Artificial Intelligence: HW 2 (v1.0)
CSPs
Due March 7, 2020

Overview

The learning objectives for this assignment are:

- solve search problem using CSP style solutions
- utilize random start positions to analysis algorithm performance

Introduction

Searching for solutions to a problem is a core component of AI. Constraint satisfaction problems represent problems more generally, defining:

- X as a set of variables $\{X_1, X_2, ..., X_n\}$
- D is a set of domains $\{D_1, D_2, ..., D_n\}$
- C is a set of constraints that specify allowable combinations of values.

Map Coloring

In this assignment, you will implement two methods to solve the map coloring problem. The map coloring problem is, given a graph $\mathcal{G}$, assign a color to each node n such that no two adjacent nodes (those connected by an edge) have the same color. For this assignment, you need to find the minimum number of colors, k, that can be used and still result in a successful map coloring. This problem has some interesting characteristics and is used in many situations (one of which is social media analysis and protein function prediction).

The two methods you will be using to find the solution to this problem are:

- backtracking (section 6.3 in the textbook)
- min conflicts (section 6.4)

Input

You will be supplied 2 graphs for this problem (one small and one large). The input will be supplied in the form of an adjacent list, where each node is an entry in a Python dictionary. Edges are considered undirected.

For each method, you must run each method several times, decreasing k each time, as the goal of the assignment is to find the minimum value of k where the graph can be successfully colored.

Tasks

You have been provided a class, *CSP*, as a way to represent a general constraint satisfaction problem. This generic class maintains:

- a list of variables
- a dictionary for each variable that contains its domain
- a dictionary for each variable that contains a list of neighbors (variables that have constraints with the current variable)
- a constraint function that returns true if a set of neighbors does not violate any constraints

I am also providing a derived class for extending this to map coloring.

Your job is to implement the backtracking method described in the textbook and the slides and the min conflicts method. The backtracking method **MUST** implement the **degree heuristic**. For extra credit (+10), you can implement a second variant of backtracking that utilizes the minimum remaining value (MRV) heuristic. If you decide to implement MRV, you must treat this as a 3rd method (and report the run time statistics independently, and compare it to the basic backtracking and min-conflicts).

For min-conflicts, your report should specifically discuss the challenges of finding the minimum k (does backtracking suffer from this problem). You need to discuss how the initial assignment of variables for *min-conflicts* effects the run-time of the method (run 10 times for a value of k and measure the variance).

Performance Investigation

Once you have completed and tested this class, you will test the performance difference between the two different methods. You can accomplish this by selecting values of k that backtracking can handle

In a report (PDF), show at least 10 values of k for each method and for each input problem. See the rubrics for a detailed list of what needs to be submitted with this assignment.

Collaboration Policy and Package Usage

For this project, you are only allow to use the following python packages contained in the template file plus time, numpy and matplotlib if you choose. If you need more than these packages, please see me prior to using them in your assignment.

If you need more than these packages, please **request** them from me (as maybe I omitted a few here).

For this assignment, you may work in a group of 2 (and with prior approval, a group of 3). Helping colleagues (other groups) to find a bug or to help them understand some concepts is OK, sharing of code or obtaining code that you did not write is prohibited. You should be able to explain all of your code to me, as failure to do so will be an indicating that you over-collaborated. I will be using plagiarism detection tools in this class.

If you have questions about this policy, please see me.

Deliverables

The following deliverables must be placed in a single zip file named PA2Canvas.zip and uploaded to canvas. This zip file must include the following:

1. Your python code in a file named *cs444_PA2.py*. Make sure that your name appears at the top of the file.
2. Sample output from your program (you can simply run your program and redirect the output to a file). Make sure this sample output show the start state to a problem, the move sequence of the solution, and the goal state. If you print more than one problem, please add some whitespace.
3. A PDF showing the results of the search techniques (timing of each method). These results should be shown in a 10 by 3 table, and if you wish, you can additionally add a plot.

Rubrics

The following is the breakdown how this assignment will be graded:

Item	Description	Points
Readability/Style	Program is commented, following naming style for either Python/Java. Points will be deducted for poor variable names or unreadable code	5
backtracking search	implemented correctly (with the degree heuristic) with examples that clearly show how to run your code.	25
min-conflicts	implemented correctly with examples that clearly show how to run your code	25
k and timing	In a report (PDF file), for each problem, a comparison of timing of each method for at least 10 different value of k. This needs to be shown in **both** a plot (with k on the X axis and runtime on the y axis) and a table	20
Ending values of k	in the report, a discussion on how you determined the minimum value of k. What challenges did each method (backtracking and min-conflicts) present. How did you determine for each method that the minimum value of k had been found	15
Color assignments	For each problem and for each method, the color assignment for the mininum reported value of k	10
Extra credit	implement an additional version of the backtracking search that utilizes the minimum remaining values (MRV) heuristic	10