

Exam 2

November 8, 2001

For NP-hardness proofs you can only use the following problems for your reduction: SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, SUBSET-SUM, PARTITION, HAM-CYCLE, TSP.

1. (25 pts) For each of the following problems either give a polynomial-time algorithm to find an optimal solution *by formulating it as a linear program* or prove that the decision version of the problem (which you should clearly state) is NP-complete.
 - (a) The CS department needs to select a set of courses to offer during Summer 2002. 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 2002 where $|S|$ is minimized such that S contains at least one element from L_i for $i = 1, \dots, s$.

- (b) A governmental planning agency for California wishes to determine the sources to purchase fuel for use by n depots from among m bidders. The maximum quantity offered by bidder i is a_i gallons at c_i dollars per gallon. The demand at depot j is b_j gallons. Let d_{ij} be the cost per gallon for the delivery from bidder i to depot j . The goal is to meet the demands at all j depots with the minimum possible total cost (fuel and shipping).

2. (15 pts) Prove that the following multiprocessor scheduling problem, MP-SCHED is NP-hard. As input you are given a set A of jobs where for $a \in A$ it has length $\ell(a)$. You are also given an integer m which is the number of processors and an integer deadline d . Each job can run on any of the m machines but only one job can run at a time on a given machine. The question is whether or not there is a partition of the jobs into m sets $A = A_1 \cup A_2 \cup \dots \cup A_m$ where A_i is the set of jobs that will run on machine i such that for $i = 1, \dots, m$, $\sum_{a \in A_i} \ell(a) \leq d$. In other words, is there a schedule in which all jobs are processed by time d ?

3. (10 pts) Consider the optimization version of MP-SCHED. The goal is to minimize 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. For $a \in A$ let $s(a)$ be the time a is started. Recall that $\ell(a)$ is the length of job a . Let job k be the job to finish last. Let C be the cost of the solution output by this approximation algorithm and let C_* be the optimal solution. Here are some useful facts:

- By definition, $C = s(k) + \ell(k)$.
- $C_* \geq \ell(k)$ since job k must be run.
- $C_* \geq (\sum_{a \in A} \ell(a))/m$ since this represents the best possible situation when all machines complete processing at exactly the same time.
- Consider $A' = A - \{k\}$. The average time a machine is finished for A' is $(\sum_{a \in A'} \ell(a))/m$. Since job k will start as soon as the first of the m machines is finished with A' it follows that $s(k) \leq (\sum_{a \in A'} \ell(a))/m$.

Put these pieces together to obtain the best approximation ratio you can for this algorithm.

THE CS 441T EXAM ENDS HERE. (If a CS 441T student does the next problem we will note it as extra credit, but no points will be added to your total. Hence CS 441T students should do all they can on problems 1-3 before even considering problem 4.)

THIS PROBLEM IS REQUIRED ONLY FOR CS 539 STUDENTS.

4. (10 pts) In this problem you will analyze an approximation algorithm for the knapsack problem. Let the input be v_1, \dots, v_n (the item values), w_1, \dots, w_n (the item weights), and W (the knapsack capacity). The goal is to find a set of items S such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i$ is maximized. Here is an approximation algorithm. First sort the items so that

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \dots \geq \frac{v_n}{w_n}$$

We now use the following variation of the greedy algorithm. Let S_{greedy} be the answer computed by the greedy algorithm. Namely, S_{greedy} chooses the first k objects that fit. That is, k satisfies

$$\sum_{i=1}^k w_i \leq W \quad \text{and} \quad \sum_{i=1}^{k+1} w_i > W.$$

The approximation algorithm returns the better of S_{greedy} or the single item $k+1$. Prove the best approximation ratio you can for this algorithm. Some useful expressions to consider are $v_1 + v_2 + \dots + v_k + v_{k+1}$ and the value of the approximate solution which is $\max\{v_1 + \dots + v_k, v_{k+1}\}$.

Additional Work for Problem _____

Problem	Points Possible	Points Received
1a	25	
1b		
2	15	
3	10	
4*	10	
total	50 + 10	