John Wu

Dr. Sen

AI

8 March 2021

Project 2: CSP

Intro:

In this project we tested various methods of solving constraint satisfaction problems from the simple depth-first search, to forward checking, to finally the arc consistency 3 method. In addition to these methods, we tested different variable ordering heuristics to accompany them, these were random, minimum remaining value, and minimum remaining value with degree.

Implementation:

First, I will begin by providing a brief outline of my implementation for each of the different processes and heuristics.

For depth first search on both the sudoku and map coloring problems I simply went through each node, assigned values and checked for their validity. If the value was valid I would continue on, otherwise I would go down the list of available values testing until the end where upon failure I would backtrack and change the most recently "solved" variable and change its value.

For forward checking, I simply selected my variable and assigned it the first remaining value in its list of remaining values. Afterwards, I would go to each of its neighbors or cells that it influenced and changed the remaining values in their list of remaining values.

For the arc consistency algorithm, I simply took a similar approach as I did with forward checking, but I instead compared my current variable to the values of its neighbors and the cells that influence it. Or, in other words, I simply changed the values available to each variable based on what variables have already been established in the cells that influence the current one.

The different heuristics were fairly simple to implement, for random I just used the random.shuffle method inherent to python, while the MRV and MRV with degree I used a series of sorts to sort my lists based on the remaining amount of unknowns it still has and the amount of variables it influences.

Results:

The picture below showcases my results. The results displayed are the average time elapsed in milliseconds over 20 runs using the different methods and heuristics.

```
Average time elapsed for mapColorAC3 using heuristic 0 is: 0.3496
Average time elapsed for mapColorDFS using heuristic 0 is: 0.54945
Average time elapsed for mapColorForward using heuristic 0 is: 0.3497
Average time elapsed for mapColorAC3 using heuristic 1 is: 0.44945
Average time elapsed for mapColorDFS using heuristic 1 is: 0.49955
Average time elapsed for mapColorForward using heuristic 1 is: 0.39954999999999996
Average time elapsed for mapColorAC3 using heuristic 2 is: 0.39960000000000007
Average time elapsed for mapColorDFS using heuristic 2 is: 0.44894999999999996
Average time elapsed for mapColorForward using heuristic 2 is: 0.34965
Attempting to solve a sudoku puzzle of form:
[9, 0, 6, 0, 5, 3, 2, 8, 7]
[2, 8, 3, 7, 0, 4, 9, 1, 5]
[7, 5, 0, 9, 2, 8, 4, 6, 0]
[0, 3, 2, 4, 8, 7, 6, 5, 9]
[0, 0, 5, 6, 9, 2, 0, 3, 4]
[6, 9, 0, 0, 0, 5, 7, 2, 8]
[4, 2, 9, 0, 0, 1, 5, 7, 6]
[5, 6, 8, 2, 7, 0, 0, 0, 0]
[0, 1, 7, 5, 4, 0, 8, 9, 2]
Average time elapsed for sudokuDFS using heuristic 0 is: 0.04995000000000001
Average time elapsed for sudokuForward using heuristic 0 is: 0.1498
Average time elapsed for sudokuAC3 using heuristic 0 is: 0.2
Average time elapsed for sudokuDFS using heuristic 1 is: 0.04995000000000001
Average time elapsed for sudokuForward using heuristic 1 is: 0.09985000000000001
Average time elapsed for sudokuAC3 using heuristic 1 is: 0.1999
Average time elapsed for sudokuDFS using heuristic 2 is: 0.0499
Average time elapsed for sudokuForward using heuristic 2 is: 0.19980000000000003
Average time elapsed for sudokuAC3 using heuristic 2 is: 0.24969999999999998
```

You can see from the picture that on average the time elapsed for the methods favors the following format: forward checking > ac3 > depth-first search. That is to say that forward checking is usually faster than AC3, while AC3 is usually faster than depth-first search. However, perhaps due to an issue with my implementation, we see that the depth-first search is actually faster than both forward checking and AC3 in the evaluation of the 20-hole sudoku puzzle.

We also see that the different heuristics make a slight change in time complexity, at least between random ordering against the other two. The MRV and MRV with degree have similar time complexity because they're similar, but pure random ordering generally takes longer.

Analysis:

In the end, we see that depth-first search is generally the least efficient method, as it should be, and it might be due to my implementation, but there is merit to using it for certain problems. Perhaps for the smaller problems, it may be simply faster to brute-force your way to the answer rather than taking the time to evaluate your way to a true answer. Furthermore, for my map-coloring solutions,

both forward-checking and AC3 are faster than the depth-first search, but depth-first search gets me the absolute minimum amount of colors that would be required to use. Whereas the other two usually require 1 extra color.