Fall 2021, CS 583: Approximation Algorithms

Homework 2

Due: 09/30/2021 in Gradescope

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 and write in your own words.

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 solutions to at least four problems. Some problems are closely related so it may benefit you to solve them together or view them as parts of an extended problem.

Please write clearly and concisely - clarity and brevity will be rewarded. Refer to known facts as necessary.

Problem 1 Read the definition of inductively k-independent graphs in the lecture notes.

- 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.
- **Extra Credit:** Obtain an $\Omega(\frac{1}{k})$-approximation for the weighted case.

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.

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.

Problem 4 We saw an $\Omega(1/\log n)$-algorithm for Rectangle Independent Set. Use this algorithm as a black box and generalized the approach to obtain an $\Omega(1/\log^2 n)$-approximation algorithm for the problem of finding the maximum independent set problem in the intersection graph of a given set of axis-aligned boxes in three dimensions. More generally, if you have an α-approximation algorithm for rectangle independent set, show that you can obtain an $\Omega(\alpha/\log n)$-approximation for boxes in three dimensions.

Problem 5 In this problem we discuss packing integer programs (PIPs) and the notion of *width*. A packing integer program (PIP) is an integer program of the form $\max\{wx \mid Ax \leq b, x \in \{0,1\}^n\}$ where w is a $1 \times n$ non-negative vector and A is a $m \times n$ matrix with entries in $[0,1]$ and b is a vector such that $b_i \geq 1$ for $1 \leq i \leq m$. One way to view this is as an m-dimensional knapsack problem: we want to pack the most profitable subset of m-dimensional vectors (corresponding to the columns) into an m-dimensional vector b. When $m = 1$ we obtain Knapsack. We observed that, in Knapsack, if all items are "small" when compared to the knapsack capacity then Greedy performs well; in fact if $s_i \leq \epsilon B$ then Greedy yields a $(1-\epsilon)$-approximation. What is the analogue of this in the more general setting? Suppose each column A_j is k-sparse which means that only k entries in any column are non-zero. And suppose $b_i \geq c\frac{\ln(k/\epsilon)}{\epsilon^2}$ for some sufficiently large but fixed constant c; note that we are assuming that each entry of A is in $[0,1]$. Prove that the LP relaxation can be used to obtain a $(1-\epsilon)$-approximation. You need Chernoff bounds for this problem.

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.

Note: There is a PTAS for the problem, even if the disks are allowed to have different sizes. 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.

Problem 7 Related machine scheduling. We saw in lecture the problem of scheduling n jobs with processing times $p_1, p_2, \ldots, p_n$ on 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; any list guarantees an approximation ratio of 2.. Now consider the problem where the machines are not identical but *related*. Machine 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. Describe a constant factor greedy approximation algorithm to minimize makespan in this more general setting.

Problem 8 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 and show how to round it appropriately.

Extra Credit: Consider a profit maximizing variant. Instead of a jobs we think of items and instead of machines we think of bins. Each item i has size s_{ij} in bin j. Placing i in bin j yields a profit p_{ij} and each bin j has a capacity c_j. We want to assign items to bins to maximize the profit of the assignment while not exceeding the capacity of any bin. Obtain a PTAS for this problem when m is a fixed constant.