

Solution Sketch for Problem Set 1

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 present the *transition matrices* 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.)

Answer: We write the following LP.

$$\begin{aligned} \min \quad & \sum_j \pi_j \\ \sum_j m_{ij} \pi_j - \pi_i \quad & = \quad 0 \\ \sum_j \pi_j \quad & \geq \quad 1 \\ \pi_j \quad & \geq \quad 0 \end{aligned}$$

The dual for this LP is the following.

$$\begin{aligned} \max \quad & \lambda \\ \sum_i m_{ij} v_i - v_j + \lambda \quad & \leq \quad 1 \\ \lambda \quad & \geq \quad 0 \end{aligned}$$

We construct a feasible solution to the dual as follows. Set $v_i = 1$ for all i . For this solution, we get the first inequality of the dual to be $\lambda \leq 1$ since the sum of the entries in each column of M is 1. We set $\lambda = 1$ to obtain a feasible dual solution. It can also be shown that the dual is bounded; its value can be no more than 1 since the dual inequality corresponding to the smallest v_j will yield $\lambda \leq 1$ (using the fact that the sum of the entries in each column is 1). By weak duality, we have a primal optimal solution with $\sum_j \pi_j \leq 1$. But this implies that in the optimal solution $\sum_j \pi_j = 1$. And we obtain that $M\pi = \pi$.

2. (3 × 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.

Answer: Let x_{ij} denote the amount of currency i converted to currency j . Let e denote euro and d denote dollar. Then, the LP can be written as:

$$\begin{aligned} \max \quad & \sum_j x_{je} r_{je} \\ \sum_j x_{ij} \quad & \leq \sum_k x_{ki} r_{ki} \quad i \neq d \\ \sum_j x_{dj} \quad & \leq D \\ x_{ij} \quad & \geq 0 \\ x_{ij} \quad & \leq \ell_{ij} \end{aligned}$$

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

Answer: The above LP models a flow from the dollar node to euro node where the amount of fluid changes (according to the exchange rate) when it travels along any change. Suppose we had a cycle in this flow, say $i_0 \rightarrow i_1 \rightarrow \dots \rightarrow i_{m-1} \rightarrow i_0$. Then, consider reducing the flow along edge $i_0 \rightarrow i_1$ by a small value ε and correspondingly reducing the flows along the other edges of the cycle (using the appropriate exchange rate). This can be done by making ε sufficiently small. We choose the largest ε that eliminates the flow along one of the edges of the cycle.

By repeatedly doing the above, we eliminate all cycles. This means we can set the trades up so that we convert a particular currency i into other currencies only after executing all other conversions into i (in topological sorted order), hence without ever borrowing i .

- (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.

Answer: Again, using the notion of the flow, we can see that if there is any currency i with surplus left over, we can find a path from dollar node d to i and reduce the flow along this path, without changing the amount of euros we have.

(**Acknowledgement:** Problem due to Piotr Indyk and David Karger at MIT.)

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.

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

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.

Answer: The first part is easy to see since we have a 1-1 correspondence between a set cover with cost C to a facility location solution with cost C .

In order to implement in polynomial time, we make the following observation. Consider any round of the algorithm, and let V' denote the set of remaining demand points. Then, the best X_{jS} for a location j will be one of the following $|V'|$ sets: X_{jS_i} , $1 \leq i \leq |V'|$ where S_i is the set of i clients v with the smallest $d_v c_{vj}$ values. So we only need to consider a polynomial number of sets in any round, hence making the algorithm polynomial time.

5. (10 points) Multiset multcover problem

The *multiset multcover 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} .

Answer: If we use a greedy algorithm, then it appears that the best one can show is an $O(\log m)$ approximation where m is the sum of the requirements, over all elements. Using randomized rounding, one can obtain an $O(\log n)$ approximation as follows.

We set up the LP similar to that for set cover. Let x_S denote the variable for multiset S . We first select $\lfloor x_S \rfloor$ copies of multiset S . When we do that for every multiset S , we have satisfied some of the requirements, so we can reduce those requirements. For the remaining problem, $y_S = x_S - \lfloor x_S \rfloor$ is a valid fractional solution.

We will now do the rounding for the remaining problem. We add another copy of S with probability y_S . The expected total cost is the same as the LP optimal. The expected number of times that an element e is covered is at least r_e . But this is not guaranteed.

If we repeat the above process $O(\log n)$ times one can argue using Chernoff bounds that the probability that element e is not completely covered is at most $1/n^c$ for constant $c > 0$ that can be made large by adjusting the hidden constant in the big-Oh term. Thus, the probability that all elements are covered up to their requirement is at least $1 - 1/n$, by choosing c appropriately.

The expected cost of the solution is $O(\log n)$ times optimal LP value. The probability that the cost exceeds $O(\log n)$ times optimal is at most $1/4$, using Markov's inequality (and selecting the hidden constant appropriately).

Now, we have a situation similar to what we had for set cover. We just repeat the above process until we get a solution whose cost is at most $O(\log n)$ times optimal LP value and every element is covered.

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.)

Answer: We formulate the problem as follows. We consider each view T as a set of all views that can be answered by T (precisely the subsets of T). The cost of computing a view S from T is the number of rows of T . The goal is to select k views such that the decrease in total cost of computing all views is maximized.

We consider the following greedy algorithm. In each of k rounds, select the view that contributes the greatest benefit in cost of computing the views (given previous selections).

Let the views selected by the greedy algorithm be S_1, \dots, S_k and those selected by the optimal be T_1, \dots, T_k . Let the benefits achieved be a_1, \dots, a_k , and b_1, \dots, b_k , respectively. Clearly, we have $a_1 \geq b_i$ for all i . Owing to the selection of S_1 , the benefit of any view T_i from the optimal solution may decrease at the time of the second selection; let this decrease be d_{i1} . Then, we have by the definition of the greedy algorithm:

$$a_2 \geq b_i - d_{i1}.$$

Similarly using the j th selection, we have

$$a_j \geq b_i - \sum_{\ell=1}^{j-1} d_{i\ell}.$$

We also have the condition

$$\sum_i d_{ij} \leq a_j.$$

The above inequalities yield the following set of inequalities:

$$\begin{aligned} OPT &\leq ka_1 \\ OPT &\leq ka_2 + a_1 \\ &\dots \\ OPT &\leq ka_k + a_1 + \dots + a_{k-1}. \end{aligned}$$

One can argue that OPT is maximized when we set all the inequalities to be equalities, for which we set $a_i = ka_{i+1}/(k-1)$. After some algebra, this yields us

$$\begin{aligned} \sum_i a_i &\geq OPT \cdot \left[1 - \left(\frac{k-1}{k} \right)^k \right] \\ &= OPT \cdot \left(1 - \frac{1}{e} \right) \end{aligned}$$

(**Acknowledgment:** This result is due to Harinarayan et al, SIGMOD 1996.)

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 client in the LP solution. Show how to use a different setting of the ball radius so as to obtain a 4-approximation.

Answer: When we applied the filtering technique, we filtered out all contributions for a client that were twice the average cost of the client, according to the LP solution. This eventually contributed an increase by a factor of 2 to the facility opening cost and the service cost. The service cost was further increased by a factor of 3 when we processed the client balls in increasing order of their radii and applied an argument based on triangle inequality. (See lecture notes for details.)

In the above argument, we are paying more with respect to client service cost (6 times) than with respect to facility cost (2 times). We can try to balance these. Suppose we filter x_{ij} away whenever facility j is at least α times the average service cost for client i . Then, we will obtain an approximation ratio of

$$\max\left\{\frac{\alpha}{\alpha-1}, 3\alpha\right\},$$

since each ball would contain $\alpha/(\alpha-1)$ fraction of a facility, by averaging, and the 3α term arises in the ball overlapping argument.

This is minimized if we set $3(\alpha-1) = 1$, yielding $\alpha = 4/3$, yielding a 4-approximation.

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

Answer: By complementary slackness, $x_{ij}^* > 0$ implies that $v_i^* = w_{ij}^* + c_{ij}$. So $v_i^* \geq c_{ij}$. Let i be assigned to j_k in the above algorithm. So the service cost for i is given by:

$$c_{ij_k} \leq c_{ij} + c_{kj} + c_{kj_k} \leq v_i^* + 2v_k^* \leq 3v_i^*.$$

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

Answer: Since j_k is the facility with least opening cost that is serving k , we obtain that

$$f_{j_k} \leq \sum_{j \in \mathcal{F}} x_{kj}^* f_j \leq \sum_{j \in \mathcal{F}: x_{kj}^* > 0} y_j^* f_j.$$

The set of facilities being added in the right-hand side are disjoint for different j_k since if $x_{kj}^* > 0$, then $x_{k'j}^* = 0$, for two different choices of clients (k and k') in the above algorithm.

Thus, the sum of the facility opening costs is at most the LP cost.

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

Answer: Follows from duality, and (a) and (b) since the service and facility costs are within 3 times and 1 time the LP optimal cost.

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.)

Answer: Consider a set cover instance with collection \mathcal{C} of sets and universe \mathcal{U} of elements. Form a k -median instance with client locations being the elements and the potential facility locations being the sets. Set the distance between a set S and element e in S to be 1 and all other distances to be M , which is a large number that is specified below.

We argue that if there exists a set cover with k sets then there exists a k -median solution with cost n . On the other hand, if there does not exist a set cover with k sets, then the best k -median solution has cost at least M . The first part is easy to see since a k -median solution with facilities opened at the selected sets yields a solution with cost 1. For the second part, we note that if k sets cannot cover all elements, then at least one client will be M distance away from its facility.

Now, we set M to be exponential in n . We can do this and maintain the poly-time computability of reduction. The above arguments establish that we cannot obtain a poly-time algorithm for non-metric k -median with approximation ratio of M/n , unless $P = NP$.

In the above argument, we could have set the distances to be 0 and 1, instead of 1 and M , respectively, and obtained an arbitrarily inapproximability bound.

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.

Answer: Let T be the all true-assignment and F be the all-false assignment. Clearly, any clause that is false under one assignment is true under the other. So there are at least half the clauses that are true under one of these two assignments, yielding the desired approximation factor.

- (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.

Answer: We let x_i be 1 if v_i is true and 0 otherwise. For clause C_j , let y_j be 1 if the clause is true and 0, otherwise. So we add a constraint

$$\sum_{v_i \in C_j} x_i + \sum_{\bar{v}_i \in C_j} (1 - x_i) \geq y_j.$$

for each clause and the constraints

$$x_i \in \{0, 1\}, y_j \in 0, 1$$

for each clause and variable.

The objective function is given by

$$\sum_j y_j.$$

- (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 $\text{OPT} \cdot (1 - (1 - 1/q)^q)$, where q is the maximum number of literals in a clause and OPT is the optimal value.

Answer: The probability that a clause C_j is not satisfied in randomized rounding is

$$\prod_{v_i \in C_j} (1 - x_i^*) \cdot \prod_{\bar{v}_i \in C_j} x_i^*.$$

By the linear constraint for the clause, we know that the sum of the terms being multiplied above is at most $q_j - y_j$, where q_j is the number of literals in C_j . The maximum value this product can take is

$$\left(\frac{q_j - y_j}{q_j} \right)^{q_j} = \left(1 - \frac{y_j}{q_j} \right)^{q_j}.$$

So the probability that clause C_j is satisfied is at least

$$1 - \left(1 - \frac{y_j}{q_j}\right)^{q_j}.$$

It can be shown that the above is at least $y_j(1 - (1 - 1/q_j)^{q_j})$, for $y_j \in [0, 1]$. We can see this by analyzing the difference between the two terms. Adding over all j and replacing q_j by the upper bound q yields the desired bound.

- (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 $\text{OPT} \cdot (1 - (1 - 1/q)^q)$ clauses.

Answer: Let Y denote the random variable corresponding to the total number of clauses satisfied. We have:

$$E[Y] = \Pr[v_1 = 0]E[Y|v_1 = 0] + \Pr[v_1 = 1]E[Y|v_1 = 1].$$

Thus, one of the two conditional expectations is at least $E[Y]$. So, we set v_1 to the appropriate value and effectively obtain a new MAX-SAT problem in which we can remove all the satisfied clauses and all terms corresponding to v_1 .

Repeating this in every iteration yields us a deterministic solution that satisfies the expected number of clauses satisfied by the randomized rounding algorithm, which is at least $\text{OPT} \cdot (1 - (1 - 1/q)^q)$.

In each iteration, the conditional expectations can be calculated in polynomial time.

(This result is due to Goemans-Williamson.)