ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Midterm Exam, Advanced Algorithms 2016-2017

- You are only allowed to have a handwritten A4 page written on both sides.
- Communication, calculators, cell phones, computers, etc... are 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 algorithms covered in class without reproving their properties.
- **Do not touch until the start of the exam.**

Good luck!

Name: _____ **N° Sciper:** _____

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

Total / 100

1 *(consisting of subproblems **a**-**b**, 30 pts)* **Basic questions.** This problem consists of two subproblems that are each worth 15 points.

1a *(15 pts)* Suppose we use the Simplex method to solve the following linear program:

$$\begin{aligned} \textbf{maximize} \quad & 2x_1 - x_2 \\ \textbf{subject to} \quad & x_1 - x_2 + s_1 = 1 \\ & x_1 + s_2 = 4 \\ & x_2 + s_3 = 2 \\ & x_1,\ x_2,\ s_1,\ s_2,\ s_3 \geq 0 \end{aligned}$$

At the current step, we have the following Simplex tableau:

$$\begin{aligned} x_1 &= 1 + x_2 - s_1 \\ s_2 &= 3 - x_2 + s_1 \\ s_3 &= 2 - x_2 \\ \hline z &= 2 + x_2 - 2s_1 \end{aligned}$$

Write the tableau obtained by executing one iteration (pivot) of the Simplex method starting from the above tableau.

1b *(15 pts)* In the maximum directed cut problem we are given as input a directed graph $G = (V, A)$. Each arc $(i, j) \in A$ has a nonnegative weight $w_{ij} \geq 0$. The goal is to partition V into two sets U and $W = V \setminus U$ so as to maximize the total weight of the arcs going from U to W (that is, arcs (i, j) with $i \in U$ and $j \in W$). Give a randomized 1/4-approximation algorithm for this problem (together with a proof that it is a 1/4-approximation in expectation).

2 *(20 pts)* **Spanning trees with colors.** Consider the following problem where we are given an edge-colored graph and we wish to find a spanning tree that contains a specified number of edges of each color:

Input: A connected undirected graph $G = (V, E)$ where the edges E are partitioned into k color classes $E_1, E_2, \ldots, E_k$. In addition each color class i has a target number $t_i \in \mathbb{N}$.

Output: If possible, a spanning tree $T \subseteq E$ of the graph satisfying the color requirements:

$$|T \cap E_i| = t_i \qquad \text{for } i = 1, \ldots, k.$$

Otherwise, i.e., if no such spanning tree T exists, output that no solution exists.

Design a polynomial time algorithm for the above problem. You should analyze the correctness of your algorithm, i.e., why it finds a solution if possible. To do so, you are allowed to use algorithms and results seen in class without reexplaining them.

(This page is intentionally left blank.)

(Primal) LP Relaxation

$$\begin{aligned} \textbf{minimize} \quad & \sum_{S \in \mathcal{T}} c(S) x_S \\ \textbf{subject to} \quad & \sum_{S \in \mathcal{T}: e \in S} x_S \geq 1 \quad \text{for } e \in \mathcal{U} \\ & x_S \geq 0 \quad \text{for } S \in \mathcal{T} \end{aligned}$$

(Dual)

$$\begin{aligned} \textbf{maximize} \quad & \sum_{e \in \mathcal{U}} y_e \\ \textbf{subject to} \quad & \sum_{e \in S} y_e \leq c(S) \quad \text{for } S \in \mathcal{T} \\ & y_e \geq 0 \quad \text{for } e \in \mathcal{U} \end{aligned}$$

Figure 1. The standard LP relaxation of set cover and its dual.

3 *(20 pts)* **Set Cover.** (For this problem, it is good to recall the standard linear programming relaxation of set cover and its dual, see Figure 1.)

You just graduated from EPFL and you started your first inspiring job. To your surprise, you immediately find yourself in the midst of the action as the company's future depends on obtaining a good solution to a set cover instance. The instance is specified by a universe $\mathcal{U} = \{e_1, \ldots, e_n\}$ of clients, a family of subsets $\mathcal{T} = \{S_1, S_2, \ldots, S_m\}$, and a cost function $c : \mathcal{T} \to \mathbb{R}_+$. The task is to find a collection $C \subseteq \mathcal{T}$ of subsets of minimum total cost that covers all elements.

After spending a good month on the problem, you have found a super heuristic that outputs (what you think) is a great solution C^*. However, as it turns out, your boss studied mathematics back in the 60's and so she is not convinced by your heuristic arguments. You therefore contact EPFL, who use their supercomputers to compute a dual certificate. They found a dual certificate y^*, which is a feasible solution to the dual of the LP relaxation of set cover satisfying the following:

- For every $S \in C^*$, we have $\sum_{e \in S} y_e^* \geq c(S)/2$.
- For every $e \in \mathcal{U}$ with $y_e^* > 0$, we have $|\{S \in C^* : e \in S\}| \leq 2$.

Using the dual solution y^*, **prove that your solution C^* is in fact a 4-approximate solution**. Since the set cover problem is NP-hard to approximate within a factor better than $\ln(n)$ in general, the fact that you have found a much better solution for the given instance will rescue the company and almost guarantee you a promotion.

(This page is intentionally left blank.)

4 *(consisting of subproblems **a**-**b**; note that you can solve subproblem **b** assuming the solution to **a** without solving it; 30 pts)*

Efficient job scheduling. In this problem we are going to investigate the linear programming relaxation of a classical scheduling problem. In the considered problem, we are given a set M of m machines and a set J of n jobs. Each job $j \in J$ has a processing time $p_j > 0$ and can be processed on a subset $N(j) \subseteq M$ of the machines. The goal is to assign each job j to a machine in $N(j)$ so as to complete all the jobs by a given deadline T. (Each machine can only process one job at a time.)

If we, for $j \in J$ and $i \in N(j)$, let x_{ij} denote the indicator variable indicating that j was assigned to i, then we can formulate the scheduling problem as the following integer linear program:

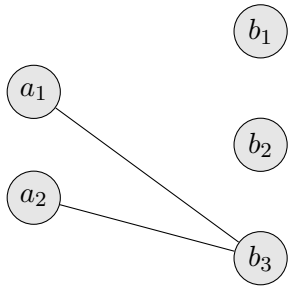
$$\sum_{i \in N(j)} x_{ij} = 1 \qquad \text{for all } j \in J \qquad \textit{(Each job } j \textit{ should be assigned to a machine } i \in N(j)\textit{)}$$

$$\sum_{j \in J : i \in N(j)} x_{ij} p_j \leq T \qquad \text{for all } i \in M \qquad \textit{(Time needed to process jobs assigned to } i \textit{ should be} \leq T\textit{)}$$

$$x_{ij} \in \{0,1\} \text{ for all } j \in J,\ i \in N(j)$$

The above integer linear program is NP-hard to solve, but we can obtain a linear programming relaxation by relaxing the constraints $x_{ij} \in \{0,1\}$ to $x_{ij} \in [0,1]$. The obtained linear program can be solved in polynomial time using e.g. the ellipsoid method.

Example. An example is as follows. We have two machines $M = \{m_1, m_2\}$ and three jobs $J = \{j_1, j_2, j_3\}$. Job j_1 has processing time $1/2$ and can only be assigned to m_1; job j_2 has processing time $1/2$ and can only be assigned to m_2; and job j_3 has processing time 1 and can be assigned to either machine. Finally, we have the "deadline" $T = 1$. An extreme point solution to the linear programming relaxation is $x^*_{11} = 1, x^*_{22} = 1, x^*_{13} = 1/2$ and $x^*_{23} = 1/2$. The associated graph H (defined in subproblem **a**) can be illustrated as follows:



4a *(15 pts)* Let x^* be an extreme point solution to the linear program and consider the (undirected) bipartite graph H associated to x^* defined as follows. Its left-hand-side has a vertex a_i for each machine $i \in M$ and its right-hand-side has a vertex b_j for each job $j \in J$. Finally, H has an edge $\{a_i, b_j\}$ iff $0 < x^*_{ij} < 1$.

Prove that H is acyclic (using that x^* is an extreme point).

4b *(15 pts)* Use the structural result proved in the first subproblem to devise an efficient rounding algorithm that, given an instance and a feasible extreme point x^* in the linear programming relaxation corresponding to the instance, returns a schedule that completes all jobs by deadline $T + \max_{j \in J} p_j$. In other words, you wish to assign jobs to machines so that the total processing time of the jobs a machine receives is at most $T + \max_{j \in J} p_j$.