Joe Shymanski 3/8/21

Project 2: Constraint Satisfaction Techniques

Objective

The objective of this project is to implement a backtracking algorithm which uses one of three variable selection heuristics [Random Variable, Minimum Remaining Value (MRV), and MRV with Degree] and one of two inference methods [Forward Checking (FC) and AC-3] on two Constraint Satisfaction Problems (CSPs) [Graph Coloring Problem (GCP) and Sudoku].

Implementation

Backtracking works with a CSP object in order to build an assignment which satisfies all the constraints of the problem. The CSP is made up of three items. The first is the list of variables within the problem. For example, in Sudoku, this would be a list containing all 81 cells on the board. The next is the list of domains for each variable. This would be the integers one through nine, paired with each cell on the grid. The last major component of the CSP is the list of constraints. A single constraint is comprised of a scope and a relation. The scope contains all the variables being constrained, and the relation is the rule which all of the variables in the scope my satisfy. In Sudoku, the scope could be a whole row of cells, and the relation would be that each cell in that row must be unique.

The assignment is the object which gets built throughout the backtracking algorithm. It contains pairs of variables and values, where each value comes from that variable's domain. Depending on the CSP, it can contain some pairings initially or be completely blank. In Sudoku, the board is usually given an initial configuration, and thus the assignment object is partially built before running the backtracking algorithm.

The skeleton of the algorithm contains four major components. The first is checking to see if the assignment is complete. This is fairly simple, as you only need to check to ensure that each variable is present in the assignment and paired with a proper value. The validity of the assignment is constantly being checked at other points in the algorithm, so the only necessary check at this point is that all variables are present in the assignment. If the assignment is complete, immediately return the assignment. If not, continue on to selecting a variable.

The second major component is variable selection. This algorithm contains three different heuristics. The first heuristic randomly picks an unassigned variable. The next heuristic, MRV, chooses the variable with the smallest number of acceptable values remaining in its domain. The last heuristic, MRV with Degree, breaks any ties from the MRV heuristic by choosing the variable with the most remaining neighbors, which are all the other unassigned variables involved in the current variable's constraints.

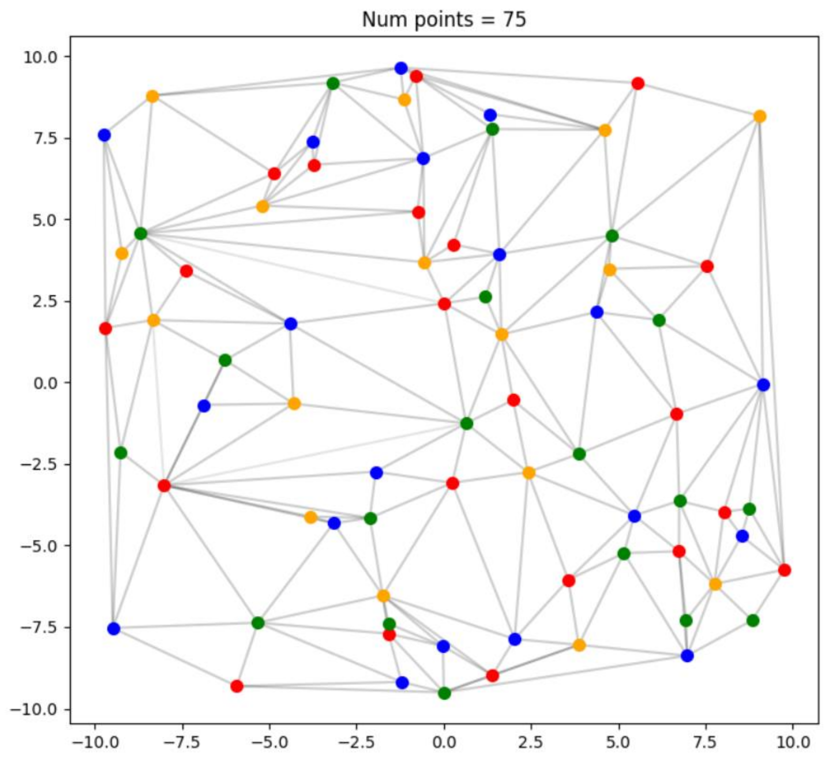
The third component is ordering the chosen variable's domain values. This implementation uses the Least Constraining Value (LCV) heuristic. LCV prioritizes the values which have the smallest number of conflicts with unassigned neighbors. The list of values returned by LCV is put in ascending order and then looped over in the backtracking algorithm.

The fourth component is assignment, inference, and recursion. If the value is consistent with the chosen variable's constraints, the pairing is added to the assignment. Then the algorithm makes inferences on the remaining variables' domains. There are two inference methods in this implementation. The first is FC, which takes the new assignment, loops over the domains of all of the variable's neighbors, and removes any values which are now in conflict from the new assignment. The second is AC-3, which operates just like FC, but every time a domain is revised, that variable's neighbors are then looped over and the algorithm repeats until no neighbors are left to check. If either inference method causes a domain to be empty, then the function returns failure. If not, it returns the revised domains. Once the inferences have been made, the backtracking algorithm is called again with the revised CSP and new assignment objects. If it ever reaches the end of the function, it recursively returns failure, meaning there is no solution given the variables, domains, and constraints.

Results & Analysis

Figure 1

GCP Solution Example



Note. Each node only has 4 color options in this example.

As shown above in Figure 1, the GCP solution ensures that every node sharing an edge is colored differently. Each edge is demonstrated as a line connecting nodes.

Table 1

GCP: All Variable Selection Heuristics with FC and AC-3 Inference Methods

Number of Nodes	Average Time Elapsed (s)					
	Random Variable[a]		MRV[b]		MRV with Degree[b]	
	FC	AC-3	FC	AC-3	FC	AC-3
10	0.002419	0.004612	0.003031	0.003117	0.002810	0.003914
30	0.02025	0.03841	0.01666	0.03681	0.01678	0.03401
50	0.04320	0.1103	0.03601	0.09016	0.05077	0.1131
75	0.1033	0.2361	0.08208	0.3470	0.1574	0.4565
100	0.1508	0.3976	0.2189	0.4861	0.2376	0.6000

Note. Each entry represents an average of 10 runs. Each run generates a new graph.

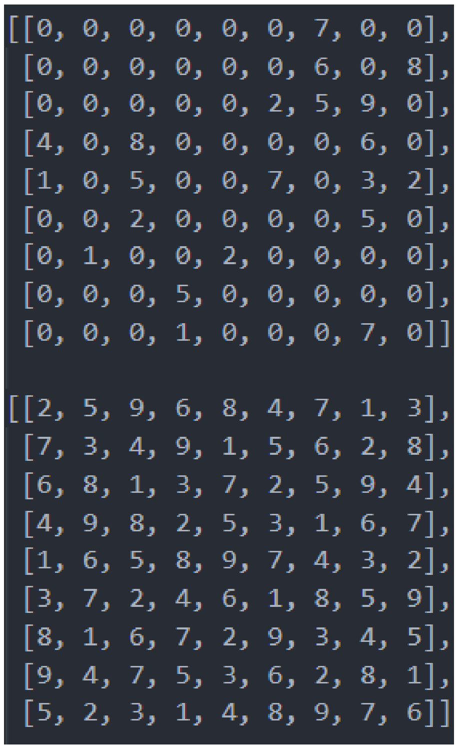
[a]Each run is tested with 6 color options in the domain, in order to save time.

[b]Each run is tested with 4 color options in the domain.

Table 1 demonstrates a series of successful runs according to number of nodes, the variable selection heuristic, and the inference method. As you can see, every combination of selection heuristic and inference method increases in average time elapsed as the number of nodes increases. In each instance, the AC-3 method takes about twice to three times as long as the FC method. The MRV heuristic performs slightly faster than the MRV with Degree heuristic on average.

Figure 2

Sudoku Problem and Solution Example



```
[[0, 0, 0, 0, 0, 0, 7, 0, 0],
 [0, 0, 0, 0, 0, 0, 6, 0, 8],
 [0, 0, 0, 0, 0, 2, 5, 9, 0],
 [4, 0, 8, 0, 0, 0, 0, 6, 0],
 [1, 0, 5, 0, 0, 7, 0, 3, 2],
 [0, 0, 2, 0, 0, 0, 0, 5, 0],
 [0, 1, 0, 0, 2, 0, 0, 0, 0],
 [0, 0, 0, 5, 0, 0, 0, 0, 0],
 [0, 0, 0, 1, 0, 0, 0, 7, 0]]

[[2, 5, 9, 6, 8, 4, 7, 1, 3],
 [7, 3, 4, 9, 1, 5, 6, 2, 8],
 [6, 8, 1, 3, 7, 2, 5, 9, 4],
 [4, 9, 8, 2, 5, 3, 1, 6, 7],
 [1, 6, 5, 8, 9, 7, 4, 3, 2],
 [3, 7, 2, 4, 6, 1, 8, 5, 9],
 [8, 1, 6, 7, 2, 9, 3, 4, 5],
 [9, 4, 7, 5, 3, 6, 2, 8, 1],
 [5, 2, 3, 1, 4, 8, 9, 7, 6]]
```

Note. The top square represents the problem and the bottom square represents the solution.

Figure 2 demonstrates an initial, unfilled Sudoku board and the subsequent solution board. In this specific example, there are 50 unfilled cells, which is one of the more difficult problems for the backtracking algorithm to solve.

Table 2

Sudoku: All Variable Selection Heuristics with FC and AC-3 Inference Methods

Number of Empty Slots	Average Time Elapsed (s)					
	Random Variable[a]		MRV[b]		MRV with Degree[b]	
	FC	AC-3	FC	AC-3	FC	AC-3
10	0.001193	0.001327	0.001029	0.001177	0.001467	0.001673
20	0.002618	0.003703	0.002576	0.003113	0.004305	0.005383
40	0.7367	0.06315	0.008032	0.01816	0.01730	0.02964
45	9.021	0.5801	0.01165	0.02789	0.02154	0.03813
50	30+[c]	3.165	0.02132	0.04961	0.03499	0.07162
55	30+[c]	30+[c]	0.03647	0.1093	0.07990	0.1698
60	30+[c]	30+[c]	0.03351	0.4147	0.7761	2.451
80	30+[c]	30+[c]	0.03773	0.2014	0.04205	0.2096

Note. Each run generates a new board arrangement.

[a]Each entry represents an average of 10 runs, in order to save time.

[b]Each entry represents an average of 100 runs.

[c]The test runs had not completed after 5 minutes, meaning the average exceeds 30 seconds.

Table 2 shows the performance of the backtracking algorithm on various Sudoku problems. Interestingly enough, the AC-3 inference method, which is generally slower than FC in all other instances, outperforms FC when paired with the Random Variable selection heuristic. This is easily seen with 45 and 50 empty slots on the initial Sudoku board. Every combination of selection heuristic and inference method takes more time on average to find a solution the as the number of empty slots increases. Around 60 empty slots, the algorithm has fewer initial constraints to worry about and thus the average time elapsed begins to decrease.

Conclusion

The combination of MRV with FC seems to find successful solutions in the quickest amount of time on average. In any case, MRV takes slightly less time than MRV with Degree, but both significantly outperform the Random Variable selection heuristic. For inferences, FC generally takes less time than AC-3. However, the data from the Sudoku table seem to imply that AC-3 could be better for instances of larger domains where there is no good selection heuristic. The algorithm could be further tested on other CSPs in order to better determine the situations in which AC-3 is better suited than FC and MRV with Degree is better suited than MRV.