Project 3

CSP with Depth First Search

David Bush

CS 470 Soule

June 6th, 2021

Abstract: For this project I implemented 3 different variations of a depth first search algorithm: the first one used simple backtracking with no heuristics, the second one used backtracking with a constraint ordering heuristic, and the third one used constraint ordering and forward checking. I began by testing each algorithm on a test case only using 5 variables to ensure they were working smoothly before moving onto the bigger test case. After each one appeared to work, I tested each algorithm on the 30 variable test case starting first at 2 colors and moving up to 3 colors and 4 until the solution was found. The algorithms were unable to find a solution for the 2 and 3 color cases but found one for the 4 color case. The results showed that algorithm with forward checking performed the best, the algorithm with constraint ordering was second, and the basic depth first search performed the worst.

Algorithms: The basic depth first search algorithm works by following a branch as far as it goes or until a constraint is violated. Once a constraint is violated the search backtracks to the previous node and tries the next branch and this continues until a solution is found or the entire tree has been searched (no solution exists). The next algorithm I used was the same as the basic depth first search but added a constraint ordering heuristic. Constraint ordering is rearranging the list of variables and placing the ones with the most constraints (in our case the number of neighbors) at the top. This allows us to start searching the variables with the most constraints first which leads to better performance overall since less branches will be explored. For the third algorithm I used the depth first search with constraint ordering and added forward checking. Forward checking is a technique that keeps track of what domains are left for each

variable and when their domain is zero and they have not been assigned a value the algorithm will backtrack. This prevents it from searching certain sub-trees since it knows ahead of time whether there is a possible solution which can save a lot of time and improve the performance of the search.

Results for 30 Variables:

	Basic Depth First Search			Constraint Ordering			Constraint Ordering + Forward Check		
# of Colors	2	3	4	2	3	4	2	3	4
Solution Found	No	No	Yes	No	No	Yes	No	No	Yes
Time (s)	0.095	272.74	0.02	0.006	1.257	0.013	0.019	1.0459	0.031
Recursive Calls	114	233121	32	22	984	29	14	714	29

Green = Successfully Found Solution; Red = Worst Performance Overall

Above is a table that contains the results of each depth first search algorithm given different assessments. The first section shows the number of colors each algorithm was tested on. The next section determines whether a solution was found. The third section shows the required time the algorithm took to complete the entire execution. The fourth section highlights the number of recursive calls the search function used during the execution.

Conclusion: In this project I implemented 3 different depth first search algorithms and recorded their results. At the start I decided to take a unique path in how I represented my variables. We were given a map of neighboring variables which were our constraints, and I decided to represent each variable in a class which contained the

name of the variable and a list of its neighboring regions. I think this made it a little easier to keep track of neighboring variables and made it easier to code. I first began by testing each algorithm with a case of only 5 variables. They were all able to find the solution at 3 colors but each one took a slightly different route to get there. Next, I tried running the case with 30 variables for each algorithm and none of them found a solution for 2 or 3 colors. If the algorithms missed a solution at 3 colors, I would guess that it was due to human error with the data being entered wrong since the algorithms had worked in the smaller case prior. Examining the results of each algorithm, its apparent there was an outlier with the basic depth search with 3 colors since it searched nearly 233,000 nodes which was magnitudes higher than any others. I am not certain if this was caused by an error or if it was to be expected since it appears to have searched the entire tree. The results appeared to be similar for the constraint ordering algorithm and forward checking algorithm, but I would assume with a larger test case, like 1000 variables, the forward checking would perform the best by a greater margin. I think the depth first search algorithms would be improved by adding more heuristics and by testing more thoroughly with additional cases. To conclude, all the algorithms used were able to find a solution using a depth first search with different attributes.