CS41 Lab 12: Randomized and Approximation Algorithms for Three-Coloring

Thursday, April 16

Recall the THREE-COLORING problem: Given a graph $G = (V, E)$, output YES iff the vertices in G can be colored using only three colors such that the endpoints of any edge have different colors. In lab 10, you saw that THREE-COLORING is NP-COMPLETE. In this lab, we'll look at several approximation and randomised algorithms for the optimization version of THREE-COLORING.

Let THREE-COLOR-OPT be the following problem. Given a graph $G = (V, E)$ as input, color the vertices in G using at most three colors in a way that maximizes the number of *satisfied* edges, where an edge $e = (u, v)$ is satisfied if u and v have different colors.

For an arbitrary graph $G = (V, E)$, let c^* denote the maximum number of satisfiable edges.

1. **Hardness of Three-Color-OPT.** Show that if there is a polynomial-time algorithm for THREE-COLOR-OPT then P = NP.

2. **Approximation Algorithm.** Give a deterministic, polynomial-time (3/2)-approximation algorithm for THREE-COLOR-OPT. Your algorithm must satisfy at least $2c^*/3$ edges.

3. **Randomised Algorithms.** Give randomised algorithms for THREE-COLOR-OPT with the following behavior:

   (a) An algorithm with expected polynomial runtime that always outputs a three-coloring that satisfies at least $2c^*/3$ edges.

   (b) An algorithm that runs in worst-case (i.e., not expected) polynomial time and produces a three-coloring such that the expected number of satisfied edges is at least $2c^*/3$.

   (c) An algorithm that runs in worst-case polynomial time, and with probability at least 99% outputs a three-coloring which satisfies at least $2c^*/3$ edges. What is the running time of your algorithm? The following inequality might be helpful: $1 - x \leq e^{-x}$ for any $x > 0$.

4. **Approximations via Reductions.** (**Extra Credit**— only work on this problem after having sketches for all previous problems). In class, you've seen many polynomial-time reductions for decision problems, and you've used them to show that several problems are NP-Complete. In this problem, you will attempt to use similar reductions to create new approximation algorithms.

   Our first reduction in class showed that Independent-Set $\leq_P$ Vertex-Cover. Given an algorithm $\mathcal{A}$ for Vertex-Cover, we created the following algorithm for Independent-Set:

   ```
   IS-ALG(G = (V,E), k)
   k' := n - k.
   z = A(G, k').
   return z.
   ```

   Now, suppose we want an approximation algorithm for Independent-Set-OPT that uses a 2-approximation algorithm $\mathcal{A}'$ for Vertex-Cover-OPT. What should your algorithm for Independent-Set-OPT do? Given the output from $\mathcal{A}'$, what should your Independent-Set-OPT algorithm output? What kind of approximation guarantee can you give?

   Design and analyze an approximation algorithm for Independent-Set-OPT. Either prove a formal guarantee for the approximation ratio of your algorithm, or give concrete evidence why that ratio is impossible (or at least hard to calculate).

   Alternately, design and analyze an approximation algorithm for Max-3-Sat using your (3/2)-approximation algorithm for Three-Color-OPT.

5. **Extra Credit.** (Only work on this problem after having sketches for all previous problems). Suppose we're somehow told that a graph is three-colorable. Could that help us color the graph? In this problem, you'll shoot for a different kind of approximation. Give a polynomial time deterministic algorithm that, given any *three-colorable* graph $G = (V, E)$, colors the graph using $O(\sqrt{n})$ colors. Note that the endpoints of each edge *must* be different colors, and you're given that its *possible* to color the graph using just three colors, but you don't know what the coloring is.

   Here are a few hints to help you along:

   (a) First, give a simple greedy algorithm that, given a graph $G = (V, E)$ such that each vertex has at most d neighbors, colors G using only $d + 1$ colors.

   (b) Second, recall the algorithm for deciding if a graph is *bipartite*.

   (c) Third, start coloring the three-colorable graph taking the vertex with the most neighbors, and coloring those neighbors using just two colors.