CS 467/567, Assignment 2

Due on 12 March

This assignment is individual. Submit your solutions by email as a single document typeset to PDF. I recommend that you solve the problems by yourself with no external references, but if references are used then they must be provided and also cited in the text.

Problem 1: Approximate Maximum Clique

Let $G = (V, E)$ be an undirected graph. For $k \geq 1$ define $G^{(k)} = (V^{(k)}, E^{(k)})$ such that $V^{(K)} = \{(v_1, v_2, \ldots, v_k) : v_i \in V, 1 \leq i \leq k\}$ and $((v_1, v_2, \ldots, v_k), (w_1, w_2, \ldots, w_k)) \in E^{(k)}$ iff either $(v_i, w_i) \in E$ or $v_i = w_i$ for all $1 \leq i \leq k$.

1. Prove $|C^{(k)}| = |C|^k$, where $C^{(k)}$ and C are the maximum cliques of $G^{(k)}$ and G, respectively.
2. Argue that the existence of an approximation algorithm for finding the maximum clique with a constant approximation ratio implies the existence of a polynomial-time approximation scheme for finding the maximum clique.

Note that the answer to this question does not say anything about how easy to approximate Clique is. It is already well known (since at least 1978) that a fully polynomial-time approximation scheme cannot not exist, but how about a polynomial-time approximation scheme? The question at hand merely says that there cannot be a middle ground: the problem either has a polynomial-time approximation scheme or does not have any polynomial-time approximation algorithm with a constant approximation ratio. To date the best known approximation algorithm for Clique has an approximation ratio of $O(n(\log\log n)^2/\log^3 n)$, which is not constant and so does not imply a polynomial-time approximation scheme. There appear to be a relatively recent (2000s) result showing that the approximation ration cannot be better than $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$, but I did not have the time to check it out.

Problem 2: Linear Inequality Feasibility

I claimed in class that the original linear programming problem (an optimization problem) turns out to be no harder than the apparently simpler (decision) problem of determining whether a certain simplex is empty (that is, just finding a point in the simplex, not necessarily the optimal one). This question will prove my claim.

Given a set of m linear inequalities over n variables, the linear inequality feasibility problem asks whether there exists an assignment of the variables that satisfies all the linear inequalities simultaneously.

1. Prove that we can use an algorithm for linear programming to solve linear inequality feasibility problems. The number of variables and constraints used in the linear programming problem must be polynomial in n and m, and the slowdown must be polynomial.

2. Prove that we can use an algorithm for the linear inequality feasibility problem to solve linear programming problems. The number of variables and linear inequalities must be polynomial in the number of variables and constraints of the linear program, and the slowdown must be (you guessed it) polynomial.