CS41 Homework 12: approximation

This homework is due at 11:59PM on Tuesday, December 11. Write your solution using LaTeX. Submit this homework using **github** as a **.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 partner *while in lab*. In this case, note (in your **README** file) 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 $\geq k$,
   - Max-Independent-Set(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,
   - Min-Vertex-Cover(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
   - Max-Clique(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) In class, we have seen many polynomial-time reductions for decision problems, and you have 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 VC-alg for Vertex-Cover, we created the following algorithm for Independent-Set:

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

   Now, suppose we want an approximation algorithm for Max-Independent-Set that uses a 2-approximation algorithm VC-Approx for Min-Vertex-Cover. What should your algorithm for Max-Independent-Set do? Given the output from VC-Approx, what should your Max-Independent-Set algorithm output? What kind of approximation guarantee can you give?

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

[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: $\{u, v\} \in E$.

2. **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 labs 10 and 11, we proved that THREE-COLORING is NP-COMPLETE. In this problem, we'll consider an approximation algorithm 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.

   Give a 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. Prove that your algorithm achieves this approximation ratio.

3. **(Kleinberg and Tardos, 11.9)**. Given disjoint sets A, B, C and a set $T \subseteq A \times B \times C$, a *3d matching* is a subset $M \subseteq T$ such that each element of $A \cup B \cup C$ appears at most once. The 3D-MATCHING problem is to find the largest 3d matching given sets A, B, C, and T.

   Give a polynomial-time 3-approximation algorithm for 3D-MATCHING.

4. **(extra challenge) Optimization vs Decision Problems.**

   Let B be an arbitrary optimization problem, and let A be the decision version of B.

   (a) Does it always hold that $B \leq_P A$? Answer YES or NO. Justify your response.

   (b) Does it always hold that $A \leq_P B$? Answer YES or NO. Justify your response.

5. **(extra challenge) 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.

6. **(extra challenge) Good approximations of TSP.**

   (a) Show that the triangle inequality assumption is necessary to achieve the 2-approximation for METRIC-TSP. Specifically, show that if there is a polynomial-time approximation algorithm for the general TSP with $\rho(n) = O(1)$, then P = NP. (Hint: Use contradiction. Show that we could use such an algorithm to solve VERTEX-COVER in polynomial-time.)

   (b) Show how in polynomial time we can transform one instance of TSP into an instance of METRIC-TSP. The two instances must have the same set of optimal tours.

   (c) Why does the polynomial-time transformation in part (6b) not contradict the fact proven in part (6a)?