Spring 2009, CS 598CSC: Approximation Algorithms
Homework 3

Due: 03/20/2009 in class

Instructions and Policy: You are not allowed to consult any material outside of the textbook and class notes in solving these problems. 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.

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

Problem 1 [30 pts] 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 $\mathcal{R} = \{R_1, \ldots, R_m\}$. 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, if each request is for less than $c/2$ units of capacity, 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' is constant.

Extra Credit: Give a constant-factor approximation for the weighted version *without* the assumption that each request is for less than $c/2$ units of capacity.

Problem 2 [25 pts] 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 [10 pts] Recall that in the Generalized Steiner Network Problem (also called Survivable Network Design Problem), the input is an undirected graph $G = (V, E)$ with edge costs $c : E \to R^+$, and a *requirement* r_{uv} for every (unordered) pair of vertices $u, v \in V(G)$; the goal is to find a minimum-cost set of edges E' such that for each u, v, there are r_{uv} edge-disjoint paths between u and v in E'.

In class, we saw a cut-based Linear Program for this problem with an exponential number of constraints. Give a polynomial-sized flow-based LP formulation. (Though the input graph is undirected, you will need to create an appropriate directed graph for your LP.)

Problem 4 [Exercise: Not to be turned in] Prove that each of the following classes of functions is skew supermodular:

1. Proper functions
2. Downward monotone functions
3. The residual functions of skew supermodular functions.

Can you think of an example of a residual function of a proper function that is not proper?
Hint: Consider the Steiner network problem with connectivity at most 2.

Problem 5 [35 pts] Let $G = (V, E)$ be an undirected graph and let f be an integer valued requirement function (not necessarily a $\{0, 1\}$ function) on the vertex set. Recall that the primal-dual algorithms require the ability to do answer the following questions. Given $F \subseteq E$, is F a feasible solution for f? Given $F \subseteq E$ what are the minimal violated sets with respect to F? Also recall that if f is skew-supermodular (or proper) the minimial violated sets are disjoint.

1. Suppose f is a proper function and it is accessible as an oracle which when given a set $S \subset V$ returns the value $f(S)$; such an oracle is called a value oracle. Show that there is a polynomial time algorithm to determine if F is a feasible solution.
   Hint: Consider the cuts in the Gomory-Hu tree T for the graph $G[F]$.

2. Now suppose f is a skew-supermodular function. We do not know a polynomial time algorithm to test if F is a feasible solution by simply using the value oracle for f. However, suppose you have an oracle that given $F \subseteq E$ returns whether F is feasible or not. Show how you can use such an oracle to compute in polynomial time the minimal violated sets with respect to a collection of edges A. First prove that if $S \subset V$ is a *maximal* set such that $A \cup \{(i,j) : i,j \in S\}$ is not feasible then $V \setminus S$ is a minimal violated set for A. Then deduce that the set of minimial violated sets can be obtained by less than $|V|^2$ calls to the feasibility oracle.