

Homework Assignment 4

October 24, 2002

Due Date: November 7

Core Problems

- (10 pts) Prove that the following multiprocessor scheduling problem, MP-SCHED is NP-hard. You can only use the following problems for your reduction: SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, SUBSET-SUM, PARTITION, HAM-CYCLE, TSP.

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 ?

- (10 pts) In proving that CLIQUE \leq_p 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 $\frac{C_*(G)}{C(G)} > c$.
- (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: $\max\{v_1 + \dots + v_k, v_{k+1}\}$.

4. (10 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 Π_{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 .

5. (15 pts) 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.

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 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, we considered 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.

7. (15 pts) Consider the optimization version of MP-SCHED (from problem 1). 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. 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.