15-820A, Spring 2003
Solutions to Homework 1

1 Coloring a Graph with k-colors

The goal of this homework is to gain familiarity with SAT. We will encode an interesting graph problem into a SAT problem and use modern SAT solvers like GRASP and Chaff to solve them.

The k-coloring problem is defined as follows:

> Given an undirected graph $G(V, E)$ and a natural number k, is there an assignment $color : V \rightarrow \{1, 2, \ldots, k\}$ such that $\forall (u, v) \in E.color(u) \neq color(v)$? Informally, is it possible to assign one of k colors to each node such that no two adjacent nodes are assigned the same color? If the answer is positive, we say that the graph is k-colorable.

As an aside, the map coloring problem, where you have to color each pair of adjacent countries with separate color, is just a special instance of this problem. All the maps that you can draw on a piece of paper make a *planar* graph. The famous four-color theorem says that all planar graphs can be colored with four colors.

With regards to this problem, answer the following questions.

1. Show a fomulation of the k-coloring problem as a satisfiability problem, i.e., for a given graph $G(V, E)$ and k, derive a CNF formula φ such that φ is satisfiable if and only if G is k-colorable. How many CNF variables and clauses does your CNF formula have in terms of the number of nodes $|V|$ and the number of edges $|E|$?

   Solution: There are at least two ways to encode the color information for each node in the graph. In the simplest encoding, we assign one boolean variable each node and color pair. Thus the variable $x_{i,j}, 1 \leq i \leq |V|, 1 \leq j \leq k$ is true iff the node v_i is assigned the color # j. This

generates $k|V|$ variables. We need to place three kinds of constraints on these variables for valid solutions.

At least one color: We want at least one color to be assigned to each node, thus out of all color variables, at least one should be true. This generates one clause for each node, a total of $|V|$ clauses. The clause for the node v_i looks like

$$x_{i,1} \vee x_{i,2} \vee \ldots \vee x_{i,k}$$

At most one color: The SAT checker might come up with a true assignment for more than one color variable for a given node. This means that that node was assigned more than one color. To prevent this, we have to generate CNF clauses for the constraint saying a node can be assigned at most one color. We can assert this by saying that if a node v_i is assigned the color c, then it should not be assigned another color $d, c \neq d$. This gives rise to a simple two literal clause like so: $(x_{i,c} \rightarrow \neg x_{i,d}) \equiv (\neg x_{i,c} \vee \neg x_{i,d})$. Thus for k colors, there are $k(k-1)/2$ pairs of different colors, giving rise to $k(k-1)/2$ clauses. This adds $|V|k(k-1)/2$ to the database. The clauses for a node x_i look like as follows.

$$\bigwedge_{1 \leq c < k} \bigwedge_{c+1 \leq d \leq k} (\neg x_{i,c} \wedge \neg x_{i,d})$$

Note that this constraint is strictly not required, as we are putting the constraint for different colors for two nodes connected by an edge. If a node is assigned multiple colors, we will just have to pick one color, as they will all satisfy the edge constraint.

Different colors: This is the most important constraint. It says that two nodes connected by an edge should have different colors. Thus for every edge $(v_i, v_j) \in E$, if v_i is colored with c, then v_j should not be colored with c. Which is the clause $(x_{i,c} \rightarrow \neg x_{j,c}) \equiv (\neg x_{i,c} \vee \neg x_{j,c})$. We have to place this constraint for each color c, thus an edge has a total of c such clauses, totalling to cE clauses for all edges. All the clauses for the edge (v_i, v_j) are listed below.

$$\bigwedge_{1 \leq c \leq k} (\neg x_{i,c} \vee \neg x_{j,c})$$

It should be clear that the total number of variables generated are $|V|k$ and the total number of clauses generated are $|V|+|V|k(k-1)/2+cE$.

Note that the constraint for at most one color is not strictly required. Without this constraint, the SAT checker might come up with a multiple color assignment, but all these assignments will be consistent with the edge constraint. However, there are two reasons to introduce these constraints. On one hand, they reduce the amount of post-processing required to infer color assignment from SAT checker outputs. On the other hand, they restrict the search space of the SAT checker. So if a graph does not have color assignment, then this will lead to faster conflict generation. Placing these constraints however clearly add a large number of clauses, and the SAT checker might spend more timing an assignment if there is any.

2. Write a program that accepts a textual representation of a graph $G(V, E)$ and the value k, and converts it to a CNF formula. The formats of the input files, CNF files and output files are described below.

   Solution: The program should be fairly simple, which shouldn't require storage of the whole graph, since every edge and vertex generates new set of clauses. I chose to write the program in a simple Perl script, `graph2cnf.pl`. Another Perl script (`grasp2sol.pl` reads a grasp dump, along with the number of colors and the number of nodes, and generates the corresponding color assignment file. GRASP/Chaff require the variables to be consequently numbered from 1 to the # of variables, $|V|k$ in our case. We use the following mapping to map a node number i and a color number c to a variable number.

   $$varid(i, c) = (i - 1)k + c$$

   The mapping from a variable number v to the node number and the color number is a dual.

   $$\begin{aligned} nodeid(v) &= \lfloor v/k \rfloor + 1 \\ colorid(v) &= v\%k + 1 \end{aligned}$$

3. Download from the course web page the 5 sample graphs. For each one of them, generate a SAT problem and check whether it is satisfiable with the help of a modern SAT solver (e.g. GRASP, Chaff, etc.). If the formula is satisfiable, write a table showing the color assignment

in a text file, whose format is described ahead. **Beware:** SAT solvers might take a long time to run, and in some cases, blow out of memory. If your SAT solver doesn't come up with a satisfying assignment within 3 hours, stop the process and report it as a time out.

Solution: I ran the SAT checker on the cnf files for the graphs 4,7,6,2,3 and 1 successfully. Both GRASP and Chaff failed to produce any answer for graph 5 even after 24 hours on a machine with 1GB memory. Graphs 7,2 and 3 did have a k coloring, while graphs 4, 6 and 1 did not have a valid k coloring. The following Table 1 summarizes the experiments.

Graph	SAT?	k	$\|V\|$	$\|E\|$	# vars	# clauses	runtime	
							GRASP	Chaff
graph4	N	3	11	20	33	104	0	0
graph7	Y	11	64	728	704	11592	4	0
graph6	N	8	74	301	592	4554	820	1
graph2	Y	19	179	3328	3211	92300	90	1
graph3	Y	53	128	3216	6784	346960	721	4
graph1	N	10	74	301	740	6414	-	61
graph5	?	20	864	18707	17280	539164	-	-

Table 1: Summary of experimental results. The runtimes are rounded to the nearest second. GRASP ran out of resources for graph1 and both GRASP and Chaff ran out of resource for graph5.

We can see that Chaff is significantly faster than GRASP for all the graphs.