Project 3

CS 4200 - Artificial Intelligence

Instructor: Dominick A. Atanasio

Computer Science Department

California State Polytechnic University, Pomona

Report Submitted On: 15-MAR-2020

Report Submitted By:

Damian Ugalde

Approach:

This program uses the Minimum Conflict Algorithm to solve the N-Queens problem. This board uses the constraint that all columns must have exactly one queen at all times. To keep track of the position of the queens, an integer array of N-length was used. The index represents the column, while the value stored in the array represents the row. The hardest part about this implementation is to keep track of all the conflicts in the board, in a way that is faster to implement than checking the board conflicts each time a change needs to be made. To keep track of the conflicts, I used a HashMap that uses the integer value of the column as the key, and a HashSet as the value. The HashSet contains integers that represent the number of the column with a conflict. An auxiliary boolean array of N-length was also used to keep track of which columns had conflicts. The index of the array was the column. If the value was set to true, it means that this column is conflicting with another value in the board. This is used for selecting a random column to minimize.

Analysis:

For the varying sizes, a max steps value of 10000 was used.
For the max steps, size 25 boards were used.
Each test was performed 400 times for each variation.

Varying sizes

Size	% Success	Time (ms):
8	100	0.2
16	100	0.6
25	100	2.0
50	100	13.6
60	100	26.9
100	100	256.6
125	93.25	725.6
150	10	1321.9
175	0	1783.6
200	0	2872.4

Varying max steps

Max Steps	% Success	Time (ms):
100	5.75	0.885
200	51.75	1.5875
300	85.75	1.625
400	95.75	1.8
500	98	1.61
600	100	1.5775
700	99.75	1.625
800	100	1.6725
900	100	1.4925
1000	100	1.6475
2500	100	1.625
5000	100	1.6125
7500	100	1.4575
10000	100	1.44
12500	100	1.44
15000	100	1.4625
17500	100	1.3525

Findings:

This algorithm was considerably faster than using the simulated annealing and genetic approaches used in the previous project. Simulated annealing had an average solve time of 100 milliseconds. The genetic algorithm had an average solution time of 1500 milliseconds. This algorithm had an average solution time of 1.67 milliseconds.

Compared to the other two implementations, this algorithm was also able to fund solutions for more board sizes. Simulated annealing had a success rate of 100% for boards of size 25, and 98% for boards of size 50. The genetic algorithm had a best success rate of 78% for board sizes of 25. This algorithm has a 100% success rate for boards smaller than 100. This algorithm's success rate drops dramatically after for boards greater than 125. For boards of size 175 or greater, the success percentage drops to 0.

This algorithm saw a decrease in the success percentage when the maximum steps were decreased. For less than 600 steps, the probability of success decreased to reach a low at 100 steps of 5% success. Above 700 steps, the success percentage was 100%.