Linear Programming
Problem Set 4 – CS 6515/4540 (Fall 2025)

This problem set is due on **Thursday October 16th**. Submission is via Gradescope. Your solution must be a typed pdf (e.g. via LaTeX) – no handwritten solutions.

13 Approx Algo

1. For the maximum matching problem in general unweighted graphs, prove that the greedy algorithm (order edges arbitrarily and add the next edge to matching if both vertices are free) returns a maximal[1] matching with size at least 1/2 times the maximum matching. Also, give an example where maximal matching size is half of the maximum matching (in other words, the approx factor 1/2 is tight).

   Suppose the maximum matching M_{max} with a edges was more than double the size of our maximal matching M_{alg} with b edges, so $a > 2b$. If all edges in M_{max} had at least one vertex from M_{alg}, then because a vertex from M_{alg} can't appear in two different edges of M_{max}, we have M_{alg} has at most a different vertices, implying $2b \leq a$ which is a contradiction. Therefore some edge in M_{max} has no vertices from M_{alg} and so M_{alg} is not maximal since we can just add that edge to M_{alg}, and therefore our algorithm must be a 1/2-approximation.

   For an example where the approx 1/2 factor is tight, just consider the path graph on three vertices. The maximal matching using the middle edge is half the size of the maximum matching which uses the two outer edges.

2. Consider placing n items into m bins. Item i has weight w_i. We want to place these n items into the bins such that the maximum total weight of a bin is minimized. Consider the following greedy algorithm:

   Sort the weights w_i in decreasing order. Going through the items in decreasing order of weight, we place each item in the bin with the least current load (as we iterate through the items and place more items into bins, the least loaded bin may change).

   Show this algorithm gives a 1.5-approximation for the least possible maximum load of any placement of these n items into the m bins.

[1] A matching M is called maximal if there is no edge $e \in E$ such that $M \cup e$ is also a matching.

Sketch: If the maximum bin B had 3 or more items, then the last item i added to B must satisfy $w_i \leq \frac{1}{2}(B - w_i) \to B \leq \frac{3}{2}(B - w_i)$ since we are looping through the items in decreasing order (there are 2 other items in B bigger than w_i). Hence we have a 1.5-approximation in this case, because $OPT \geq B - w_i$, which is because adding an item to a bin via greedy means that bin had the smallest load at that iteration, and the smallest at a specific point in time means it was smaller than the average, and the maximum of any configuration can never be smaller than the average load.

For the case where the maximum bin B has 2 items w_i and w_j where $w_i \geq w_j$, we can see the optimal max would be at least twice the weight of the second item w_j added to B, since being forced to reuse a loaded bin means we placed at least m items (of weight $\geq w_j$) so far, thus by pigeonhole some bin of any configuration must have ≥ 2 items of weight $\geq w_j$. So $OPT \geq 2w_j$. But the optimal max would also be $\geq$ weight of the first item added to B, so $OPT \geq w_i$. Hence $B = w_i + w_j \leq 1.5OPT$.

and the case where B has 1 item is trivial, as it must be optimal.

14 LP-Based Set Cover

Prove that for the set cover problem, there is an f-approximation algorithm where f is the maximum number of sets in which an element could appear.

(Hint: First write an LP and then round it, generalizing the 2-approx LP based algo for vertex cover.)

Consider the LP relaxation for Set Cover

$$\min \sum_{i=1}^{m} x_i$$

$$\sum_{e \in S_i} x_i \geq 1 \text{ for all } e \in [n]$$

$$x_i \geq 0 \text{ for all } i \in [m]$$

Consider the minimal solution $\vec{x}$. Since every element e appears in at most f sets, by pigeonhole principle on each of the membership inequalities, we have that there exists some i where $e \in S_i$ and $S_i \geq \frac{1}{f}$. Thus, if we round all variables $\geq 1/f$ to 1 and the rest to 0, the membership inequalities will still be satisfied. Consider a variable $x_i \geq 1/f$ that gets rounded to $x_i' = 1$. Then $x_i'/x_i \leq f$. The rest of the variables decrease in value because we round to 0. Thus the sum of the x_i' is at most f times the sum of the x_i and we are done.

15 Facility Location

For the facility location problem discussed in class, in this problem we want to improve the 6-approx algorithm from class to a 4-approximation algorithm. In particular, analyze the same algorithm from class except that now each ball B_j has radius $4/3 \cdot L_j$ (instead of $2L_j$) where $L_j := \sum_i d_{ij} y_{ij}^*$.

(You don't need to prove that the LP relaxation from class gives a lower bound on the optimum cost. But repeat any steps of the LP rounding analysis as needed.)

Note that $\sum_{i \in B_j} x_i^* \geq 1/4$ now, since if $3/4$ fraction of facilities were outside $4/3L_j$, our average length would be

$$L_j = \sum_i d_{ij}^* y_{ij}^* = \sum_{i \in B_j} d_{ij}^* y_{ij}^* + \sum_{i \notin B_j} d_{ij}^* y_{ij}^*$$

$$\geq \sum_{i \notin B_j} d_{ij}^* y_{ij}^* > \sum_{i \notin B_j} 4/3L_j y_{ij}^* = 4/3L_j \sum_{i \notin B_j} y_{ij}^* > 4/3L_j \cdot \frac{3}{4} = L_j$$

which is a contradiction, so we have a 4-approx for facility opening costs. For facility assignment costs, we repeat the argument from lecture and get that our distance from client j to its assigned facility is at most 3 times the radius of the ball B_j, so we have $3 \cdot 4/3 = 4$ and hence we have a 4-approx for facility assignment as well, so total cost is also a 4-approx.

16 Probability Theory

1. Give an example of a correlated distribution over two non-negative random variables, X and Y, such that $\mathbb{E}[XY] \neq \mathbb{E}[X] \cdot \mathbb{E}[Y]$.

   Let X be the result of a 6-sided die roll and Y track the same die, so these two random variables are obviously correlated. And indeed $\mathbb{E}[XY] \neq \mathbb{E}[X] \cdot \mathbb{E}[Y]$ as $\mathbb{E}[XY] = \mathbb{E}[X^2] = (1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2)/6 \approx 15.167$ whereas $\mathbb{E}[X] \cdot \mathbb{E}[Y] = 3.5^2 = 12.25$.

2. Suppose there are n distinct coupons $\{1, \ldots, n\}$. In each step, you draw a uniformly random coupon independently with replacement (i.e., you may redraw coupons), and you repeat this until you have drawn *all n coupons* at least once.

   (a) Prove that the expected number of total draws is $O(n \log n)$.

   Hint: Use linearity of expectation.

Approach using linearity of expectation: Let X_k be a random variable denoting the number of coin tosses before we draw the k-th distinct coupon given that we have already drawn $k-1$ distinct coupons. The total number of coin tosses is

$$\sum_{k=1}^{n} X_k$$

Thus, the expected number of coin tosses by linearity of expectation is

$$\sum_{k=1}^{n} \mathbb{E}[X_k]$$

To calculate $\mathbb{E}[X_k]$, note that the chance of the next coin toss generating the k-th distinct coupon is $\frac{n-(k-1)}{n}$, which means $\mathbb{E}[X_k]$ is its reciprocal $\frac{n}{n-(k-1)}$. This means that the expected total number of coin tosses is

$$\sum_{k=1}^{n} \mathbb{E}[X_k] = \sum_{k=1}^{n} \frac{n}{n-(k-1)} = n \cdot \sum_{k=1}^{n} \frac{1}{n-(k-1)} = n \cdot \sum_{k=1}^{n} \frac{1}{k} = \mathcal{O}(n \log n)$$

Approach using recurrence: Let E_k denote the expected number of steps to draw all coupons if there are k coupons you have not yet drawn. The base case is $E_0 = 0$. E_k can be computed via the recurrence

$$E_k = 1 + \frac{k}{n} E_{k-1} + \frac{n-k}{n} E_k$$

as there is a $\frac{k}{n}$ chance of drawing a new coupon and bringing us to the situation E_{k-1}, and a $\frac{n-k}{n}$ chance of drawing a coupon we already own and thus staying in the same situation E_k. Rearrange to isolate E_k, so we get

$$E_k = \frac{n}{k} + E_{k-1}$$

So

$$E_n = \sum_{k=1}^{n} \frac{n}{k} = n \sum_{k=1}^{n} \frac{1}{k} = \mathcal{O}(n \log n)$$

(b) Prove that with probability at least $1 - 1/n$, the process takes $O(n \log n)$ steps.

Hint: First calculate the probability that a particular coupon is never drawn after $O(n \log n)$ steps. Next, apply the union bound, which states that for any events A, B, we have $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$.

When we say the process takes $\mathcal{O}(n \log n)$ steps, we are saying that it takes $cn \log n$ steps for some constant c. The probability of a particular coupon not being drawn after $cn \log n$ is

$$\left(1 - \frac{1}{n}\right)^{cn \log n}$$

as each step has an independent $1 - 1/n$ probability of not drawing the desired coupon. We now use the standard inequality $\left(1 - \frac{1}{n}\right)^n \leq e^{-1}$, so

$$\left(1 - \frac{1}{n}\right)^{cn \log n} \leq e^{-c \log n} = n^{-c} = \frac{1}{n^c}$$

Taking a union bound over all n coupons upper bounds the total probability of failure as

$$n \cdot \frac{1}{n^c} = \frac{1}{n^{c-1}}$$

So the probability of success is at least

$$1 - \frac{1}{n^{c-1}}$$

Picking $c = 2$ yields $1 - 1/n$.

17 Max 2-SAT

In the Max 2-SAT problem, there are n boolean variables $\{x_1, \ldots, x_n\}$. We define a literal to mean a (boolean) variable or its negation. We are given m clauses $J_1 \cup J_2$ of two types depending on the number of literals in them: each clause in J_1 consists of a single literal y and a clause in J_2 is the OR of two literals, say $y \vee z$ for some literals y, z. The goal is to find an assignment of the variables to maximize the number of satisfied clauses. (A clause in J_1 is satisfied if its literal is true and a clause in J_2 is satisfied if either of its literals are true.)

Since the maximization problem is NP-Hard, we will design an LP based approximation algorithm. Consider the following LP relaxation. We have a variable z_j for the jth clause in $J_1 \cup J_2$, where the intended meaning is that it is 1 if the assignment decides to satisfy that clause and 0 otherwise. (Of course the LP can choose to give z_j a fractional value.)

$$\begin{aligned}
\max \quad & \sum_{j \in J} z_j \\
& 1 \geq x_i \geq 0 \qquad \forall i \\
& z_j \leq 1 \qquad \forall j \in J_1 \cup J_2 \\
& y_{j1} \geq z_j \qquad \forall j \in J_1 \\
& y_{j1} + y_{j2} \geq z_j \qquad \forall j \in J_2
\end{aligned}$$

Here y_{j1} is shorthand for x_i if the first literal in the jth clause is the ith variable, and shorthand for $1 - x_i$ if the literal is the negation of the i variable. (Similarly for y_{j2}.)

1. Prove that the optimal LP value is at least the maximum number of clauses that can be simultaneously satisfied by any assignment of the boolean variables.

Set each of the x_i to 0 or 1 from the optimal max 2-SAT solution. Set $z_j = 1$ if the jth clause is satisfied in the optimal max 2-SAT solution and 0 otherwise. Then for each inequality, it is always satisfied if $z_j = 0$ (the clause does not need to be satisfied) and will be satisfied if $z_j = 1$ because we set at least one of its literals to be true. So the optimal max 2-SAT solution is a feasible solution to the LP, so $\mathrm{OPT}_{\text{Max 2-SAT}} \leq \mathrm{OPT}_{\text{LP}}$.

2. Suppose we set each boolean variable to be true independently with probability x_i and false otherwise, prove that the expected number of satisfied clauses is at least $\frac{3}{4}\sum_j z_j$ (i.e., at least 3/4 times the optimal LP value), and hence we obtain a 3/4 approximation in expectation.

   Hint: Use linearity of expectation.

If we can show the jth clause is satisfied with probability at least $\frac{3}{4}z_j$ for every j, then we can apply linearity of expectation to get a total expectation of $\frac{3}{4}\sum_j z_j$. If the jth clause is in J_1, then it is satisfied with probability

$$y_{j1} \geq z_j \geq \frac{3}{4}z_j$$

If the clause is in J_2, then it is satisfied with probability

$$y_{j1} + y_{j2} - y_{j1}y_{j2}$$

From now on I will write xy_{j1} and yy_{j2} for brevity. So we want to show

$$x + y - xy \geq \frac{3}{4}z_j$$

Since the LP attempts to maximize $\sum z_j$, $z_j = \min\{x+y, 1\}$. Now consider two cases.

Case 1: $z_j = x + y$ We want to show

$$x + y - xy \geq \frac{3}{4}(x+y) \impliedby \frac{1}{4}(x+y) \geq xy$$

Define $sx + y$, then we want to show

$$\frac{1}{4}s \geq x(s-x) \impliedby x^2 - sx + \frac{1}{4}s \geq 0$$

This is a quadratic with respect to x and is minimized when $x = \frac{s}{2}$. Plugging that back in yields

$$\frac{1}{4}s \geq \frac{s}{2}\left(s - \frac{s}{2}\right) \impliedby 1 \geq s$$

which is true because we are in the case where $z_j = s$ instead of $z_j = 1$.

Case 2: $z_j = 1$ We want to show

$$x + y - xy \geq \frac{3}{4}$$

Again substitute $sx + y$, so we want to show

$$s - x(s-x) \geq \frac{3}{4} \impliedby x^2 - sx + s \geq \frac{3}{4}$$

Again we have a quadratic on the LHS with respect to x which is minimized when $x = \frac{s}{2}$. Plugging that back in yields

$$s - \frac{s}{2}\left(s - \frac{s}{2}\right) \geq \frac{3}{4} \impliedby s - \frac{s^2}{4} \geq \frac{3}{4}$$

This is a quadratic with respect to s that is concave, so it is minimized on the boundaries of the domain. In case 2, $1 \leq s \leq 2$ so we try both $s = 1$ and $s = 2$. $s = 1$ achieves the minimum of $\frac{3}{4}$ on the LHS so the inequality holds true.