Problem Set 1 (due Tuesday, October 16))

Grading. This problem set has a total of 120 points. We plan to have three problem sets in all, for a total of at least 300 points. From a grading perspective, the total points allocated for problem sets is 200 points. So you should plan on attempting at least 200 points in all, over all problem sets. Any score you earn beyond 200 points will be bonus points for you.

Collaboration. You are welcome to collaborate on solving these problems. But you must write the solutions on your own. Please cite any collaborators and/or references that help you in arriving at your solutions.

1. (10 points) Stochastic matrices and LP duality

A square matrix M is called *stochastic* if all entries are nonnegative and every column sums to 1. Stochastic matrices arise frequently in diverse applications. For instance, they are used to presesent the *transition matrics* of Markov chains.

Prove that for any stochastic matrix M, there exists a probability vector π (i.e. a vector with nonnegative entries whose sum of entries is 1) such that $M\pi = \pi$. [4] (Hint: use duality and the fact that if the maximizing dual is feasible and bounded then the primal must be feasible and bounded as well.)

2. ($3 \times 5 = 15$ points) Currency trading

You are embarking on a trip to Europe and would like to purchase the maximum number of Euros possible for D US Dollars. Given the rate at which dollar is sliding with respect to many currencies, you try to perform this exchange through a collection of trades through other currencies. For instance, you may exchange some of your dollars for British pound, some of which you exchange for Japanese yen, which you convert to Euros.

You log on to your currency trading account on oanda.com with the goal of maximizing the number of Euros you can get for D US Dollars today. Oanda offers you a fixed rate r_{ij} for converting any currency i into any other currency j, but imposes a limit ℓ_{ij} on how much of currency i you can convert to currency j on a given day, for all ordered pairs (i, j). Assume that there are no arbitrage opportunities – that is, there is no directed cycle of currency trades whose product of rates exceeds 1. (Also assume that there are no other individual transaction fees.)

(a) Formulate a linear program for determining the collection of trades that will yield you the maximum amount of Euros for your D US Dollars.

(b) Show that it is possible to achieve your objective without every borrowing currency. (*Hint:* Argue that your solution can be made "acyclic".)

(c) Show that it is possible to achieve your objective such that at the end of the day you end with an optimum amount of Euros and no other currency except US Dollars.

3. (10 points) Analysis of an LP rounding algorithm for unweighted set cover

Consider the LP-based rounding "greedy" algorithm for the set cover problem that we considered in class (which we called Hooman's algorithm!), and also had a discussion via email. Recall that the LP for the set cover problem is the following, where $\mathcal{U}$ being the universe of elements, $\mathcal{C}$ being the collection of sets, and c the cost function on sets.

$$\begin{array}{lll} \min & \sum_{S \in \mathcal{C}} x_S \cdot c(S) & \\ \text{s.t.} & \sum_{S \in \mathcal{C}: e \in S} x_S \geq 1 & \forall e \in \mathcal{U} \\ & x_S \geq 0 & \forall S \in \mathcal{C} \end{array}$$

The algorithm proceeds as follows.

- Solve the LP for set cover. Let x^* denote the computed optimal solution.
- Sort the sets in nonincreasing order of their x^*_S values. Let Σ denote the resulting sequence.
- Select the smallest prefix of Σ that covers all the elements.

We saw that the above algorithm may perform poorly if the sets have non-uniform costs. Does the above algorithm yield an $O(\log n)$ approximation factor when $c(S)$ is 1 for all S? Justify your answer with a proof or a counterexample.

4. (10 points) A greedy algorithm for uncapacitated facility location

In class, we briefly discussed the following greedy algorithm for uncapacitated facility location by following the approach for set cover. Let V denote the demand points, $\mathcal{F}$ denote the potential facility locations, d_i the demand of client i, f_j the cost of opening a facility at j, and c_{ij} the distance between two points i and j.

For each $j \in \mathcal{F}$ and each $S \subseteq V$, we construct a set $X_{jS} = S$ with cost given by

$$c(X_{jS}) = f_j + \sum_{i \in S} d_i c_{ij}.$$

We now apply the greedy algorithm for the set cover problem.

Prove that the above algorithm has an approximation ratio of H_n. The algorithm, as described above, takes exponential time since we have to enumerate all the subsets of V. Show how to implement the above algorithm in polynomial time.

5. (10 points) Multiset multicover problem

The *multiset multicover problem* is a generalization of set cover in which we have multisets instead of sets and an element may be required to be covered multiple times. Formally, we are given a universe $\mathcal{U}$ of elements, a positive integer r_e for each element $e \in \mathcal{U}$, a collection $\mathcal{C}$ of multisets over $\mathcal{U}$, and a cost $c(S)$ for each multiset $S \in \mathcal{C}$. (A multiset contains a specified number of copies of each element.)

The goal of the multiset multicover problem is to determine a minimum-cost multiset of multisets (thus, you are allowed to pick multiple copies of a multiset) such that each element e is covered at least r_e times. The cost of picking a multiset S, k times, is $k \cdot c(S)$. (You may assume that the number of times that an element e appears in any multiset S is at most r_e.)

Generalize either the greedy algorithm or the randomized rounding algorithm for set cover to achieve an $O(\log n)$ approximation, where n is the number of elements in $\mathcal{U}$.

6. (15 points) Selecting optimal views for a data warehouse

Data warehouses often store precomputed *views* of data, in addition to the original data, for efficient computation of user queries. For instance, consider a business data warehouse holding data about sales of parts to customers by suppliers. Suppose all queries of interest are concerning total sales: e.g., determine the total sales for part p, or the total sales for part p by supplier s, or the total sales by supplier s to customer c. If the data warehouse stores the total sales for each triple (p, s, c), then a query of the form "total sales for part p to customer s" can be answered by adding, over all suppliers s_i, of the total sales for the triple (p, s_i, c). On the other hand, if in addition to the triples, we store the total sales corresponding to each pair (p, c), the above query can be answered more efficiently. In other words, by storing pre-computed views of the original data, future queries can be processed more efficiently. Of course, there is a tradeoff between the amount of space allocated to storing views and the efficiency of query processing.

One simple model for views is given by a lattice formed over the set A of all attributes (columns) – part, supplier, and customer in the above example – of the data. That is, we have a node S in the lattice for each subset S of the attributes and have a directed edge from S to T if $S \supset T$. A query corresponding to a view S can be answered if the view corresponding to any of its ancestors (any $T \supseteq S$) is in the database. For simplicity, we assume in this problem that the time taken to answer the query S using an ancestor T equals the number of rows in T. Note that the number of rows in a view is at least as much as the number of rows in any of its descendant views. Let $r(S)$ denote the number of rows in view S.

One problem facing a data warehouse designer is to determine a set of precomputed views to store so that the average cost of answering any of the other queries (corresponding to the views) is minimized. Clearly, we need to store the raw data which corresponds to the set A of all attributes. The goal of our problem is to determine, for a given integer k, which k additional views to store such that the decrease in total cost of computing all views is maximized.

Formulate the above problem as a variant of the set cover problem. (Note that here we have a maximization objective with a cardinality constraint while the original set cover problem is a minimization problem with a full covering constraint.) Design and analyze the best approximation algorithm that you can for this problem. (*Hint:* A greedy algorithm with an approximation ratio of $e/(e-1)$ is achievable.)

7. (5 points) Approximation factor for the LP deterministic rounding algorithm

In class, we presented a 6-approximation algorithm for the uncapacitated metric facility location problem by applying a deterministic rounding procedure to the solution of the LP relaxation (using the filtering approach). In this algorithm, we set the radius of the ball around a client to be twice the service cost of the clien tin the LP solution. Show how to use a different setting of the ball radius so as to obtain a 4-approximation.

8. (7 + 6 + 2 = 15 points) Another 4-approximation LP-based algorithm for unca-

pacitated facility location

Here is yet another approximation algorithm for uncapacitated facility location. This one is based on solving both the primal and the dual. We assume that the demand of each client is 1. Let (x^*, y^*) and (v^*, w^*) denote the optimal solutions to the primal and dual.

1. Set U to be V, the set of all clients.
2. Repeat until U is empty:
   - Let k be the client in U with least v_k^*.
   - Let j_k be the facility with least f_j among all facilities j such that $x_{kj}^* > 0$. Set $F \leftarrow F \cup j_k$
   - Let N_k denote the set of all clients i in U for which there exists j such that $x_{ij}^* > 0$ and $x_{kj}^* > 0$.
   - For each $i \in N_k$, set $\sigma(i) \leftarrow j_k$.
   - Set $U \leftarrow U - N_k$.
3. Return F and σ.

In this problem we show that the algorithm has an approximation ratio of 4.

(a) Show that the service cost for any client i is at most $3v_i^*$. (*Hint:* Use complementary slackness conditions and triangle inequality.)

(b) Show that the total facility cost is at most the facility cost of the primal LP solution, which is $\sum_j y_j^* f_j$.

(c) Conclude that the algorithm has an approximation ratio of 4.

9. (10 points) Hardness of the non-metric k-median problem

Show that the non-metric k-median problem has no polynomial-time algorithm with an approximation ratio even exponential in the number n of demand points, unless $P = NP$. (*Hint:* Reduce from set cover.)

10. (2 + 6 + 6 + 6 = 20 points) Maximum satisfiability and randomized rounding

In the maximum satisfiability problem (MAX-SAT), we are given a set $\mathcal{C}$ of m clauses over n variables and the goal is to determine a boolean assignment to the n variables that maximizes the number of satisfied clauses.

(a) An easy approximation algorithm for this problem is to set either all variables to true or all variables to false, picking whichever assignment satisfies more clauses. Show that this algorithm has an approximation factor of 2.

(b) Write the maximum satisfiability problem as an integer linear program by having an LP variable x_i for each boolean variable v_i and a linear constraint for each clause.

(c) Consider the following randomized algorithm. Solve the linear relaxation for the above integer linear program to obtain solution x^*. Set each boolean variable v_i to 1 with probability equal to x_i^*. Show that the expected number of clauses satisfied is at least $\mathrm{OPT} \cdot (1 - (1 - 1/q)^q)$, where q is the maximum number of literals in a clause and OPT is the optimal value.

(d) Derandomize the above algorithm to obtain a deterministic algorithm as follows. Proceed through the n variables v_i, for i going from 1 to n. In step i, assign v_i to true if the expected number of clauses satisfied, conditioned on setting v_i to true and the preset values for variables v_1 through v_{i-1}, is at least the expected number of clauses satisfied, conditioned on setting v_i to false and the preset values for variables v_1 through v_{i-1},

Show that the derandomized algorithm can be implemented in polynomial time and satisfies at least $\mathrm{OPT} \cdot (1 - (1 - 1/q)^q)$ clauses.