FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE - CS161
Fall 2019

Assignment 4 – Due November 21 at 11:30pm

Please submit your homework via CCLE. The Lisp file should be named **hw4.lsp**. The answers to the questions should be given in a file named **hw4.txt**.

1. In this problem, you will write a Lisp program that converts graph coloring problems into SAT problems and use a SAT solver to solve them. We have broken the task into multiple functions and provided you with some basic code for file I/O and for gluing all the functions together (see **hw4-skeleton.lsp**).
   For each graph coloring problem, each node will be represented by a positive integer (node index). Each color is also represented by a positive integer (color index).

   Similarly, for a SAT problem, each variable is represented by a positive integer (variable index). As a result, a positive integer is used for a positive literal and a negative integer is used for a negative literal. A clause is simply a list containing the literals of the clause. A CNF formula is then simply a list of clauses.

   For example, if variable *a* has index 1, variable *b* has 2, and variable *c* has index 3, then the clause (a v ¬b v ¬c) is represented as the list (1 -2 -3). The CNF formula (a v b v c) ^ (¬a v b v ¬c) is represented as ((1 2 3) (-1 2 -3)). Of course, the order of clauses and literals in each clause does not matter.

   Here are your tasks:

   i) Write a function called node2var that takes a node index (n), a color index (c), and the maximum color index (k). This function should return the index of the propositional variable that represents the constraint: "node n receives color c" (with k colors being considered). Use the following conversion convention:

   $$variable\ index = (n-1)*k + c$$

   ii) Write a function called at-least-one-color that takes a node index (n), a color index (c), and the maximum color index (k). This function should return a clause that represents the constraint: "node n must be colored with *at least* one color whose index comes from the set {c,c+1,…,k}."

   iii) Write a function called at-most-one-color that takes the same arguments as at-least-one-color. This function should return a list of clauses that represent the constraint: "node n must be colored with *at most* one color whose index comes from the set {c,c+1,…,k}."

iv) Write a function called generate-node-clauses that takes a node index (n) and the maximum color index. This function should return a list of clauses that constrain node n to be colored with *exactly* one color whose index is in the set {1,2,…,k}.

v) Write a function called generate-edge-clauses that takes an edge (x y) and the maximum color index (k). An (undirected) edge is simply a list of two node indices. This function should return a list of clauses that prohibit nodes x and y from having the same color in the set {1,2,…,k}.

After finishing all the above parts, you should be able to convert a graph coloring problem into a SAT problem. To do so, call

(graph-coloring-to-sat <graph filename> <SAT filename> <max color index>)

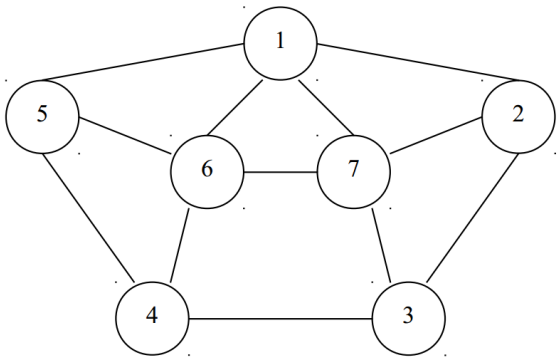
The graph filename is the name of the input graph file. The SAT filename is the name of the output file you want the program to write to. Max color index is simply the number of colors being considered in the problem.

The graph file has the following format:

- The first line contains 2 numbers. The first one is the number of vertices in the graph, the second one is the number of edges.
- Each subsequent line describes an edge. An edge is represented by two numbers—the node indices of the two nodes linked by the edge.

Now that you have a converting program working, you will use it to convert some actual graph coloring problems into SAT problems and solve them with a SAT solver.

First, consider the following graph (whose nodes are labeled with their node indices):



A graph file for this graph is also provided (graph1.txt). Convert the graph coloring problem of this graph with 3 colors into a SAT instance using the program you wrote.

Then, download the RSat SAT solver from (http://reasoning.cs.ucla.edu/rsat/). Read the manual carefully. Use RSat to solve the SAT instance obtained above. **Is the instance satisfiable?**

Do the conversion again, this time, with 4 colors. Use RSat to solve this new SAT instance. **Is the instance satisfiable?**

What do the answers of these two SAT instances tell you about the graph coloring problem of the above graph? Can you give a solution (a coloring) to the graph coloring problem of the above graph based on the results of RSat?

Now, use a similar approach to solve the graph coloring of the graph described in graph2.txt. **What is the minimum number of colors required to properly color this graph?**