Spring 2018, CS 583: Approximation Algorithms

Homework 2

Due: 02/22/2018

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. You may be able to find solutions to the problems in various papers and books but it would defeat the purpose of learning if copy them. You should cite all sources that you use.

Read through all the problems and think about them and how they relate to what we covered in the lectures. Solve as many problems as you can. Please submit your solutions to at least 4 problems.

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 Read the definition of inductively k-independent graphs in the lecture note. Prove that the Greedy algorithm that considers the vertices in the inductive k-independent order gives a $\frac{1}{k}$-approximation for the maximum independent set problem.

Problem 2 Consider unweighted interval scheduling problem where we are given a collection of open intervals on the line and the goal is to find a maximum cardinality subset of non-overlapping intervals. There is a simple greedy algorithm that gives an optimum solution. 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.

Hint: Charge each interval of the solution to two points to construct a feasible dual solution.

Problem 3 In the uniform-capacity Resource/Bandwidth Allocation Problem, the input is a path $P = \{v_1, v_2, \dots v_n\}$, 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.)

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. You can apply the randomized rounding with alteration technique that we saw for intervals.

Problem 4 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 1: Consider the jobs in decreasing sizes and minimize makespan time in each step.

Hint 2: Suppose the job with maximum completion time is i-th job. What can we argue about the processing time of jobs on each machine in terms of p_i and the completion time?

Hint 3: 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 5 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. (Hint: Rank Lemma)

Problem 6 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 in the optimum solution is intersected by a grid line?

Hint 3: Derandomize your algorithm by reducing the number of choices of L.

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 Sariel Har-Peled's book and also Chapter 11 in Vazirani book and Chapter 10 in Shmoys-Williamson book.