HW1 Constraint Satisfaction: Sydney Chen

My backtracking solution implements backtracking with arc consistency and variable/value ordering. The backtrack function chooses unassigned nodes then tries each of its colors to find a solution, going back to the previous call whenever an assignment leads to a dead end. When choosing a node to assign, it chooses the most constrained unassigned node and breaks ties by choosing the most constraining out of those. After that, it tries assigning that node’s possible colors in order from least to most constraining. If none of its neighbors are already assigned to the color, (node = color) is added to the assignment. Any time it adds an assignment (node = color) it checks arc consistency to prune invalid colors in other nodes or show that the assignment will lead to no solution. In the former case, backtrack continues to recurse in pursuit of the next assignment for a possible solution. In the latter, backtrack restores the pruned colors and goes back to the previous call to try a different color.

My local search solution modifies a random complete assignment each time to minimize conflicts. In each loop, it picks a random node and reassigns it to the color that matches the fewest of its neighbors (if it isn’t already that color). It either loops until the assignment is a solution--no node shares a color with its neighbor--or it loops until it has looped 50x the number of nodes in the CSP. If it has looped that much, it likely got stuck in an unsolvable loop. In that case, it re-randomizes the initial assignment to give us a chance of finding a solution. Before adding re-randomizing, my local search never solved the continental US. If the loop still keeps going past a minute, the search halts.