Simon Weber

CSC173

Scheme Week 3-4

N-Queens Problem in Scheme

Overview

The purpose of this assignment was to implement and analyze various algorithms for solving the N-Queens problem. The N-Queens problem is to place n queens on a chessboard of size n by n, so that no queen attacks any other.

Algorithms:

Backtracking

This algorithm searches through all possible placements of queens until it finds a valid one. It starts by placing one queen in the first column and first row, then tries to place the other queens so that they do not conflict. Once there is no possible place to put a queen, it pulls the current queen off the board and tries the previous queen in its next row. The order of rows is important in the efficiency of the algorithm (this is discussed in Analysis). It will always find a solution, but it is inefficient and a weak method.

Backtracking with MRV Heuristic

This is same algorithm as above, but with an important modification. The next row that is chosen to be moved is the row that is currently causing the most attacks. This usually improves the efficiency of the algorithm over the backtracking algorithm.

Min-Conflicts

The min-conflicts algorithm is a hillclimbing algorithm; it starts with all queens placed on the board and then tries to improve the arrangement. It randomly chooses a conflicting queen and then moves it to the row that causes the least conflicts. A problem with this hillclimbing technique is that an arrangement can be created that cannot be improved any further. This is called a local minima. To eliminate these situations, I had to consider a few special cases. The board would be caught in a local minima if only two queens conflict and the rest are fine. When this situation arises, a conflicting queen is moved to a random row to create conflicts and start the process over again.

Analysis

Measurement

To measure the efficiency of the algorithms relative to one another, I measured the number of queen placements that each made.

Column Order in Backtracking Search

I tried a few different ways to pick the order of columns in the backtracking search: inside-out, left to right, and random. Here are the results:

Problem Size	Inside-Out	Left-right	Random (average over 5 trials)
5	92	90	102
6	3454461	5595877	45188846.8
7	6449	572	2120.2
8	10843	1.58E+021	57399846397.8
10	149695890888	2.13E+019	2.05E+033
12	1.45E+026	1.22E+047	4.73E+021

This data shows that the inside-out column order is generally the most efficient, with case 7 being a notable exception. There, it performs worse than both of the other types. The inside-out approach was based on the idea that the inside columns will affect the outer columns more, so they should be chosen first.

Initial State of Min-conflicts

The initial state of min-conflicts should have a large affect on the efficiency of the algorithm. I tried two different initial states, a diagonal row of queens from top-left to bottom-right, and a random arrangement that differs with each run of the algorithm. Each of the values below is based off of 5 runs of the algorithm.

Diagonal

Problem Size	Median	Min	Max	Std. Deviation
4	9	6	19	6.3
5	14	8	68	24.8
6	80	20	102	37.37
7	13	10	25	6.19
8	42	13	328	133.72
9	44	22	191	67.82
10	92	37	338	129.13
15	283	82	443	137.46
20	88	61	215	65.16

Random

Problem Size	Median	Min	Max	Std. Deviation
4	2	41	26	11.32
5	8	2	13	3.96
6	75	28	151	44.48
7	44	27	98	33.44
8	45	5	54	20.51
9	52	20	144	50.05
10	161	74	250	80.87
15	182	37	324	106.04
20	183	25	521	184.65

A few interesting trends arise from this data. First, while certainly the initial states affect the outcome, it is tough to say that one is better than the other. Judging by the median, neither stands out as significantly more efficient. The minimum and maximum measurements do not show much of a difference either. However, how the problem size relates to the median is interesting. For the diagonal tests, the median jumps around, while for the random tests, it increases steadily with the problem size (excluding problem size 6 as an outlier). Another interesting trend is how the Standard Deviation relates to the problem size. For the random arrangement, it increases with the problem size, but the diagonal arrangement does not follow a pattern. Based on these results, I chose the random arrangement to represent the min-conflicts algorithm in the later tests.

In Artificial Intelligence: A Modern Approach, Russell claims that the n-queens problem, when solved with the min-conflicts technique, is independent of problem size. My data does not firmly support that conclusion. The median value does go up with problem size, but the minimum value remains relatively constant. Therefore, my data is inconclusive.

Algorithm Comparisons

Here are the results when comparing the algorithms against each other:

Problem Size	Backtrack	Backtrack + MRV	Min Conflicts
5	92	90	14
6	3454461	26453	80
7	6449	215	13
8	10843	1197955	42
10	149695890888	10773068	92
12	1.45E+026	2624	117

Some conclusions can be drawn from this data. First, the MRV approach is clearly more efficient than just backtracking alone. Second, the hill climbing method is the overall best method. However, the efficiency measurement of min conflicts does not take in to account how "difficult" each decision to move a queen is. So, in reality it can take longer to use the hill climbing method. Here are the algorithms measured with cpu time, as measured by the built in scheme function *time*:

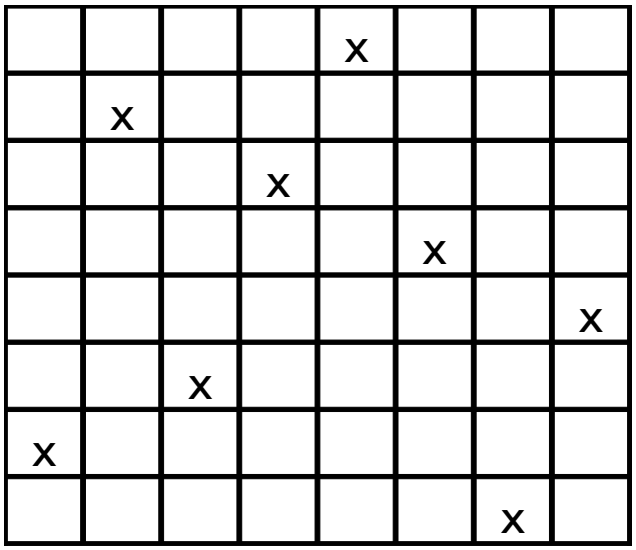
Problem Size	Backtrack	Backtrack + MV	Min Conflicts
17	62	47	110
18	93	62	73
19	265	62	173
20	921	78	280

Here, different results are shown. Min-conflicts is now less efficient than the backtracking technique with the heuristic. This can mostly be written off on the implementation of min-conflicts. Operations such as getting the columns with conflicts or getting the best row to place a column in could be improved, which would lead to a better run-time. Also, the heuristics applied in min-conflicts are not very well developed, and while they do decrease the number of operations, they may increase the run-time of the algorithm.

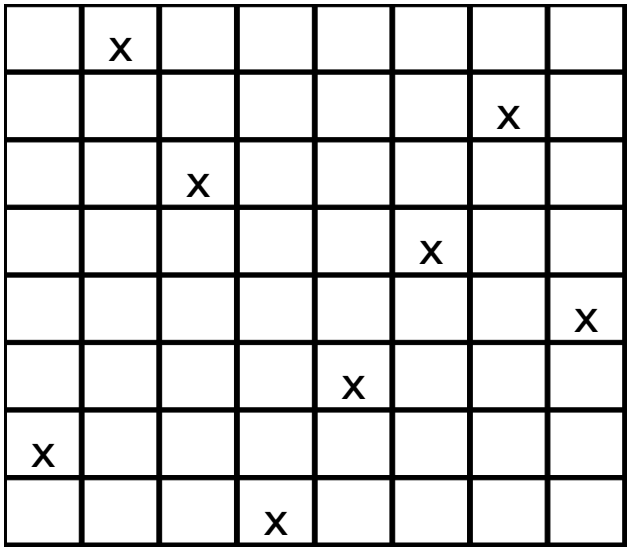
Selected Solutions

These solutions illustrate how the different searches will find different results for the same given problem size.

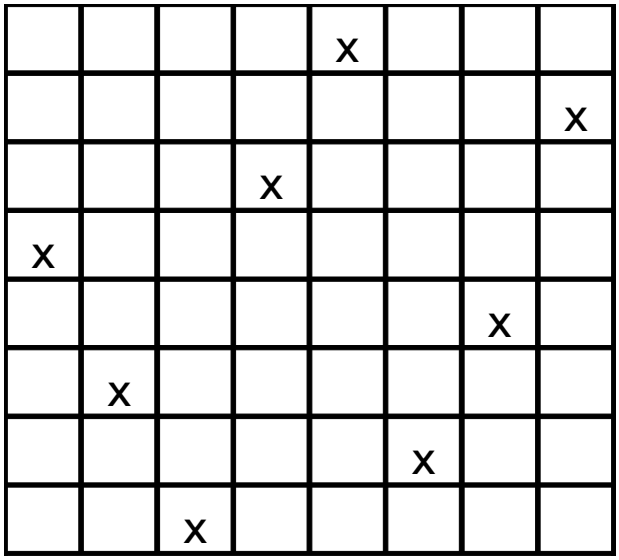
Backtracking:



Backtracking with MRV:



Min-Conflicts:



Note that the first two solutions are the same, but the board is simply rotated.

Conclusions

The inside-out column order for backtracking is the best option. No conclusion can be drawn from the initial state of Min-conflicts, however the standard deviation of the random approach rises with problem size. Of the different algorithms, the min-conflicts approach is the best in terms of operations, however it does not have the best run-time. The run-time could be improved through improvement of the helper functions. For the backtracking methods, the work required increases with problem size. However, for min-conflicts, no conclusion can be drawn because of the minimum values.