EECS 279: Approximation Algorithms — Release: 09-16-2014

Homework 2-A — Due: 3:00pm, Thursday, 9-25-2014

Make your solutions concise and formal. Your goal is to convince me that you know the solutions. You are highly encouraged to typeset your solutions in Latex. See the course website for further homework policies. Please write down your collaborators names and references if any.

1. In the Knapsack problem we are given a set U of n items where each item i has size s_i and profit p_i. The goal is to find a maximum profit subset of items that can fit in a knapsack of size B.

   In the class, we learned that the algorithm that guesses the $\lceil 1/\epsilon \rceil$ most profitable items in the optimal solution and runs the greedy algorithm on the remaining Knapsack problem yields a PTAS. We would like to design another PTAS – first guess the set X of $\lceil 1/\epsilon \rceil$ largest items in the optimal solution and run the greedy algorithm on the remaining Knapsack problem by packing items i in $U \setminus X$ in non-increasing order of p_i/s_i until the knapsack runs out of space. Show that this algorithm is a PTAS for the Knapsack problem.

   Note: In fact, this algorithm as described is not a PTAS. But by making a small change we can indeed obtain a PTAS: First, consider every set X of *up to* $\lceil 1/\epsilon \rceil$ items, then run the greedy algorithm on the remaining Knapsack problem by packing items smaller than a certain size (that we guess) in X in non-increasing order of p_i/s_i until the knapsack runs out of space.

   There is an instance where if we pack all $\lceil 1/\epsilon \rceil$ largest items in OPT, then we may miss a very profitable small item in the following greedy procedure, failing to obtain a PTAS. However, if we discard one of the $\lceil 1/\epsilon \rceil$ largest items, we can pack the last small item we couldn't otherwise.

   Due to the above reason, every student will get full points for this problem.

2. In the Bin Packing problem, we are given as input a set of n items where item i has size $s_i \in (0, 1]$. Our goal is to pack in the minimum number of bins of size 1.

   In the class, we learned a PTAS where we partitioned items into groups of an equal size (except the last one), and each item in B_k is rounded to the size of the smallest item in B_{k-1}, for all $k \geq 2$. Instead of this, partition items of exponentially increasing sizes as we did for the min-makespan problem on identical machines – more precisely, if an item has size $s_i \in (\epsilon(1+\epsilon)^j, \epsilon(1+\epsilon)^{j+1}]$, then it is rounded to size $\epsilon(1+\epsilon)^{j+1}$. This rounding also reduces the number of distinct item sizes. However, this rounding does not lead to a PTAS. Explain why.

3. We consider the minimum makespan scheduling problem when given n jobs of sizes $p_1, p_2, \ldots, p_j, \ldots, p_n$ for related machines where machines have different speeds, $s_1 \geq s_2 \geq s_3 \geq \ldots \geq s_m$. Machine i with speed s_i finishes workload L_i at time L_i/s_i. The goal is to assign jobs to machines such that all jobs are finished as early as possible, i.e. $\max_i L_i/s_i$ is minimized. Give an $O(1)$-approximation for this problem.

   Hints:

(a) Consider the following algorithm. Guess the optimal value, OPT. Assign jobs in decreasing order of their sizes. When considering a job j, assign j to the slowest machine i where the current load of the machine plus job j does not exceed $s_i \cdot c \cdot \mathrm{OPT}$, where c is some constant you can fix. That is, the makespan of the machine will not exceed $c \cdot \mathrm{OPT}$ if you place job j on the machine.

(b) Each job j cannot be assigned to machine i such that $s_i \cdot \mathrm{OPT} < p_j$ in the optimal solution. Why? This will define a subset of machines that can possibly schedule the job.

(c) Think how to modify the 2-approximation we learned in the class to this setting where machines have different speeds.