

## Homework Assignment 4

October 18, 2001

Due Date: November 1

## Core Problems

1. (10 pts) Let the input for KNAPSACK be  $v_1, \dots, v_n$  (the item values),  $w_1, \dots, w_n$  (the item weights),  $W$  (the knapsack capacity), and  $V$  (the desired value). The question is whether or not there is a subset of the  $n$  items with weight at most  $W$  and value at least  $V$ . All of the  $v_i, w_i, W$  and  $V$  are non-negative integers.

Prove that the KNAPSACK problem is NP-Complete. You must use one of the following NP-complete problems for your reduction: CIRCUIT-SAT, SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, SUBSET-SUM, HAM-CYCLE, TSP.

2. (10 pts) In proving that  $\text{CLIQUE} \leq_p \text{VERTEX-COVER}$  for a given the input  $\langle G = (V, E), k \rangle$  for CLIQUE we converted it to the input  $\langle \overline{G}, |V| - k \rangle$  for VERTEX-COVER. We then proved that  $G$  has a clique of  $k$  vertices if and only if  $\overline{G}$  has a vertex cover of  $|V| - k$  vertices.

Recall that we have a 2-approximation algorithm, call it  $A$ , for the optimization version of the vertex cover problem. We now propose a scheme to use  $A$  to create an approximation algorithm for clique (that is, given input  $G$ , the goal is to find a set of vertices that form a maximum-sized clique). First compute  $\overline{G}$ . Next run  $A$ , the vertex cover approximation algorithm, on  $\overline{G}$  to obtain the approximate vertex cover  $V' \subseteq V$ . Then output  $V - V'$  (i.e. all vertices not in  $V'$ ) as the clique approximation.

Let  $C_*(G)$  be the size of the maximum clique in  $G$  and let  $C(G)$  be the size of the clique output by the above approximation algorithm. Do ONE of the following two tasks:

- Give the smallest constant  $c$  for which you can prove that for all inputs  $G$  to CLIQUE that  $\frac{C_*(G)}{C(G)} \leq c$ . That is, prove that the resulting approximation has an approximation ratio of  $c$ .
  - Prove that for any constant  $c$  there exists an input  $G$  for which  $\frac{C_*(G)}{C(G)} > c$ .
3. In this problem we consider the relationship between the traveling salesman problem with an unrestricted cost function (TSP) and the traveling salesman problem when the cost function must satisfy the triangle inequality (TSPWTI).

Here is a polynomial time reduction from the optimization version of TSP to the optimization version of TSPWTI. Let  $G$  be the TSP input in which there are vertices  $V$  and for each pair  $u, v \in V$ , there is an edge with weight  $w(u, v)$ . Let  $W$  be the largest edge weight. We construct the TSPWTI input  $G'$  as follows.  $G'$  has the same vertices as in  $G$ . In  $G'$  for each pair  $u, v \in V$ , there is an edge with weight  $w'(u, v) = W + w(u, v)$ .

- (a) (2 pts) Argue that the edge weights in  $G'$  satisfy the triangle inequality. That is for an arbitrary three vertices  $x, y, z$  prove that  $w'(x, y) + w'(y, z) \geq w'(x, z)$ .

- (b) (3 pts) Since  $G$  and  $G'$  have the same set of vertices a tour  $T$  in  $G$  is also a tour in  $G'$ . Prove that  $T$  is an optimal tour in  $G$  if and only if it is an optimal tour in  $G'$ .
- (c) (5 pts) Suppose one uses this reduction to create the following approximation algorithm  $A$  for TSP. Given TSP input  $B$ , apply the given transformation to create  $G'$ . Now apply the 2-approximation given in class (and in the text) to the the TSPWTI instance  $G'$ . Output the tour computed by this approximation algorithm (for  $G'$ ) as the tour for  $G$ . Prove the best ratio bound you can for this approximation algorithm for the general TSP problem.
- (d) (5 pts) Assuming  $P \neq NP$ , we proved that for any constant  $c \geq 1$  there is no polynomial time approximation algorithm with ratio  $c$  for the general TSP problem. Explain why your answer for part (c) does not contradict this result.
4. (10 pts) Consider the following optimization problem. You have a single machine that can run one job at a time. For the input there are  $n$  jobs where the  $i$ th job,  $J_i$  is specified by an earliest start time (or release time)  $r_i$ , a length  $\ell_i$ , and a delivery time  $q_i$ . So, job  $i$  can be scheduled starting at any time  $s_i \geq r_i$ , and would finish using the machine at time  $s_i + \ell_i$  (and at this point the machine is available for another job) and finally, arrives at its destination at time  $s_i + \ell_i + q_i$ . The goal is to find a schedule so as to minimize the time  $t$  at which all jobs have had their results delivered. That is, if  $s_i$  is the start time for job  $i$ , then the goal is to minimize  $\max_i (s_i + \ell_i + q_i)$ .

Consider the following approximation algorithm: Whenever the machine is available, schedule an arbitrary job (if there is one) which is past its release time. You are to prove that this yields a 2-approximation. Here's a little guidance. Let  $\sigma$  be the schedule produced by the approximation algorithm and let  $\sigma_*$  be the optimal schedule. Let job  $k$  be the last job delivered by the approximation algorithm. Thus the cost of  $\sigma$  (denoted by  $cost(\sigma)$ ) is  $s_k + \ell_k + q_k$  since that is when the last delivery is completed.

- (a) Briefly argue that  $cost(\sigma_*) \geq L = \sum_{i=1}^n \ell_i$
- (b) Briefly argue that  $cost(\sigma_*) \geq r_k + \ell_k + q_k$
- (c) Finally, complete the proof that the proposed algorithm is a 2-approximation algorithm. *Hint: Use the fact that between time  $r_k$  when job  $k$  was released and  $s_k$  when job  $k$  was started, the machine must have been in use since the approximation algorithm never left the machine idle when a job was available.*
5. (15 pts) Consider the following problem. Each job consists of a set of *operations* whose processing cannot overlap in time. The set of all operations is denoted  $\{O_1, \dots, O_k\}$ ; each operation  $O_k$  belongs to some job  $J_j$  and must be processed on a specific machine  $M_i$ . *However, the operations of a job can be processed in any order.* The goal is to minimize the time at which all operations have been completed.

You are to give a 2-approximation algorithm for this problem. (Be sure to clearly describe your algorithm and then prove that it has an approximation ratio of 2.)

*Hint: The algorithm need not be very sophisticated. Here are some definitions that you might find useful. Let  $P_{max}$  be the maximum processing time required by any job and let  $\Pi_{max}$  be*

the maximum processing time required on any machine. Finally, let  $M_i$  be the machine that finishes last in the approximate solution and let  $J_j$  be the job whose operation  $O_k$  was the last to run on machine  $M_i$ .

**ALTERNATE PROBLEM 5:** Implement (in any language) the fully polynomial-time approximation scheme (PTAS) given in the text for the optimization version of the subset sum problem. You are to output the actual solution (i.e. the list of the elements selected) along with their sum.

On the web page is a sample data file. The first line contains  $n$ , the cardinality of the set  $S$  and the integer  $t$ . Then second line contains  $n$  integers (the elements of  $S$ ).

Provide your code as well as your output for the data set on the course web page for  $\epsilon = .2$ ,  $\epsilon = .1$  and  $\epsilon = .01$ .

## Advanced Problems (Required only for CS 539T students)

6. (15 pts) Consider the following problem. You are given a simple undirected weighted graph  $G$  and an integer  $f \leq |V|$ . Your goal is to pick the location for  $f$  vertices of  $G$  as fire station locations so that the *length of the shortest path* between any vertex in  $G$  and its closest fire station is the minimum possible. From this optimization problem we create the following decision problem that we will call the FIRE-STATION-PLACEMENT problem:

Given an undirected weighted graph  $G$ , an integer  $f \leq |V|$ , and a real number  $d \geq 0$ , is it possible to select  $f$  vertices of  $G$  as fire station locations so that the length of the shortest path between any vertex in  $G$  and its nearest fire station is at most  $d$ .

You are to do **ONE** of the following two tasks:

- Prove that there is a polynomial time algorithm for the FIRE-STATION-PLACEMENT problem. (In other words give an algorithm that you can prove will give the correct answer and that runs in polynomial time.)
- Prove that FIRE-STATION-PLACEMENT is NP-complete.

*Note: If you do both of the above I will give you an A+ for the course. ;-)*

7. (15 pts) Consider the following scheduling problem. You are given  $n$  jobs where job  $i$  is specified by an earliest start time  $s_i$  and a processing time  $p_i$ . In homework 1, problem 7 we consider a preemptive version of this problem and gave a greedy algorithm to give an optimal preemptive schedule.

In this problem we consider the non-preemptive version of this scheduling problem. Here a job CANNOT be suspended but rather must be performed in a contiguous time interval. Consider the following heuristic for the non-preemptive problem: schedule the jobs in the order in which they complete in an optimal preemptive schedule starting each job as soon as the one before it completes. You are to prove that this algorithm is a 2-approximation algorithm.