

Noemi Turner

Dr. Soule

CS 470

12 June 2022

## Project 3: CSP

### Abstract

For this project, I looked at different constraint satisfaction problem (CSP) algorithms and compared how they perform when used to solve map-vertex coloring problems. Some of the results surprised me a little. In the beginning I expected adding a heuristic to depth-first search would dramatically change the results and the problem-solving speed, but it didn't make a huge difference. That being said, I only tested my CSP algorithms on two graphs. Further testing in the future could provide more insights.

### CSP Algorithms

I coded two depth-first search constraint satisfaction algorithms and then also researched how the greedy local search CSP algorithms perform, since I had limited time resources.

First I used the CSP depth-first search algorithm we learned in class as a starting point to try to find a solution for the map in CSPData.csv.

Then I modified the CSP depth-search algorithm so that it would always assign a color to the vertex with the highest degree first. Here is how I thought through how to write the algorithm:

Colors: 0, 1, 2, 3

- 1: Choose the vertex with the highest degree and assign it the color 0
- 2:     Choose the vertex with the highest degree and also assign it the color 0
- 3:         If there are any conflicts, switch the vertex in step 2 to the color 1
4.     Choose the next highest degree vertex and assign it color 0
5.         check for conflicts and switch to color 1,
- 6:         if still conflicts, then switch to color 2
- 7: etc.

I had wanted to explore local search and use hill climbing to try to solve the map coloring problem. Since I didn't have time to code up a hill climbing algorithm, I did some research and found that GSAT could've been a cool algorithm to test out. GSAT (greedy SAT) is a greedy local search algorithm that helps to solve satisfiability problems using hill climbing. I suspect that GSAT would've worked well on the small 5 node graph but would've not performed well on the

large graph since GSAT often, “gets easily stuck in local minima of the evaluation function.”  
<https://www.sciencedirect.com/topics/computer-science/unsatisfied-clause>

## Results

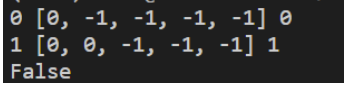
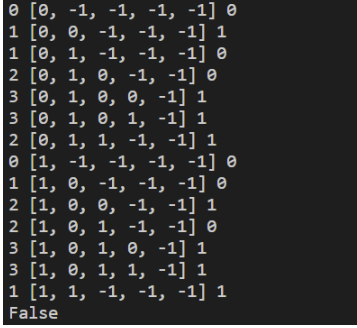
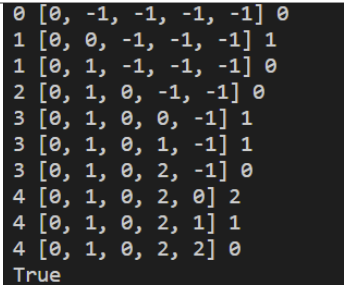
Number of Colors	Result	Screenshot of Results	Steps before result was found
1	False		2
2	False		14
3	True		10

Figure 1: CSP Depth First Search Results for 5 Node Map



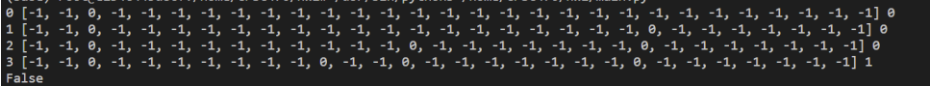
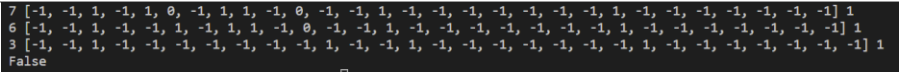
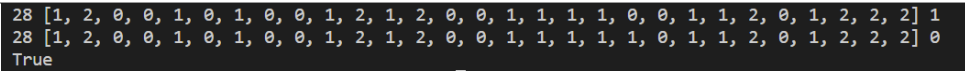
Number of Colors	Result	Screenshot of Results	Steps before result was found
1	False		4
2	False		126
3	True		13,715

Figure 4: CSP Highest Degree Depth-First Search for CSPData.csv Map

In Figure 5, we see how the two algorithms perform when tested against the large graph from CSPData.csv. When we are trying to find out the minimum amount of colors, we need to color a graph, it seems like my Highest Degree DFS algorithm might solve the problem a little faster.

Number of Colors	Steps before result was found for: CSP DFS	Steps before result was found for: Highest Degree DFS
1	2	4
2	230	126
3	21,201	13,715
4	72	62
5	60	62
6	60	62
7	60	62
8	60	62
9	60	62

Figure 5: CSP DFS versus Highest Degree DFS

## Conclusion

After analyzing my results, I noticed that the two algorithms performed very similar, except when finding the smallest number of colors to color the map with as we saw in Figure 5.

This project helped me strengthen my python coding skills, improve my spatial thinking skills, and be more curious about using constraint satisfaction algorithms to solve problems.

In the future, I would like to try creating an algorithm to generate different maps to test the performance of the CSP algorithms I write.