CSCI 670: Advanced Analysis of Algorithms
5^{th} Assignment

Fall 2021

Problem 1. Let M be an $n \times n$ Markov matrix, i.e.,

- $M_{i,j} \geq 0$ for every $1 \leq i, j \leq n$; and
- $\sum_{1 \leq j \leq n} M_{i,j} = 1$ for every $1 \leq i \leq n$.

Assume M has a unique stationary distribution π, so by definition, $\pi M = \pi$. Define Π as the diagonal $n \times n$ matrix corresponding to π, meaning that $\Pi_{i,i} = \pi_i$ for every $1 \leq i \leq n$ (and all other entries of Π are zero). Let $X = \Pi M$.

(a) What is $\sum_{1 \leq j \leq n} X_{i,j}$ for a fix i (i.e., sum of the elements in the i^{th} row of X)?

(b) What is $\sum_{1 \leq i \leq n} X_{i,j}$ for a fix j (i.e., sum of the elements in the j^{th} column of X)?

Solution 1. Notice that $\pi M = \pi$ implies that $\sum_{i=1}^{n} \pi_i M_{i,j} = \pi_j$ for all $1 \leq j \leq n$.

$$X_{i,j} = \sum_{k=1}^{n} \Pi_{i.k} \cdot M_{k,j} = \Pi_{i,i} \cdot M_{i,j} = \pi_i \cdot M_j \qquad (\text{since } \Pi_{i,j} = 0 \text{ when } i \neq j)$$

1. $\sum_{j=1}^{n} X_{i,j} = \sum_{j=1}^{n} \pi_i M_{i,j} = \pi_i \cdot \sum_{j=1}^{n} M_{i,j} = \pi_i$.
2. $\sum_{i=1}^{n} X_{i,j} = \sum_{i=1}^{n} \pi_i M_{i,j} = \pi_j$ since $\pi M = \pi$.

Problem 2. Suppose a gambler starts with n dollars and makes a list of bets. Gambler wins 1 dollar with probability p and loses 1 dollar with probability $1 - p$. A gambler plays until loses all his dollars or wins all m dollars of the casino and reaches $N = n + m$. What is the probability that the gambler wins all dollars before he ruins?

Solution 2. Let's start with the uniform case where $p = \frac{1}{2}$. First of all, we will define notations. Let X_i be the gambler's budget, and T be the time play ends. In other words, T is the first time when the gambler has 0 dolar or $n + m$ dolars. We can define P_i as the winning probability of the gambler given that the budget is i dolar, formally

$$P_i = \Pr[X_T = n + m \mid X_0 = i].$$

Since this is a markov process, we can write down a recurrence relation based for P_i's.

$$P_i = \frac{1}{2} \cdot P_{i-1} + \frac{1}{2} \cdot P_{i+1} \iff P_{i+1} - P_i = P_i - P_{i-1}$$

for $1 \leq i \leq n + m - 1$. In addition to these relations, we know that $P_0 = 0$ and $P_N = 1$. Combining everyting gives us

$$P_i = P_N - \frac{1}{N} \cdot (N - i) = \frac{1}{N} \cdot i$$

Thus, $P_n = \frac{n}{N} = \frac{n}{n+m}$.

For the non-uniform case, we can write the recurrence relation as follows.

$$P_i = (1-p)P_{i-1} + p \cdot P_{i+1} \iff P_{i+1} - P_i = \frac{1-p}{p} \cdot (P_i - P_{i-1})$$

for $1 \leq i \leq n_m - 1$. Solve the relation by using $P_0 = 0$, and $P_N = 1$. Then, obtain

$$P_i = \frac{1 - (\frac{1-p}{p})^i}{1 - (\frac{1-p}{p})^N}.$$

Problem 3. Design a way to implement any 3-SAT formula $\mathcal{F}$ as an integer linear program. So, we can deduce that solving integer linear programs is NP-hard.

Solution 3. Let $\mathcal{F}$ be an instance of 3SAT and $x_1, \ldots, x_n$ be the Boolean variables in $\mathcal{F}$. To turn in to an instance of INTEGER PROGRAMMING problem, corresponding to each x_i, define an integer variable y_i with the following constraint:

$$0 \leq y_1 \leq 1$$

Let $C = l_i \vee l_j \vee l_k$ be a clause where literal l_i is either x_i or $\overline{x_i}$, literal l_j is either x_j or $\overline{x_j}$, and literal l_k is either x_k or $\overline{x_k}$. Corresponding to clause C, we add a constraint to the the integer program as follows. If l_i is a positive literal (i.e., $l_i = x_i$), let $t_i = y_i$, otherwise (i.e., $l_i = \overline{x_i}$), let $t_i = 1 - y_i$. In a similar way, define t_j and t_k. The constraint that we add to the program (corresponding to C) will be

$$t_i + t_j + t_k \geq 1$$

It is easy to see that a Boolean assignment to $x_1, \ldots, x_n$ satisfies $\mathcal{F}$ if and only if the corresponding integer assignment to $y_1, \ldots, y_n$ satisfies all of the integer program's constraints. Hence, $\mathcal{F}$ is satisfiable if and only if there is a feasible assignment to variables $y_1, \ldots, y_n$.

Problem 4. It can be shown that any linear program can be reduced to an equivalent linear program that has the standard primal from:

$$\begin{array}{ll} \max_x & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array}$$

The dual problem of this problem is:

$$\begin{array}{ll} \min_y & b^T y \\ \text{s.t.} & A^T y \geq c \\ & y \geq 0 \end{array}$$

Prove that for any feasible point y of the dual problem and feasible point x of the primal problem, we have that $b^T y > c^T x$ (weak duality).

Solution 4. Let x and y be arbitrary feasible points of the corresponding programs. We have that

$$b^T y \geq \left(x^T A^T\right) y = x^T (A^T y) \geq x^T c = c^T x$$

Problem 5. Let G be the ground set and $\mathcal{F} = \{S_1, \ldots, S_m\}$ be a family of its subsets (i.e., $S_i \subseteq G$ for all i) with $\bigcup_i S_i = G$. Notice that every S_i is a subset of G (not a member). A "set cover" T is a subset of $\mathcal{F}$ which covers entire ground set. More formally, we say $T \subseteq \mathcal{F}$ is a cover for G if $\bigcup_{S_i \in T} S_i = G$. Notice that we assumed $\mathcal{F}$ itself is a set cover. Size of a set cover T is simply $|T|$. MINIMUM SET COVER problem is the problem of finding a minimum size set cover.

(a) Write this optimization problem as an integer program (i.e., reduce MINIMUM SET COVER problem to INTEGER PROGRAMMING problem).

(b) (Optional) Relax the integer program you wrote in part (a) to a linear program and write down its dual. Interpret the dual problem as a combinatorial problem.

Solution 5. (a) For every set S_i, define an integer variable x_i with the following constraint:

$$0 \leq x_i \leq 1$$

Intuitively speaking, $x_i = 1$ corresponds to picking S_i in the cover and $x_i = 0$ is corresponding to not picking it. The covering constraint is that

$$\sum_{i:S_i \ni x} x_i \geq 1 \quad \text{for all } x \in G$$

(i.e., for every x, there should be at least a set in the cover that covers x). Subject to these constraints, we would like minimize $\sum x_i$ (which is a linear objective function).

(b) By writing it in the matrix form, this is easy to see that in the dual, for every element $x_j \in G$ there is a variable y_j and corresponding to each set S_i, there is a constraint in the following form:

$$\sum_{x_j \in S_i} y_j \leq 1$$

The objective function is to minimize $\sum y_j$. To translate this mathematical problem into a combinatorial problem, you can think of each y_j as the amount of money you charge element x_j. The objective function is to get as much as possible in total. The constraints, however, are saying that for every set S_i, total charge to the elements of S_i must be no more than 1.

[Note: For instance, assume that you are charging the elements to provide some service for them and you know for every set S_i, there exists another provider willing to give elements of S_i the service if they pay 1 dollar combined. This means if you charge elements of any set more than 1 bucks, they leave the game and get the service from a different provider.]

Problem 6. (a) Consider the following optimization problem:

$$\begin{array}{lll} \min\limits_{x} & \max\{f_1(x), f_2(x), \ldots, f_k(x)\} & \\ \text{s.t.} & g_i(x) \geq 0, & 1 \leq i \leq r \\ & h_j(x) = 0, & 1 \leq j \leq s \end{array} \tag{1}$$

where each $f_i(x)$ is a linear function, and $g_i(x)$ and $h_j(x)$ are affine functions. Show that this problem can be written as a linear program.

Solution 6. Note that $\max\{f_1(x), \ldots, f_k(x)\}$ is the smallest number ξ such that $\xi \geq f_i(x), \forall i$. With this observation, we write down the following linear program:

$$\begin{array}{lll} \min\limits_{x,\xi} & \xi & \\ \text{s.t.} & g_i(x) \geq 0, & 1 \leq i \leq r \\ & h_j(x) = 0, & 1 \leq j \leq s \\ & \xi \geq f_i(x), & 1 \leq i \leq k \end{array} \tag{2}$$

To show that (1) is equivalent to (2) we need to show that from an optimal solution of one we can construct an optimal solution of the other.

1. Consider an optimal solution x^* of (1). We claim that (ξ^*, x^*), where $\xi^* = \max\{f_1(x^*), \ldots, f_k(x^*)\}$ is an optimal solution for (2). Assume it is not, and the optimal solution is $\left(\xi^\#, x^\#\right)$. It is easy to see that from optimality it follows that $\xi^\# = \max\left\{f_1\left(x^\#\right), \ldots, f_k\left(x^\#\right)\right\}$. But $\xi^\# < \xi^*$ implies that $x^\#$ is a strictly better solution for (1) that x^*, which is a contradiction.

2. Similarly, if (ξ^*, x^*) is an optimal solution of (2) then x^* is an optimal solution of (1). Assume it is not, and $x^\#$ is the optimal solution. Then $\left(\xi^\#, x^\#\right)$ is a feasible point of (2) that has better value that the optimal solution, which is a contradiction.

Problem 7. Consider a regression problem with n data points $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Linear regression seeks to find $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$, such that the hyperplane $w^T x + b$ approximates the data points well:

$$\min_{w,b} \sum_{i=1}^n \left(w^T x_i + b - y_i\right)^2$$

One can measure the approximation error on the i-th sample with $\left|w^T x_i + b - y_i\right|$. With this definition we get the following problem:

$$\min_{w,b} \sum_{i=1}^n \left|w^T x_i + b - y_i\right| \tag{3}$$

Show that this problem can be solved in polynomial time using linear programming.

Solution 7. The main observation is that $|a|$ is the smallest number ξ such that $\xi \geq a$ and $\xi \geq -a$. Therefore, we introduce one variable ξ_i for each error term in (3) and formulate the following problem:

$$\begin{array}{lll} \min\limits_{w,b,\xi} & \xi_1 + \xi_2 + \cdots + \xi_n & \\ \text{s.t.} & \xi \geq w^T x_i + b - y_i, & i = 1, \ldots, n \\ & \xi \geq -w^T x_i - b + y_i, & i = 1, \ldots, n \end{array} \tag{4}$$

Again, to show equivalence between (3) and (4) we show how to construct an optimal solution of one given an optimal solution of another.

1. Assume (w^*, b^*, ξ^*) is an optimal solution of (4). Then $\xi_i^* = \left|\left(w^*\right)^T x_i + b^* - y_i\right|$, otherwise we could improve the objective value by replacing ξ_i with $\left|\left(w^*\right)^T x_i + b^* - y_i\right|$. We claim that (w^*, b^*) is an optimal solution of (3) . Assume it is not and $\left(w^\#, b^\#\right)$ is the optimal solution. Let $\xi^\# \triangleq \left|\left(w^\#\right)^T x_i + b^\# - y_i\right|$. Then $\left(w^\#, b^\#, \xi^\#\right)$ is a feasible point of (4) with better objective value than (w^*, b^*, ξ^*), which is contradiction.

2. Assume (w^*, b^*) is an optimal solution of (3). Let $\xi_i^* \triangleq \left|\left(w^*\right)^T x_i + b^* - y_i\right|$. We claim that (w^*, b^*, ξ^*) will be an optimal solution of (4). This is true, because otherwise from a better solution $\left(w^\#, b^\#, \xi^\#\right)$ of (4), we could construct a better solution $\left(w^\#, b^\#\right)$ of (3).

Problem 8. Given a directed graph $G = (V, E)$ with non-negative edge capacities $c : E \to R^+$ and two distinct vertices $s, t \in V$. Formulate the problem of finding a maximum $s - t$ flow as a linear program. Now consider adding edge costs $p(e) \geq 0$ - the price of sending a unit of flow along the given edge. The cost of a flow is defined as $\sum_{e \in E} p(e) f(e)$. Show that finding a maximum flow with minimum cost can be solved in polynomial time using linear programming.

Solution 8. The maximum flow problem can be formulated as a linear problem as follows:

$$\begin{aligned}
\max_{f} \quad & \sum_{u:e=(s,u)\in E} f_e \\
\text{s.t.} \quad & \sum_{v:e=(u,v)\in E} f_e = \sum_{v:e=(v,u)\in E} f_e, \quad \forall u \in V\backslash\{s,t\} \\
& f_e \geq 0, \qquad \forall e \in E, \\
& f_e \leq c_e, \qquad \forall e \in E,
\end{aligned} \tag{5}$$

To find minimum cost maximum flow we can first find the size of the maximum flow in the graph. Let F denote the size of the maximum flow. Then finding minimum cost flow with size F can be formulated as a linear problem as follows:

$$\begin{aligned}
\min_{f} \quad & \sum_{e\in E} p(e) f_e \\
\text{s.t.} \quad & \sum_{v:e=(u,v)\in E} f_e = \sum_{v:e=(v,u)\in E} f_e, \quad \forall u \in V\backslash\{s,t\}, \\
& \sum_{u:e=(s,u)\in E} f_e = F \\
& 0 \leq f_e \leq c_e, \qquad \forall e \in E,
\end{aligned} \tag{6}$$

Problem 9. In the graph coloring problem we are given an undirected graph $G = (V, E)$. The goal is to color the nodes of the graph with minimum number of colors, such that adjacent nodes have different colors. Formulate this problem as an integer linear program.

Solution 9. Note that n colors are enough to color any graph with at most n nodes. Let $V = \{v_1, \ldots, v_n\}$. We introduce n^2 binary variables $x \in \{0,1\}^{n\times n}$, where $x_{i,j}$ denotes whether node v_i has color j. Additionally, we introduce one binary variable u_j for each color j, denoting whether color j is used or not. With this variables, we can write the graph coloring problem as follows:

$$\begin{aligned}
\min_{x,u} \quad & u_1 + u_2 + \ldots + u_n \\
\text{s.t.} \quad & x_{i,j} + x_{i',j} \leq 1, && \forall j \text{ and } \forall (i,i') \text{ s.t. } (v_i, v_{i'}) \in E, && \text{(adjacent nodes have different colors)} \\
& \sum_{j=1}^{n} x_{i,j} = 1 && i = 1, \ldots, n, && \text{(one color for each node)} \\
& x_{i,j} \leq u_j && i = 1, \ldots, n, \quad j = 1, \ldots, n, && \text{(using active colors only)} \\
& 0 \leq u_j \leq 1, && j = 1, \ldots n, \\
& 0 \leq x_{i,j} \leq 1 && i = 1, \ldots n, \quad j = 1, \ldots n \\
& u_j \in \mathbb{Z}, x_{i,j} \in \mathbb{Z} && i = 1, \ldots n, \quad j = 1, \ldots, n
\end{aligned} \tag{7}$$

Problem 10. Suppose there are n facilities and m customers. We wish to choose

1. which of the n facilities to open
2. which (open) facilities to serve m customers

Let f_i denote the cost of opening facility i. Let $c_{i,j}$ denote the cost of serving customer j using facility i. The goal is to serve all customers with minimum total cost (i.e., total price of opening facilities and total price of serving costumers). Formulate this problem as an integer linear program.

Solution 10. Let $s_i \in \{0,1\}$ denote whether we open a facility i. Let $x_{i,j} \in \{0,1\}$ denote whether the costumer j is served by facility i. The "tricky" part of formulating this problem as an integer linear problem is to write down the constraint that not opened facilities cannot serve and opened facilities can serve anyone. This can be done by writing $\sum_j x_{i,j} \leq M s_i$, where M is a sufficiently large number. In this problem, it is enough to set M=m.

$$
\begin{aligned}
\min_{x,s} \quad & \sum_{i=1}^{n} f_i s_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} x_{i,j} & & \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{i,j} = 1 & & j = 1,\dots,m, \quad \text{(each costumer is served)} \\
& \sum_{j=1}^{m} x_{i,j} \leq m s_i & & i = 1,\dots,n \qquad \text{(serving constraints)} \\
& 0 \leq s_i \leq 1 & & i = 1,\dots,n \\
& 0 \leq x_{i,j} \leq 1 & & i = 1,\dots,n, \quad j = 1,\dots,m \\
& s_i \in \mathbb{Z}, x_{i,j} \in \mathbb{Z}, & & i = 1,\dots,n, \quad j = 1,\dots,m
\end{aligned}
$$

Problem 11. In the traveling salesman problem we are given a weighted undirected graph $G = (V, E)$. The goal is to find a shortest walk that starts at some node, visits each node exactly once, and returns to the starting node. Show that this problem can be formulated as an integer linear program. In this problem you are allowed to write exponentially many constraints, but you need to provide a polynomial-time algorithm that checks whether a given assignment of variables is feasible or not, and if it is not feasible returns a violated constraint.

Solution 11. Please search "Miller-Tucker-Zemlin TSP formulation" for a TSP formulation as an integer linear program that uses polynomial number of constraints. Here we describe another formulation which has exponentially many constraints, but allows polynomial-time membership and separation oracles.

Let G be connected, and $w(e)$ denote the weight/length of the edge e. We keep one binary variable x_e for each edge, denoting whether edge e is selected. We require each node to have exactly one incoming selected edge and exactly one outgoing selected edge. With this constraints, the selected edges will form number of directed cycles. The only constraint we need to add is that selected edges form exactly one cycle. This holds if and only if for each $S \subsetneq V$, the number of selected edges between nodes of S is no more than $|S| - 1$. To see this, assume that selected cycles form at least two cycles. Let $C_1 \neq V$ be one of the cycles. Then the number of edges selected from C_1 is equal to $|C_1|$. Conversely, assume $\exists S \subsetneq V$, for which the number of selected edges is at least $|S|$. Then either there exists a vertex in S that at least two incoming edges, or there exists a vertex in S that has two outgoing edges, or the selected edges form a single directed cycle spanning S. All these cases imply that the walk is invalid. With this observations, the formulation becomes:

$$
\begin{aligned}
\min_{x} \quad & \sum_{e \in E} w(e) x_e \\
\text{s.t.} \quad & \sum_{e=(v,u) \in E} x_e = 1, && \forall u \in E, \\
& \sum_{e=(u,v) \in E} x_e = 1, && \forall u \in E, \\
& \sum_{e=(u,v) \in E, u \in S, v \in S} x_e \leq |S| - 1, && \forall S \subsetneq V, \\
& 0 \leq x_e \leq 1, && \forall e \in E, \\
& x_e \in \mathbb{Z}, && \forall e \in E.
\end{aligned} \tag{8}
$$

Note that checking whether a given set of selected edges form a valid walk is easy and can be done in linear time. Finding a violated constraint is also easy. First we can check whether the constraints that each node has exactly one incoming and one outgoing edge. Then if that holds, we can check whether the selected edges from a single cycle or multiple cycles. If they form a multiple cycles, then by selecting S to be one of those cycles, we find a violateed constraint.

Problem 12. Read chapter 11.6 of "Algorithm Design" by Jon Kleinberg and Eva Tardos, which describes a 2-approximation algorithm for vertex covering that uses LP relaxation.

Solution 12. Read the section!

Problem 13. Using LP relaxation and rounding provide a polynomial-time approximation algorithm for set covering problem that finds a $\Theta(\log |G|)$-approximation with at least $1/2$ probability. For the definition of the set covering problem refer to Problem 5. Hint: apply randomized rounding $\Theta(\log |G|)$ times and combine these $\log |G|$ rounded points.

Solution 13. Let $G = \{u_1, \ldots, u_m\}$ be the ground set and $S_1, \ldots S_n$ be subsets of G. Let $x_i \in \{0, 1\}$ denote whether we select the i-th set or not. As we see in Problem 5 we can formalize the set cover problem as follows:

$$
\begin{aligned}
\min_{x} \quad & x_1 + x_2 + \cdots + x_n \\
\text{s.t.} \quad & \sum_{i : S_i \ni u_j} x_i \geq 1, && j \in G, \\
& 0 \leq x_i \leq 1, && i = 1, \ldots n, \\
& x_i \in \mathbb{Z}, && i = 1, \ldots n,
\end{aligned}
$$

Now we remove the integrality constraints, getting a standard linear program. Let LP-OPT be the optimal value of it and x^* be an optimal solution of it. Obviously, LP-OPT $\leq$ OPT. Let us consider a randomized rounding:

$$
\bar{x}_i^* = \begin{cases} 1 & \text{with probability } x_i^* \\ 0 & \text{with probability } (1 - x_i^*) \end{cases}
$$

Note that in expectation $\bar{x}^*$ selects LP-OPT number of sets. Let $\bar{c}_j$ denote whether u_j is covered after the rounding. WLOG, we can assume that $S_1, S_2, \ldots S_r$ contains the item u_j.

We now, compute the probability of an element being covered.

$$\begin{aligned}
\mathbb{P}[\bar{c}_j = 1] &\geq 1 - \prod_{i=1}^{r} \mathbb{P}[\bar{x_i}^* = 0] \\
&= 1 - \prod_{i=1}^{r} (1 - x_i) \\
&\geq 1 - \left(\frac{\sum_{i=1}^{r} 1 - x_i}{r} \right)^r && \text{(AM-GM inequality)} \\
&\geq 1 - \left(1 - \frac{1}{r} \right)^r \\
&\geq 1 - 1/e
\end{aligned}$$

The idea is that if we consider k random roundings and take the union of the selected sets of all roundings, the probability of u_j remaining uncovered will be equal to $(1/e)^k$, which can be made small by picking large k. Furthermore, the probability of at least one element remaining uncovered can be upper bounded by $m\frac{1}{e^k}$ using the union bound. If we want this probability at most $\frac{1}{m}$, we can set $k = 2 \ln m$. Summing up, we provided a randomized algorithm that in expectation selects at most $(2 \ln m \mathrm{LP} - \mathrm{OPT}) \leq (2 \ln m \mathrm{OPT})$ sets and covers all elements with probability at least $1 - 1/m$.