EPFL

Midterm Exam, Advanced Algorithms 2020-2021

- You are allowed to consult lectures notes of the course, but **no outside material**.
- **Communication is not allowed.**
- Your explanations should be clear enough and in sufficient detail that a fellow student can understand them. In particular, do not only give pseudo-code without explanations. A good guideline is that a description of an algorithm should be such that a fellow student can easily implement the algorithm following the description.
- You are allowed to refer to material covered in the lecture notes including theorems without reproving them.

Good luck!

Name: _____ **N° Sciper:** _____

Problem 1 / 22 points	Problem 2 / 30 points	Problem 3 / 20 points	Problem 4 / 28 points

Total / 100

1 *(22 pts)* **LP duality.** Write down the duals of the linear programs below together with complimentary slackness conditions.

1a

$$\begin{aligned} \text{minimize} \quad & 3x_1 + 4x_2 \\ \text{s. t.} \quad & x_1 + 2x_2 \geq 1 \\ & 2x_1 + 4x_2 \geq 5 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Solution: The dual problem is

$$\begin{aligned} \text{maximize} \quad & y_1 + 5y_2 \\ \text{s.t.} \quad & y_1 + 2y_2 \leq 3 \\ & 2y_1 + 4y_2 \leq 4 \\ & y_1, y_2 \geq 0 \end{aligned}$$

Complimentary slackness conditions are

$$\text{x; y are both optimal solutions} \Leftrightarrow \begin{cases} x_1 > 0 \Rightarrow y_1 + 2y_2 = 3 \\ x_2 > 0 \Rightarrow 2y_1 + 4y_2 = 4 \\ y_1 > 0 \Rightarrow x_1 + 2x_2 = 1 \\ y_2 > 0 \Rightarrow 2x_1 + 4x_2 = 5 \end{cases}$$

1b

$$\begin{aligned} \text{maximize} \quad & 3x_1 + 4x_2 + x_3 \\ \text{s. t.} \quad & x_1 + 2x_2 - x_3 = 0 \\ & x_1 + 2x_3 \leq 3 \\ & 5x_1 + x_2 \leq 2 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Solution: First we need to convert this problem to a standard form. We convert minimization problem into a maximization one by multiplying the loss function with -1. Then, we convert an equality $x_1 + 2x_2 - x_3 = 0$ by replacing it with two inequalities: $x_1 + 2x_2 - x_3 \geq 0$ and $x_1 + 2x_2 - x_3 \leq 0$. Last, we convert all the inequalities with a sign $\leq$ to inequalities with a sign $\geq$ by multiplying these inequalities with -1.

After these transformations, our problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & -3x_1 - 4x_2 - x_3 \\ \text{s.t.} \quad & x_1 + 2x_2 - x_3 \geq 0 \\ & -x_1 - 2x_2 + x_3 \geq 0 \\ & -x_1 - 2x_3 \geq -3 \\ & -5x_1 - x_2 \geq -2 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

The dual of this problem is

$$\begin{aligned}
\text{maximize} \quad & -3y_3 - 2y_4 \\
\text{s.t.} \quad & y_1 - y_2 - y_3 - 5y_4 \le -3 \\
& 2y_1 - 2y_2 - y_4 \le -4 \\
& -y_1 + y_2 - 2y_3 \le -1 \\
& y_1, y_2, y_3, y_4 \ge 0
\end{aligned}$$

Complimentary slackness conditions are

$$\text{x; y are both optimal solutions} \Leftrightarrow \begin{cases}
x_1 > 0 \Rightarrow y_1 - y_2 - y_3 - 5y_4 = -3 \\
x_2 > 0 \Rightarrow 2y_1 - 2y_2 - y_4 = -4 \\
x_3 > 0 \Rightarrow -y_1 + y_2 - 2y_3 = -1 \\
y_1 > 0 \Rightarrow x_1 + 2x_2 - x_3 = 0 \\
y_2 > 0 \Rightarrow -x_1 - 2x_2 + x_3 = 0 \\
y_3 > 0 \Rightarrow -x_1 - 2x_3 = -3 \\
y_4 > 0 \Rightarrow -5x_1 - x_2 = -2
\end{cases}$$

2 *(30 pts)* **Optimal vertex removal.** Suppose that you are given a directed graph $G = (V, E)$ together with an assignment of costs to vertices $c : V \to \mathbb{R}_+$ and a set of vertices $S \subseteq V$ that form an independent set (i.e. none of the vertices in S are neighbors in G). We say that a subset $R \subseteq V \setminus S$ disconnects S if no vertex in S can reach another vertex in S when vertices in R are removed together with all their edges. Your task is to find the cheapest set of vertices $R \subseteq V \setminus S$ that disconnects S, where the cost of a set $R \subseteq V$ is defined as $\sum_{v \in R} c_v$.

2a *(8 pts)* Let $\mathcal{P}$ denote the set of simple[1] paths in G connecting pairs of vertices in S (here a path $P = (u_1, u_2, \ldots, u_k)$ is a sequence of vertices of G such that for every $i = 1, \ldots, k-1$ one has $(u_i, u_{i+1}) \in E$; a path is simple if it contains no repeated **vertices**). Consider the linear program below:

$$\begin{aligned}
\text{minimize} \quad & \sum_{v \in V \setminus S} c_v d_v \\
\text{s. t.} \quad & \sum_{v \in P, v \in V \setminus S} d_v \ge 1 \qquad \text{for every } P \in \mathcal{P} \\
& d_v \ge 0 \qquad\qquad\qquad \text{for all } v \in V \setminus S.
\end{aligned}$$

Prove that for every $R \subseteq V \setminus S$ that disconnects S there exists a feasible solution $(d_v)_{v \in V \setminus S}$ to the LP above with cost bounded by the cost of R.

Solution: For any $R \subseteq V \setminus S$ we can simply set $d_v = 1$ for every $v \in R$ and $d_v = 0$ for every $v \notin R$. This is a feasible solution: Since R disconnects S, any path $P \in \mathcal{P}$ must contain a vertex v^* in R. Therefore,

$$\sum_{v \in P, v \in V \setminus S} d_v \ge d_{v^*} = 1.$$

Furthermore, the cost of this solution is indeed the cost of R:

$$\sum_{v \in V \setminus S} c_v d_v = \sum_{v \in R} c_v.$$

[1] We call a path simple if it contains no repeated **vertices**.

2b *(11 pts)* Write down the dual of the LP in **(a)** together with complimentary slackness conditions.

Solution: For each path $P \in \mathcal{P}$ we introduce a variable x_P. Then the dual LP is as follows:

$$\begin{aligned} \text{maximize } & \sum_{P \in \mathcal{P}} x_P \\ \text{s. t. } & \sum_{P \ni v} x_P \leq c_v && \text{for all } v \in V \setminus S \\ & x_P \geq 0 && \text{for all } P \in \mathcal{P}. \end{aligned}$$

Complementary slackness states that for any pair of solutions d and x (to the primal and dual LP's respectively), both solutions are optimal if and only if

$$\text{for all } v \in V \setminus S: \ d_v = 0 \text{ or } \sum_{P \ni v} x_P = 1.$$

Alternately one can say that both solutions are optimal if and only if

$$\text{for all } P \in \mathcal{P}: \ x_P = 0 \text{ or } \sum_{v \in P, v \in V \setminus S} d_v = 1.$$

2c *(11 pts, half ⋆)* Show how, given a candidate solution $d = (d_v)_{v \in V \setminus S}$, one can in polynomial time check whether d is feasible for the LP above, and find a violated constraint if d is not feasible.

Solution: For a given vector d, we must determine if there are any paths P between two distinct point of S, such that

$$\sum_{v \in P, v \in V \setminus S} d_v < 1.$$

We modify the graph G into G' to turn this into a shortest path problem. Let $G' = (V', E')$ also be a directed graph. Let V' consist of vertices of S, as well as vertices v^{in} and v^{out} for all vertices $v \in V \setminus S$. For every directed edge $(u, v) \in E$, let there be a corresponding directed edge $(u', v') \in E'$ where $u' = u$ if $u \in S$ and $u' = u^{\text{out}}$ if $u \notin S$, as well as $v' = v$ if $v \in S$ and $v' = v^{\text{in}}$ if $v \notin S$. Furthermore, let E' contain the edge $(v^{\text{in}}, v^{\text{out}})$ for each $v \in V \setminus S$.

There is a natural, one-to-one correspondence between paths of G connecting two vertices in S, and paths of G' connecting two vertices in S. Indeed, the path $P = (u_1, u_2, \ldots, u_k)$ corresponds to the path where each u_i is either left as it is (if $u_i \in S$), or replaced by $u_i^{\text{in}}, u_i^{\text{out}}$ (if $u_i \notin S$). We call this new path P'. Note that u_1 and u_k are both in S by definition. Let us call the set of simple paths in G', connecting two distinct points in S, $\mathcal{P}'$.

We give weights to edges of E': Any edge of the form $(v^{\text{in}}, v^{\text{out}})$ has weight d_v, while all other edges have weight zero. Note that for all $P \in \mathcal{P}$

$$\sum_{e' \in P'} w(e') = \sum_{v \in P, v \in V \setminus S} d_v.$$

Therefore, it suffices to verify that the shortest path in $\mathcal{P}'$ is of length at least one.

Since we are required only to give a polynomial time algorithm, this is very simple. For example, one can run Dijkstra's Algorithm from each vertex of S in G'. If there is no path shorter than one in $\mathcal{P}'$, d is feasible. If there is such a path, the equivalent path in G corresponds to a violated constraint.

3 *(20 pts)* **Collaborative basis.** In the collaborative basis problem $d \geq 2$ participants are given $d \times n$ matrices $A_1, \ldots, A_d$ for some $n \geq 1$. Their task is to select one column from each of the matrices $A_i, i = 1, \ldots, d$, so that the selected columns form a basis for the entire space $\mathbb{R}^d$. Give an efficient algorithm that, given matrices $A_1, \ldots, A_d$, outputs **YES** if such a collection of columns exists and **NO** otherwise.

Example 1. Suppose that $d = 2, n = 4$, and matrices A_1, A_2 are given by

$$A_1 = \begin{pmatrix} 5 & 2 & 1 & 2 \\ 5 & 1 & 5 & 0 \end{pmatrix} \quad \text{and} \quad A_2 = \begin{pmatrix} 1 & 0 & 2 & 3 \\ 1 & 0 & 2 & 3 \end{pmatrix}.$$

Then taking the second column of A_1 and the first column of A_2, we obtain a basis for $\mathbb{R}^2$. Indeed, the matrix

$$\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

is full rank, so the answer is **YES**.
Example 2. Suppose that $d = 3, n = 2$, and matrices are given by

$$A_1 = \begin{pmatrix} 1 & 2 \\ -2 & 2 \\ 1 & 4 \end{pmatrix}, A_2 = \begin{pmatrix} -1 & 3 \\ 2 & 0 \\ -1 & 5 \end{pmatrix} \text{ and } A_3 = \begin{pmatrix} -2 & 6 \\ 4 & 0 \\ -2 & 10 \end{pmatrix}$$

Here one notes that columns of A_3 are just a scaled version of columns of A_2, and columns of A_2 can be obtained by taking a linear combination of columns of A_1, so it is not possible to choose one column from each matrix to obtain a basis for $\mathbb{R}^3$.

Hint: use matroids. You may use the fact that, given a collection of k vectors in $\mathbb{R}^d$, one can check if the vectors are linearly independent (i.e., if the matrix whose columns are these vectors has rank k) in time polynomial in k and d.

Solution: Let the ordered pair (i, j) correspond to the j-th column of A_i. Let $E = [d] \times [n]$ be the set of all column indices. Define I_1 to contain all subsets of E such as X where the matrix containing the columns included in X has rank $|X|$. The matroid $\mathcal{M}_1 = (E, I_1)$ is a linear matroid. Furthermore define I_2 to contain all subsets of E with at most one column from each matrix. Setting $E_i = \{i\} \times [n]$ and $k_i = 1$ it is clear to see that the matroid $\mathcal{M}_2 = (E, I_2)$ is a partition matroid.

Now note that we are interested in finding an independent set in the intersection of these two matroids. Furthermore, using the given hint the inclusion of a set X in I_1 can be checked in polynomial time. Checking if X is in I_2 can also be done in polynomial time by comparing the matrix index of all pairs in X in $O(|X|^2)$. Therefore Edomond's theorem gives an efficient algorithm for finding a max weight independent set in the intersection of $\mathcal{M}_1$ and $\mathcal{M}_2$. Setting the weight of all columns to 1, our algorithm should answer YES if the found independent set contains d columns and should answer NO otherwise.

4 *(28 pts)* **Matrix reconstruction.** A well known advertisement agency decided to hire you to find the right advertisement placement strategy for their clients. Suppose that there are n types of billboards in Lausanne and m clients, and you used a linear program to determine how many billboards of each type every client should use. Specifically, your LP solver produced an $n \times m$ matrix A, where for $i = 1, \ldots, n$ and $j = 1, \ldots, m$ the (i, j)'th entry of the matrix shows how many billboards of type i the j'th client should use. There is a problem though: the entries in the matrix are not integers. At the very least they are non-negative, however, and every row sum as well as every column sum is an integer. You will design an efficient algorithm to round the matrix entries to integers with the following constraints:

- Any non-integer element x in the matrix can only be replaced by $\lfloor x \rfloor$ or $\lceil x \rceil$.
- In the output matrix, for any row (or column), the sum of entries in that row (or column) should remain the same as in the initial matrix.

Note: there might be many correct output matrices for a given initial matrix, and you only need to output one of them. You should design the algorithm, prove its correctness and establish runtime bounds.

(Hint: use ideas developed in class for a problem on graphs)
Example 1: The matrix

$$\begin{pmatrix} 1 & 2.2 & 1 & 5.8 \\ 3 & 0 & 1 & 2 \\ 2 & 1.8 & 3 & 0.2 \end{pmatrix}$$

can be rounded to the matrix

$$\begin{pmatrix} 1 & 2 & 1 & 6 \\ 3 & 0 & 1 & 2 \\ 2 & 2 & 3 & 0 \end{pmatrix}.$$

Example 2: The matrix

$$\begin{pmatrix} 0.3 & 0.3 & 0.3 & 0.1 \\ 0.2 & 0.5 & 0.1 & 0.2 \\ 0.5 & 0.2 & 0.6 & 0.7 \end{pmatrix}$$

can be rounded to the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

Solution: We can convert this problem to a graph problem as follows. Create a bipartite graph $G(V, E)$, with $V = R \cup C$ and $R \cap C = \emptyset$, where each vertex in R correspond to a row of our matrix and each vertex in C correspond to a column of our matrix. Now, connect each vertex in R to all vertices in C (complete bipartite graph). Assign value $A_{i,j}$ to edge connecting i'th vertex in R to j'th vertex in C. Now, the problem looks very similar to what we had in the lecture notes proving that any extreme point solution to the maximum weight bipartite matching LP is integral. We apply the same approach here: Since we know that sum of each row and each column is an integer, we conclude that if we only consider non-integer valued edges we can find an even sized cycle in this graph. Now, let the values of edges in this cycle be $x_1, x_2, \ldots, x_{2k}$. Let $\epsilon = \min_{i \in [2k]} \{x_i - \lfloor x_i \rfloor, 1 - (x_i - \lfloor x_i \rfloor)\}$. Color the edges of this cycle in an alternating fashion and subtract (or add) ϵ to value of edges of one color and add (or subtract) ϵ from value of edges

of the other color, so that value of at least one edge becomes integer. Note that this way, we guarantee three events:

- At least value of one edge becomes integer.
- If value of an edge was $\in [r, r+1]$ for $r \in \mathbb{Z}$ before this operation, it remains in that interval after the operation.
- For any row and column the sum is unchanged.

So, in each step value of at least one edge becomes integer and this algorithm will terminate when all edges get integer values.

Runtime analysis: Let n_r be the number or rows and n_c be the number of columns of matrix. Then the graph has $n_r n_c$ edges. In order to find cycles one can run DFS which takes time $O(n_r n_c)$ time, and this procedure can continue for at most $n_r n_c$ rounds (since at each round we make at least one edge integer valued). So, the total runtime is $O\left(n_r^2 n_c^2\right)$. However, one can find such a cycle takes $O(n_r + n_c)$ time (more specifically $O(\max\{n_c, n_r\})$), using DFS: one starts from a vertex and explores new vertices in each round, and as soon as one visits a vertex that has already visited, one stops. Then, the runtime becomes $O(n_r n_c (n_r + n_c))$.