CS41 Homework 12

This homework is due at 11:59PM on Wednesday, December 8. Note the unusual due date. This is a 14-point homework. Write your solution using LaTeX. Submit this homework using **github** as **.tex** file. This is a **partnered homework**. You should primarily be discussing problems with your homework partner.

It's ok to discuss approaches at a high level with others. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you've done with a lab teammate *while in lab*. In this case, note (in your **homework submission poll**) who you've worked with and what parts were solved during lab.

1. **Reductions and approximations.** Recall that many problems have a decision version and an optimization version, so for example we can consider the problems

   - Independent-Set(G, k) returns yes iff there is an independent set in G of size $\geq k$,
   - Independent-Set-OPT(G) returns the size of the largest independent set in G,
   - Vertex-Cover(G, k) returns yes iff there is a vertex cover of G of size at most k,
   - VC-Opt(G) returns the size of the smallest vertex cover of G,
   - Clique(G, k) returns yes iff there is a clique in G of size k, and
   - Clique-OPT(G) returns the size of the largest clique[1] in G.

   We know that all NP-complete problems reduce to each other. It would be nice if this meant that an approximation algorithm for one NP-complete problem can be adapted easily into an equally good approximation algorithm for any other NP-complete problem.

   (a) Our first reduction in class showed that Independent-Set $\leq_P$ Vertex-Cover. Given an algorithm VC-alg for Vertex-Cover, we created the following algorithm for Independent-Set:

   IS-alg $(G = (V, E), k)$:
   1 $k' := n - k$ // where $n = |V|$
   2 $z =$ VC-alg(G, k').
   3 **return** z.

   Now, suppose we want an approximation algorithm for Independent-Set-OPT that uses a 2-approximation algorithm VC-Approx for VC-Opt. What should your algorithm for Independent-Set-OPT do? Given the output from VC-Approx, 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).

   (b) Assume we have an k-approximation algorithm for Clique-OPT where k is a constant. Can we use this to construct a decent approximation algorithm for Independent-Set-OPT? Justify your answer by designing an approximation algorithm for Independent-Set-OPT, and either proving an approximation ratio or explaining why this ratio is hard to calculate.

2. **Chromatic Number.** Consider the optimization problem ChromaticNumber, defined as follows. Given a graph $G = (V, E)$ as input, determine the smallest number k such that it is possible to k-color the graph.

   (a) Prove that ChromaticNumber is NP-hard.

[1] A **clique** is a set of nodes $C \subseteq V$ such that every two vertices $u, v \in C$ are connected by an edge: $\{u, v\} \in E$.

(b) Prove that there is no efficient $\frac{5}{4}$-approximation to ChromaticNumber unless P = NP.

(c) Prove that for any $0 < \epsilon < 1/3$ there is no efficient $1+\epsilon$-approximation to ChromaticNumber unless P = NP. Hint: recall that $\forall k > 2$, k-coloring is NP-complete.

3. **Three-Coloring, approximated.** 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 homework 11, you showed that Three-Coloring is NP-complete. In this lab, we'll look at several approximation and randomized algorithms for the optimization version of Three-Coloring.

   Let Three-Coloring-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. Given some graph G, let c^* be the maximum number of satisfied edges in a 3-coloring of G.

   Describe and analyze *randomized* algorithms for Three-Coloring-OPT with the following behavior:

   (a) 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 $2c^*/3$.

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

   (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 $x > 0$.

   (Hints: start with a very basic idea. Don't overthink the algorithm design! The challenge here is doing the analysis.)

4. **(extra challenge) Even more three-coloring.**

   Give a worst-case polynomial-time (3/2)-approximation algorithm for Three-Coloring-OPT. Your algorithm must satisfy at least $2c^*/3$ edges, where for an arbitrary input $G = (V, E)$, c^* denotes the maximum number of satisfiable edges. Your algorithm must be deterministic (i.e., cannot use randomness). Prove that your algorithm achieves a 3/2 approximation ratio.

   (Yes, this algorithm would solve all three parts of problem 3 above without even using randomness. But problem 3 is asking for algorithms which *do* use randomness, and this problem is asking for an algorithm that *does not* use randomness.)

5. **(extra challenge) Even *more* coloring! ... with not *too* many colors.**

   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 it is *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.