TU Berlin
Institut für Mathematik
Prof. David Williamson
Jannik Matuschke

Approximation Algorithms (ADM III)
Winter term 2010

Problem Set 2

(due date: *November 10, 2010*)

Exercise 2.1 **4 points**

The k-*suppliers* problem is similar to the k-center problem given in class. The input to the problem is a positive integer k, and a set of vertices V, along with distances d_{ij} between any two vertices i, j that obey the same properties as in the k-center problem. However, now the vertices are partitioned into *suppliers* $F \subseteq V$ and *customers* $D = V - F$. The goal is to find k suppliers such that the maximum distance from a supplier to a customer is minimized. In other words, we wish to find $S \subseteq F$, $|S| \leq k$, that minimizes $\max_{j \in D} d(j, S)$. Give a 3-approximation algorithm for the k-suppliers problem.

Exercise 2.2 **3 points**

Show that for any input to the problem of minimizing the makespan on identical parallel machines for which the processing requirement of each job is more than one-third the optimal makespan, the longest processing time rule computes an optimal schedule.

Exercise 2.3 **5 points**

We consider scheduling jobs on identical machines as in class, but jobs are now subject to *precedence constraints*. We say $i \prec j$ if in any feasible schedule, job i must be completely processed before job j begins processing. A natural variant on the list scheduling algorithm is one in which whenever a machine becomes idle, then any remaining job that is *available* is assigned to start processing on that machine. A job j is available if all jobs i such that $i \prec j$ have already been completely processed. Show that this list scheduling algorithm is a 2-approximation algorithm for the problem with precedence constraints.

Exercise 2.4 **6 points**

In this problem, we consider a variant of the problem of scheduling on parallel machines so as to minimize the length of the schedule. Now each machine i has an associated speed s_i, and it takes p_j/s_i units of time to process job j on machine i. Assume that machines are numbered from 1 to m and ordered such that $s_1 \geq s_2 \geq \cdots \geq s_m$. We call these *related* machines.

(a) A ρ-*relaxed decision procedure* for an scheduling problem is an algorithm such that given an instance of the scheduling problem and a deadline D either produces a schedule of length at most $\rho \cdot D$ or correctly states that no schedule of length D is possible for the instance. Show that given a polynomial-time ρ-relaxed decision procedure for the problem of scheduling related machines, one can produce a ρ-approximation algorithm for the problem.

(b) Consider the following variant of the list scheduling algorithm, now for related machines. Given a deadline D, we label every job j with the slowest machine i such that the job could complete on that machine in time D; that is, $p_j/s_i \leq D$. If there is no such machine for a job j, it is clear that no schedule of length D is possible. If machine i becomes idle at a time D or later, it stops processing. If machine i becomes idle at a time before D, it takes the next job of label i that has not been processed, and starts processing it. If no job of label i is available, it looks for jobs of label $i+1$; if no jobs of label $i+1$ are available, it looks for jobs of label $i+2$, and so on. If no such jobs are available, it stops processing. If not all jobs are processed by this procedure, then the algorithm states that no schedule of length D is possible.

Prove that this algorithm is a polynomial-time 2-relaxed decision procedure.

Exercise 2.5 **5 points**

In the *maximum coverage problem*, we have a set of elements E, and m subsets of elements $S_1, \ldots, S_m \subseteq E$, each with a nonnegative weight $w_j \geq 0$. The goal is to choose k elements such that we maximize the weight of the subsets that are covered. We say that a subset is covered if we have chosen some element from it. Thus we want to find $S \subseteq E$ such that $|S| = k$ and that we maximize the total weight of the subsets j such that $S \cap S_j \neq \emptyset$.

(a) Give a $(1 - \frac{1}{e})$-approximation algorithm for this problem.

(b) Show that if an approximation algorithm with performance guarantee better than $1 - \frac{1}{e} + \epsilon$ exists for the maximum coverage problem for some constant $\epsilon > 0$, then every *NP*-complete problem has a $O(n^{O(\log \log n)})$ time algorithm (Hint: Recall the hardness theorems about the set cover problem.)