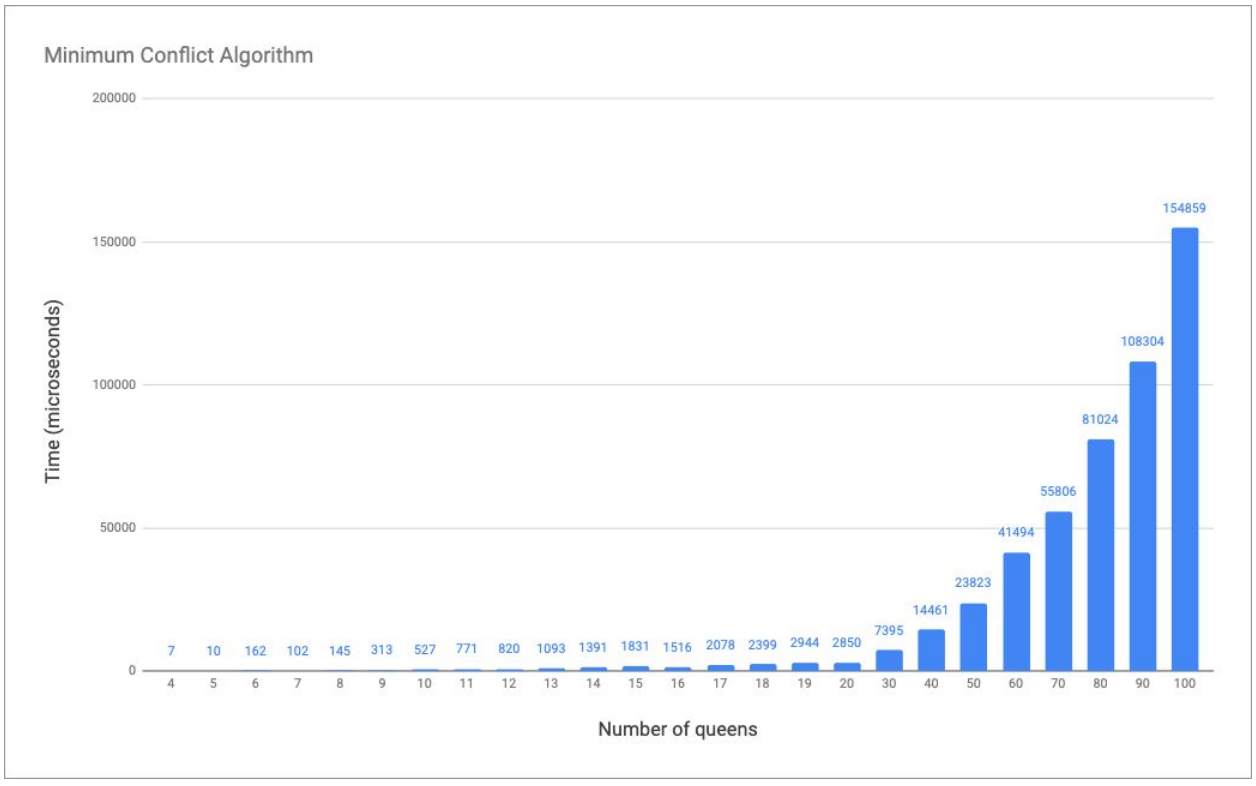

Figure 2) Performance measure of time on minimum conflicts algorithm for n = 4-100

VI. Discussion

Through the results shown, we can see that the backtracking algorithm grows exponentially as the number of queens increases. This is proven to be inefficient due to the fact that this algorithm can only handle very small cases with no room for growth. Minimum conflicts on the other hand can be seen to grow in a linear manner which will prove to be much more useful as the problem/data being tested grows to a large scale.

Through this project, we have effectively shown the implementation of the minimum conflicts algorithm on the n-queens problem in comparison to other implementations such as backtracking. Through this comparison we can see why it is important to use effective algorithms to find quick and efficient solutions as the input data grows larger and larger.