Fall 2013, CS 583: Approximation Algorithms

Homework 2
Due: 10/07/2013

Instructions and Policy: Each student should write up their own solutions independently. You need to indicate the names of the people you discussed a problem with; ideally you should discuss with no more than two other people.

Solve as many problems as you can. I expect at least four.

Please write clearly and concisely - clarity and brevity will be rewarded. Refer to known facts as necessary. Your job is to convince me that you know the solution, as quickly as possible.

Problem 1 Multi-processor scheduling: given n jobs $J_1, \ldots, J_n$ with processing times $p_1, p_2, \ldots, p_n$ and m machines $M_1, M_2, \ldots, M_m$. For identical machines greedy list scheduling that orders the jobs in non-increasing sizes has an approximation ratio of 4/3. See Section 2.3 in the Shmoys-Williamson book.

Now consider the problem where the machines are not identical. Machines M_j has a speed s_j. Job J_i with processing time p_i takes p_i/s_j time to complete on machine M_j. Give a constant factor approximation for scheduling in this setting to minimize makespan (the maximum completion time). (Hint: consider jobs in decreasing sizes. Assuming $p_1 \geq p_2 \geq \ldots \geq p_n$ and $s_1 \geq s_2 \geq \ldots s_m$, show that $OPT \geq \max_{i \leq m}(\sum_{j \leq i} p_j / \sum_{j \leq i} s_j.)$

Problem 2 In the Generalized Assigment problem, you are given n jobs, and m machines/bins. For each job i and machine j, there is a size s_{ij} that job i occupies on machine j. (Note that the s_{ij}s may be completely unrelated to each other.) A feasible assignment is one in which each jobs is assigned to some machine.

The *makespan* of an assignment is the maximum, over all machines i, of the total size (on i) of jobs assigned to it. Give a PTAS for the problem of minimizing makespan when the number of machines m is a constant. Use the following scheme.

- Guess all the "big" items and their assignments.
- Write an Linear Program for assigning the residual "small" items.
- Show that a basic feasible solution (a vertex solution) for the linear program has at most m fractionally assigned jobs. Use this to assign them greedily.

Problem 3 In the uniform-capacity Resource/Bandwidth Allocation Problem, the input is a path $P = \{v_1, v_2, \ldots v_n\}$, where v_i is adjacent to v_{i+1}; an integer capacity c; and a set of demand requests $\mathcal{R} = \{R_1, \dots, R_m\}$. Each request R_h consists of a pair of vertices v_i, v_j, and an integer demand d_h; this is to be interpreted as a request for d_h units of capacity from v_i to v_j. Note that there can be multiple requests between the same pair of nodes. The goal is to find a largest subset of requests, $\mathcal{R}$, that can be satisfied simultaneously; that is, the total demand of satisfied requests going through any edge v_i, v_{i+1} should not exceed the capacity c.

(Note that when the path P is a single edge, this problem is equivalent to KNAPSACK.)

1. Assume $c = 1$ and all requests are for one unit of demand. In this case we are asking for the largest independent set in an interval graph. Consider the algorithm that orders the requests in increasing order of *length* and greedily selects them while maintaining feasibility. Show that this algorithm is a 1/2-approximation using the technique of dual-fitting. Write an LP and find a feasible dual to the LP and relate the solution output by the greedy algorithm to the dual value.

2. Consider the weighted version, where each request R_h also has a profit/weight p_h, and the goal is to find a maximum-profit set of requests that can be satisfied simulataneously. (Note that an optimal solution may have overlapping requests since the demands are now varying.) Write a Linear Program for this problem, and show that the LP has constant integrality gap:

   Hint 1: If you randomly round each request independently, with probability proportional to "how much" the request is selected by the LP, show that the expected profit of the integral "solution" is large, though the solution obtained may not be feasible.

   Hint 2: Scale down all probabilities by a constant factor (say 10), and round independently. Let S be the set of selected requests. Now, initialize set S' to be empty, and order requests in S by their left endpoint, from left to right. In this order, select $s \in S$ for S' if it can be added to S' without violating feasibility. Show that the probability a request $s \in S$ is selected for S' conditioned on it being in S is a constant. The analysis is similar to the scheme for k-sparse PIPs.

Problem 4 In this problem, we solve MAXIMUM INDEPENDENT SET (MIS) in another family of graphs, the intersection graphs of disks in the Euclidean plane: Given a set of disks in the plane, construct a graph by creating a vertex for each disk, and connecting two vertices by an edge if the corresponding disks intersect. Give a PTAS for MIS problem in these graphs, assuming all disks have unit radius.

Hint 1: Consider a grid of lines spaced $\frac{1}{\epsilon}$ units apart. If no disks of an optimal solution intersect these grid lines, can you find an *exact* algorithm with running time polynomial in n for any fixed ϵ?

Hint 2: Consider a grid with *random* offset: Take a grid of lines spaced $\frac{1}{\epsilon}$ apart, such that the origin is at the intersection of a horizontal and vertical grid line. Pick a shift/offset L uniformly at random from $[0, \frac{1}{\epsilon})$, and shift the grid vertically and horizontally by an distance L. (Equivalently, consider the grid of spacing $\frac{1}{\epsilon}$ such that the point (L, L) is at the intersection of two grid lines.) What is the probability that a disk is intersected by a grid line? Can you give a deterministic approximation scheme?

Note: There is a PTAS for the problem, even if the disks are allowed to have different sizes. Do you see how to obtain a PTAS? For more information about geometric approximation, see the Chapter 11 in Vazirani book or Chapter 10 in Shmoys-Williamson book or the upcoming book of our own faculty member Prof. Har-Peled which is available on his website.

Problem 5 We consider the non-metric facility location problem. We are given a set of clients $\mathcal{D}$ and a set of facilities $\mathcal{F}$. Each facility $i \in \mathcal{F}$ has a cost $f_i \geq 0$ for opening it. There is a non-negative cost $c(i, j)$ to connect client j to facility i. The goal is to open a set of facilities and connect each client to an open facility so as to minimize the sum of the facility opening cost and the connection costs of the clients. In the non-metric version we do not assume that the $c(i, j)$ values form a metric so they can be arbitrary non-negative numbers.

- Show that an $\alpha(|\mathcal{D}|)$-approximation for non-metric facility location implies an $\alpha(n)$-approximation for the set cover problem with n elements.
- Given an $O(\log |\mathcal{D}|)$-approximation for the non-metric facility location via the LP relaxation we discussed in class with y_i variables for the facilities and $x_{i,j}$ variables for connecting clients to facilities. You can use the following hints.
  - Let $\bar{y}, \bar{x}$ be an optimum solution for the LP realxation. For each client j define $\alpha_j = \sum_i c(i, j)x_{i,j}$ to be the LP connection cost.
  - Reduce the problem to set cover (via the standard LP relaxation for it) as follows. For each client j restrict it to be connected to only those facilities i where $c(i, j) \leq 2\alpha_j$.

Problem 6 Recall the Maximum Independent Set (MIS) problem for the intersection graphs of disks in the Euclidean plane: Given a set of disks in the plane, construct a graph by creating a vertex for each disk, and connecting two vertices by an edge if the corresponding disks intersect. Assume as before that all disks have unit radius. We say that a solution X to the Independent Set problem is s-optimal if we cannot get a larger independent set by removing at most s vertices from X and adding at most $s + 1$ vertices from $V - X$. Consider the following local search algorithm for Independent Set in unit disk graphs: Start with an arbitrary solution, and as long as the current solution is not s-optimal, find a larger independent set by removing at most s vertices and adding at most $s + 1$. Prove that there is a (small) constant s such that the local search algorithm gives a constant-factor approximation. Try to make the approximation ratio as small as you can.

Problem 7 Problem 2.13 part (a) from the Shmoys-Williamson book.