

Homework Assignment 4

October 21, 2003

Due Date: November 4

Unless the problem specifies otherwise, for any NP-completeness proof you can only use the following problems for your reduction: CIRCUIT-SAT, FORMULA-SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, SUBSET-SUM, PARTITION, HAM-CYCLE, TSP.

Core Problems

1. (20 pts) Consider the following Restaurant Critic problem. There are n restaurants you need to compare. You have asked for many opinions with each one modeled by a weighted directed edge. The edge e from i to j with weight $w(e)$ indicates that j is better than i and the weight $w(e)$ indicates how strongly this belief is held. So the opinions can be modeled by a weighted directed graph $G = (V, E)$. Your goal is to produce a directed acyclic graph (i.e. a directed graph without any cycles) $G' = (V, E')$ where $E' \subseteq E$ such that $\sum_{e \in E'} w(e)$ is maximized. In other words you want to provide a non-conflicting set of recommendations with as much weight as possible.
 - (a) (12 pts) Define RC as a decision version of this problem, and then prove that RC is NP-complete. To do this you can make use of the fact that FVS (defined below) is NP-complete. A *feedback vertex set* in directed graph $G = (V, E)$ is a subset $V' \subseteq V$ such that V' contains at least one vertex from each directed cycle in G . The *feedback vertex set problem* (FVS) is: Given a directed graph G and an integer ℓ , does G have a feedback vertex set with at most ℓ vertices?
 - (b) (8 pts) Consider the following approximation algorithm for the optimization version of the Restaurant Critic problem.
 - Assign an arbitrary ordering to the vertices in V .
 - Let $E_1 = \{(i, j) \mid i < j\}$ and let $E_2 = \{(i, j) \mid i > j\}$. That is if you put the vertices in order from left to right, E_1 are the edges that go to the right and E_2 are the edges that go to the left.
 - If $\sum_{e \in E_1} w(e) \geq \sum_{e \in E_2} w(e)$ then return E_1 . Else return E_2 .

Prove the above is a 2-approximation.

2. (10 pts) Given a set of m linear constraints over n variables, the **integer-programming (IP)** decision problem asks whether there is an *integer* n -vector x giving values for each of the n variables such that all the constraints are satisfied. Prove that IP is NP-complete.

Hint: The arithmetic expression $1 - x$ performs a negation when x is 0 or 1.

3. (15 pts) The CS department needs to select a set of courses to offer during Summer 2003. All students are invited to submit a list of courses which they would like to take. There are s students who submit such a list where student i gives list L_i . The goal is to find a set of courses S to offer in Summer 2003 where $|S|$ is minimized such that S contains at least one element from L_i for $i = 1, \dots, s$. You are to either give a polynomial time algorithm for this problem (and prove it is correct), OR state a decision version of this problem and prove it is NP complete.

4. (15 pts) Recall the following two polynomial-time reductions that we went over in class last week when showing that INDEP-SET is NP-complete.
- CLIQUE \leq_p INDEP-SET. The transformation function used here takes the CLIQUE input $\langle G = (V, E), k \rangle$ and converts it to INDEP-SET input $\langle \overline{G}, k \rangle$. It can be proven that G has an clique of k vertices if and only if \overline{G} has an independent set of k vertices.
 - VERTEX-COVER \leq_p INDEP-SET. The transformation function used here takes the vertex cover input $\langle G = (V, E), k \rangle$ and converts it to independent set input $\langle G, |V| - k \rangle$. It can be proven that G has a vertex cover of k vertices if and only if G has an independent set of $|V| - k$ vertices.

It is not hard to see that IF there were a polynomial time algorithm that returned a maximum size independent set, then the approximation algorithms described below, would optimally solve the clique and vertex cover optimization problems. Instead, suppose you were given a c -approximation algorithm A (for some constant c) for computing the largest independent set in a graph G . That is, on input G , A outputs an independent set in G of at least opt/c vertices where opt is the number of vertices in the largest independent set in G . (No such algorithm is really known, but assume it existed just for the sake of this homework problem).

- (a) Consider the following approximation algorithm, A_{CLIQUE} , for the clique problem. Let G be the input for A_{CLIQUE} . First compute \overline{G} . Next run A on \overline{G} to obtain the independent set $V' \subseteq V$. A_{CLIQUE} returns V' as the clique approximation. Prove or disprove: A_{CLIQUE} is a c -approximation algorithm for finding a maximum clique in G .
- (b) Consider the following approximation algorithm, A_{VC} , for the vertex cover problem. Let G be the input for A_{VC} . Run A on G to obtain the independent set $V' \subseteq V$. Finally, A_{VC} returns $V - V'$ (i.e. all vertices not in V'). Prove or disprove: A_{VC} is a c -approximation algorithm for finding a minimum vertex cover.

Advanced Problems (Required only for CS 539T students)

5. (15 pts) Consider the optimization version of MP-SCHED (from HW 3). For $1 \leq i \leq m$, let A_i contain the set of jobs that will run on machine i in the solution and let $\ell(a)$ be the time required by job a . (So $A = A_1 \cup A_2 \cup \dots \cup A_m$). The goal in the optimization version is to find a partition A_1, \dots, A_m that minimizes the time when the last job completes. That is, you want to minimize $\max_{A_i} \sum_{a \in A_i} \ell(a)$. Here is an approximation algorithm for this problem. You are given a list of jobs in an arbitrary order. Whenever a machine becomes available, the next job on the list is assigned to begin processing on that machine. Prove the best approximation ratio you can for this algorithm.

Here is some guidance to get you started. For $a \in A$ let $s(a)$ be the time a is started in the approximate solution S . Let job k be the job to finish last in S . Let C be the cost of S and let C_* be the cost of the optimal solution. Think about the relationship between C and C_* in terms of $s(k)$ and $\ell(k)$. Also think about what you can say about C_* with respect to the average job length. Finally, give the best upperbound you can for $s(k)$ and then put all of these pieces together. *Hint: You can do better than a 2-approximation.*