Minimum Conflicts

Minimum conflicts is a local search algorithm that can be used to solve the graph coloring problem. The approach is to randomly color the graph and assign each node a conflict score determined by how many of its neighbors have a color in common with that node. The algorithm then randomly chooses a conflicting node and change it to a color that results in the least conflicts. The algorithm continues to choose conflicting nodes and changing them until there are no more conflicts or an until a predefined limit on how long the program runs is reached.

There are several design decisions that have been made to facilitate the use of this algorithm. One decision is to include a conflict score variable with each point. In addition, some functions will have to be implemented such as a node selector that ignores nodes with zero conflicts and a function that selects a color that causes the least amount of conflicts. The experimental design will consist of a counter that is incremented each time a new node is chosen. This should be sufficient as all other operations should be of constant or O(N) complexity at worst.

Backtracking

Backtracking is a variant of the depth-first search. For the graph coloring problem the approach is to choose a node and assign it a valid color. The algorithm continues to do this until a solution is found or a node is reached that cannot be assigned a valid color. If the node cannot be assigned a valid color the algorithm back tracks to the previous node and tries a different color.

This algorithm will have several design features that are required to make the algorithm work. The foremost feature is that the algorithm will be recursive as it is a convenient way of keeping track of old states. Each instance of the recursive function will have a list of previously used colors so it can know when to backtrack. The experimental design will consist of a counter that is incremented each time a node is colored.