Group Members:

Joshua Tabor

Report

Contributions: I did everything, but a huge thanks to Juan Garcia as he helped me immensely

Solution 1: Backtracking Search

For my backtracking solution I implemented a most constrained variable (however in my code I mostly address it as minimum remaining value [MRV]). My code functions recursively, at each step finding the state that has the least number of colors still available for that state. It then eliminates that color from the surrounding states, and continues to search down the recursion line until it either reaches a solution or runs into a node that has run out of available colors, meaning that it cannot legally be assigned a color.

Solution 2: Local Search

For my local search implementation, I tried two approaches, one coming from a combination of the book and Juan, and one as a suggestion from Professor Palay. The idea behind both is very similar: simulated annealing. I start by randomizing the problem space and the randomly picking a state and then assigning that state a random color. If that color resolves conflicts, i.e it is not the same color as more of its neighbors than the current state, then I automatically take it. Otherwise I allow bad moves based off how bad the move is and according to some probability given by $e^{\Delta E/T}$. The difference between the two simulated annealing functions is that one randomly restarts after it has gone through enough searches, and the other only searches one problem space the entire time. Both of them search for a max of one minute.