Algorithms for Graph Coloring

About the Lab

The *Graph Coloring Problem* is: Given an undirected graph *G*, assign *colors* to the vertices such that no two adjacent vertices have the same color; the goal is to use the smallest possible number of distinct colors.

Graph Coloring is NP-Hard, which means that all known optimal algorithms run in exponential time. This lab describes three heuristic algorithms. Colors are identified by the integers 1 through *k*.

GREEDY:

```
For each vertex v:
   For each color k in increasing order:
      If (color k has not been assigned to neighbors of v):
                  Assign k to v, and
                  skip to the next vertex.
```

RANDOM:

```
Iterate the following process I times:
  Randomly permute the vertices;
  Apply GREEDY;
  If the current coloring is the best one found, save it.
```

SIMPLIFIED ITERATED GREEDY (SIG). This is an incremental-improvement algorithm. At each iteration, it re-orders the vertices, re-orders the colors, and then re-colors the graph. The number of colors used is guaranteed not to increase; if the solution does not improve after some number of tries, the algorithm reverts to an earlier solution. For a more complete description se:

- Chapter 2 of *A Guide to Experimental Algorithmics*, by C. McGeoch, Cambridge University Press, 2011.
- Joseph Culberson and Feng Luo, ``Exploring the k-colorable landscape with Iterated Greedy,'' in David S. Johnson and Michael A. Trick, eds., *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 26, 1996. This article describes the Iterated Greedy algorithm, on which the simplified version, SIG, is based.

Resources

Attached are the files in this lab:

- A C implementation of Greedy. See comments for usage notes.
- A C implementation of Random. See comments for usage notes.
- A C implementation of SIG. See comments for usage notes.
- A C program to generate random graphs containing **n** nodes and **m** edges.

Part 1

Answer the following questions about the performance of GREEDY and RANDOM:

a. How much time do they take on average, as a function of **n**, **m** and **I** (number of iterations of RANDOM)?
b. On what types of inputs are they most and least effective?
c. How does **I** affect the tradeoff between time and color count in RANDOM?
d. Is any performance difference between GREEDY and RANDOM statistically significant?

- Describe your experimental designs (what you would measure, what input classes, what input sizes, how many random trials, etc.) for addressing these questions.
- Carry out the experiment -- what did you learn about the algorithms?

Part 2

The performance of SIG is determined by six algorithm parameters:

- INITIAL: the initial coloring. Three choices are greedy (vertices in input order), random-colors (assign random colors that don't conflict), distinct (each vertex a distinct color), and random-greedy (permute vertices then color by greedy).
- MAXITER: total iterations.
- TARGET: a target color count -- stop early if it is reached.
- CWEIGHTS: a vector of six integers that assign weights to six color re-order rules (reverse, random, largest, smallest, increase, and decrease).
- VWEIGHTS: a vector of four integers that assign weights to vertex re-order rules (random, same, reverse, XX).
- RLIMIT: revert to previous best solution if no change after RLIMIT iterations.

Several input parameters and environment parameters may also be considered. Design and carry out experiments to learn how parameter levels affect algorithm performance.

a. Which three parameters are most important to performance, and which three are least important to performance?
b. Find three settings of CWEIGHTS that appear to give good performance; and three settings that appear to give bad performance. Do the same for VWEIGHTS. Are some combinations better than others?
c. Are they statistically significant?
d. Which initialization rule is best? Under what circumstances is it best?
e. What is the minimum value of MAXITER (as a function of **n** and **m**) that reliably shows improvements over the initial coloring?
f. Are those improvements statistically significant?

Deliverables

Turn in:

- Your experimental design and description of runs for part 1 and part 2 (10 pts)
- Answers to the 10 questions and show experimental evidence supporting your conclusions (30 pts – 3 pts per question)
- Visual aids and description of your runs and experiments (10 pts - 2 pts per question)