Advanced algorithms

INF550 – Exercise sheet #1 – Linear programming – Solutions

September 21, 2022

Exercice 1 (Shortest path)**.** Given a weighted directed graph $G = (V, E)$ with positive weights, and two distinguished vertices s and t, we consider the problem of finding the shortest path from s to t (where the length of a path is the sum of the weights of the edges of the path).

To each vertex v is associated a variable $d(v)$. Consider the following linear problem:

$$\begin{aligned} &\text{maximize } d(t) - d(s) \\ &\text{subject to } \forall (u,v) \in E, d(v) - d(u) \leqslant c(u,v). \end{aligned}$$

(a) Show that for any assignment of the variables $d(v)$ in the feasible set, the difference $d(t)-d(s)$ is a lower bound on the length of the shortest path from s to t.

Solution — Let $\gamma = (x_i)_{1\leqslant i\leqslant n}$ with $x_1 = s$, $x_n = t$ and $(x_i, x_{i+1}) \in E$ be the shortest path from s to t. If the variable $d(v)$ are assigned a feasible solution, then

$$d(t) - d(s) = \sum_{i=1}^{n-1} d(x_{i+1}) - d(x_i) \leqslant \sum_{i=1}^{n-1} c(x_i, x_{i+1}),$$

which is exactly the length of the path γ.

(b) Show that assigning to $d(v)$ the length of the shortest path from s to v is feasible.

Solution — Let (u, v) be an edge of E. The shortest path from s to u can be extended as a path from s to v by adding the edge (u, v), giving a path of length $d(u) + c(u, v)$. Thus $d(v) \leqslant d(u) + c(u, v)$.

(c) Conclude that the optimal value of the linear program is exactly the length of the shortest path from s to t.

Solution — Follows directly from the previous questions.

(d) Write the dual linear program and interpret it.

Solution — Let's rewrite the primal problem (P) in matrix form:

$$\begin{cases} \textit{Maximize} & c^\top x \\ \textit{Subject to} & Ax \leq b \end{cases}$$

With c^T being indexed by V and A having a row for each edge (a,b) and a column for each vertex v:

$$A((a,b),v) = \begin{cases} -1 & \text{if } a = v \\ 1 & \text{if } b = v \\ 0 & \text{otherwise} \end{cases} = \mathbb{1}_{b=v} - \mathbb{1}_{a=v}$$

and

$$c^T(v) = \begin{cases} -1 & \text{if } v = s \\ 1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} = \mathbb{1}_{v=t} - \mathbb{1}_{v=s}$$

Furthermore, the variables of (P) have unconstrained *sign. Therefore the dual problem (D) will be of the form*

$$\begin{cases} \text{Minimize} & y^\top b \\ \\ \text{Subject to} & y^\top A = c \\ & y \geq 0 \end{cases}$$

- Variables - *In (P), the inequality constraints are in one-to-one correspondance with the edges. Therefore, (D) will have one variable $y_{u,v} \geq 0$ for each edge (u,v).*
- Constraints - *For each vertex $v \in V$,*

$$\begin{aligned} y^T A(v) &= \sum_{(a,b)\in E} y_{a,b} A((a,b),v) = \sum_{(a,b)\in E} y_{a,b}\mathbb{1}_{b=v} - \sum_{(a,b)\in E} y_{a,b}\mathbb{1}_{a=v} \\ &= \sum_{a:(a,v)\in E} y_{a,b} - \sum_{b:(v,b)\in E} y_{v,b} \end{aligned}$$

The dual program is

$$\begin{cases} \text{Minimize} & \sum_{(u,v)\in E} c(u,v) y_{u,v} \\ \\ \text{Subject to} & \forall (u,v) \in E, y_{u,v} \geqslant 0 \\ & \forall v \in V, \sum_{a:(a,v)\in E} y_{a,b} - \sum_{b:(v,b)\in E} y_{v,b} = \begin{cases} -1 & \text{if } v = s \\ 1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \end{cases}$$

It can be interpreted as a flow problem: The last constraint stands for the conservation of the flow, while the others mean that the total flow going out from

s is 1 and the total flow arriving at t is also 1. This is not *a max-flow problem: Here the total flow is fixed, and the edges have infinite capacity. Here we want to* minimize *the cost of sending a certain amount of flow. This problem is also known as* MIN-COST FLOW. *This exercise shows that this problem is dual to the* SHORTEST PATH *problem, as the* MIN-CUT *problem was the dual of the max-flow.*

Exercice 2 (Max-norm linear regression)**.** A physicist measures a quantity theoretically given by a linear function $y(x) = ax + b$. The results are points (x_i, y_i). He wants to find the best values for the coefficients a and b such that the vertical distance between the points (x_i, y_i) and the line $y = ax + b$ is minimized.

(a) Write this optimization problem as a three-variable linear program.

Solution — We introduce 3 variables: a, b the coefficients of the linear function, and m the maximal vertical distance between one point and the line. The question is then to minimize m with the constraint that m is greater than all vertical distances:

$$\begin{cases} \textit{Minimize} & m \\ \textit{s. t} & \forall i, |y_i - (ax_i + b)| \leq m \end{cases}$$

i.e.

$$\begin{cases} \textit{Minimize} & (0,0,1)\begin{pmatrix} a \\ b \\ m \end{pmatrix} \\ \textit{s. t} & \forall i,\ ax_i + b - m \leq y_i \\ & \forall i,\ -ax_i - b - m \leq -y_i \end{cases}$$

(b) Write the dual problem.

Solution — The inequalities are in $\leq$ form but the primal problem is a minimization *problem. In order to write the dual problem, we begin by multiplying each constraint by -1:*

$$\begin{cases} \textit{Minimize} & (0,0,1)\begin{pmatrix} a \\ b \\ m \end{pmatrix} \\ \textit{s. t} & \forall i,\ -ax_i - b + m \geq -y_i \\ & \forall i,\ ax_i + b + m \geq y_i \end{cases}$$

Since the variables are unconstrained *in the primal problem, the dual problem will only have equality constraints.*

We introduce the variables p_i for the first set of contraints and q_i for the other. Then the dual problem is

$$\begin{array}{lll} \textit{maximize} & \sum_i y_i(q_i - p_i) & \\ \textit{such that} & \sum_i (q_i - p_i)x_i = 0 & \textit{(constraint corresponding to a)} \\ & \sum_i (q_i - p_i) = 0 & \textit{(constraint corresponding to b)} \\ & \sum_i (q_i + p_i) = 1 & \textit{(constraint corresponding to m)} \\ & p_i \geqslant 0, \quad q_i \geqslant 0. & \end{array}$$

We can simplify this problem by substituting $q_i - p_i$ by a real variable α_i and $q_i + p_i$ by $|\alpha_i|$.

Remark - *While the primal problem has few variables, and a lot of constraints, the dual problem has only 3 constraints. When the number of points is big, the geometry of the dual might be easier to handle for LP programs.*

Exercice 3 (Preemptive scheduling on parallel computers)**.** A set of computational tasks $\{1, \ldots, n\}$ is to be executed simultaneously on m computers. Each task has a duration p_i and can be stopped and restarted arbitrarily on another computer. However, a task may run on a single computer at a time. A schedule is a plan describing which task will be executed on which computer at which time.

(a) Show that the duration of a schedule is at least $\max\left(\max_{1 \leqslant i \leqslant n} p_i, \frac{1}{m}\sum_{i=1}^{n} p_i\right)$.

Solution — All the work has to be done and there is only m computers, to the total time cannot be less that $\frac{1}{m}\sum_{i=1}^{n} p_i$. Moreover, a task cannot be run simultaneously on several computer, so the schedule cannot be shorter that the duration of the longest task.

(b) Show that there exists a schedule that achieves the bound and that it can be computed in $O(n)$ operations.

Solution —

Let $M_1, \ldots, M_m$ be all the machines, and let denote by M_c the current machine and by d the curent date. Initially, $M_c = M_1$ and $d = 0$. For $i = 1$ to n, we compute the date $f = d + p_i$. If $f \leq D$, then the task i can be executed on the current machine M_c between dates d and f. On the contrary, if $f > D$, task i needs to be executed in two parts: One on the current machine between dates d and D, and the other on M_{c+1} between dates 0 and $p_i - (D - d)$. This affectation is valid because the condition $D \geq p_i$ ensures that the two parts executed on the two different machines will not overlap. Moreover, the index of the current machine after running task i is $\lceil \sum_{k=1}^{i} p_k / D \rceil$, therefore the index of the current machine does not exceed m by definition of D.

It can be summed up in the following algorithm:

$D \leftarrow \max\left(\max_{1 \leqslant i \leqslant n} p_i, \frac{1}{m}\sum_{i=1}^{n} p_i\right)$
$c \leftarrow 1$ ▷ *current machine*
$d \leftarrow 0$ ▷ *current date*

for $i \in \{1, \ldots, n\}$ ***do***
  if $d + p_i \leqslant D$ ***then***
    Assign the task i to the machine c
    $d \leftarrow d + p_i$
  else
    Assign a time $D - d$ of the task i to the machine c
    Assign the rest of task i to the machine $c + 1$
    $c \leftarrow c + 1$
    $d \leftarrow p_i - (D - d)$
  end if
end for

(c) Each task has now a time interval $[d_i, f_i]$ in which it must be performed. Write a linear system of constraints whose feasibility is equivalent to the existence of a schedule that runs all the tasks in the appropriate time intervals.

Solution — We sort all the dates d_i and f_i (deleting duplicates) in a increasing sequence $t_0, \ldots, t_r$. There is a variable $x_{i,k}$ for each task i and each interval $[t_k, t_{k+1}]$. It represents the total time spent on task i in this time interval. The constraints are the following. Firstly, the task i must only be run in the interval $[d_i, f_i]$:

$$x_{i,k} = 0 \text{ if } [t_k, t_{k+1}] \not\subseteq [d_i, f_i].$$

Secondly, the tasks must be completed, so for all i

$$\sum_{k=0}^{r-1} x_{i,k} = p_i.$$

Thirdly,

$$x_{i,k} \leqslant t_{k+1} - t_k.$$

And lasly, there are only m computers, so

$$\sum_{i=1}^{n} x_{i,k} \leqslant m(t_{k+1} - t_k).$$

Question (b) (applied to each interval $[t_k, t_{k+1}]$) ensures that any solution of this linear program can be realized by a sound schedule.

(d) Write an algorithm that computes a schedule obeying the time constraints and minimizing the termination time.

Solution — In the previous linear program, we replace the data t_r by a variable T (representing the termination time) and we minimize T. If the minimal value is t_{r-1}, we remove the last time interval and repeat the process. Once the best solution is found, we run the algorithm of Question (b) on each interval $[t_k, t_{k+1}]$ to compute the actual schedule.

Exercice 4 (Min-cut and Ising model)**.** Let $G = (V, E)$ be a nondirected weighted graph with positive weights J_{ij}. Each vertex i has an associated scalar h_i. We want to solve the following optimization problem:

$$\text{maximize} \sum_{(i,j)\in E} J_{ij}\sigma_i\sigma_j + \sum_{i\in V} h_i\sigma_i$$
$$\text{such that } \sigma_i \in \{-1, 1\}\,.$$

Show that an optimal solution can be computed in polynomial time. (Hint: first assume that $h_i = 0$ and use a reduction to Min-cut; then refine the reduction to take h_i into account.)

Solution — We recall the MIN-CUT *problem for an undirected graph:*

Let $G = (V, E)$ be an undirected graph with weight $J_{i,j}$ for $i, j \in E$. A cut $\sigma = (S_1, S_2)$ of G is a partition of V in two sets S_1, S_2. The size of σ is the total weight of edges that connect S_1 to S_2:

$$C(\sigma) = \sum_{\substack{(i,j)\in E\\ i\in S_1\\ j\in S_2}} J_{i,j}.$$

The problem MIN-CUT *is now the following:*

Input: *An undirected graph $G = (V, E)$ and a weight function J on the edges.*
Problem *Find a cut σ of minimal size.*

Remark - *When all the weights are* positive*, this problem can be solved in polynomial time. The algorithm works as follows:*

One builds a weighted directed graph $G' = (V', E')$ where $V' = V$ and each edge $\{i, j\} \in E$ yields two reversed edges of same weight $J_{i,j}$: (i, j) and (j, i).

Then, given two vertices s and t, the max-flow/min-cut algorithm theorem asserts that the maximum value of a flow from s to t is the minimum value of an $s-t$ cut of G' and standard flow algorithms can find this corresponding cut easily. Now, in order to solve the MIN-CUT *problem, one just fixes one arbitrarily vertex v and compute the minimal $v - x$ cut for all $n - 1$ other vertex x. The cut which is minimal is a minimal size cut of the original undirected graph.*

Here, it is necessary *that the weights are* positive*.*

Let's proceed to the reduction:

- ***Step 1:*** $\forall i, h_i = 0$
  To a cut $\sigma = (S_1, S_2)$ we associate a function

$$\sigma : \begin{cases} V & \to & \{\pm 1\} \\ i & \mapsto & \sigma_i \doteq \begin{cases} 1 & \text{if } i \in S_1 \\ -1 & \text{if } i \in S_2 \end{cases} \end{cases}$$

  Then, using the equality $\mathbb{1}_{i\in S_1}\mathbb{1}_{j\in S_2} = \dfrac{1 - \sigma_i\sigma_j}{2}$, we have:

$$
\begin{aligned}
C(\sigma) &= \sum_{\{i,j\}\in E} J_{i,j} \mathbb{1}_{i\in S_1} \mathbb{1}_{j\in S_2} \\
&= \sum_{\{i,j\}\in E} J_{i,j} \left(\frac{1-\sigma_i\sigma_j}{2}\right) \\
&= \frac{1}{2} \sum_{(i,j)\in E} J_{i,j} - \frac{1}{2} \sum_{(i,j)\in E} J_{i,j}\sigma_i\sigma_j.
\end{aligned}
$$

i.e.

$$
\sum_{(i,j)\in E} J_{i,j}\sigma_i\sigma_j = \sum_{(i,j)\in E} J_{i,j} - 2 \times C(\sigma)
$$

Since the sum $\sum_{(i,j)\in E} J_{i,j}$ *is constant, maximizing the left hand side is equivalent to minimizing* $C(\sigma)$, i.e. *to solving* MIN-CUT.

- ***Step 2:*** $h_i \neq 0$

  We modify the graph G' *in the previous reduction by adding two new vertices* s *and* t *with the following new edges:*

  - *For each vertex* $i \in V$ *such that* $h_i > 0$, *we add an edge* (s, i) *with weight* $2h_i$,
  - *For each vertex* $i \in V$ *such that* $h_i < 0$, *we add an edge* (s, i) *with weight* $-2h_i$ *(Recall that we need positive weights).*

  Notice that any cut $\sigma = (S_1, S_2)$ *of this new graph* G'' *satisfies that* s *and* t *are not in the same part. Without loss of generality, we can assume that* $s \in S_1$ *and* $t \in S_2$. *Now, let* σ *be a cut of* G''.

$$
C(\sigma) = \sum_{(i,j)\in E} J_{i,j} \mathbb{1}_{i\in S_1} \mathbb{1}_{j\in S_2} + \sum_{i\in V} \frac{h_i}{2} \mathbb{1}_{i\in S_2} + \sum_{i\in V} \frac{-h_i}{2} \mathbb{1}_{i\in S_1}
$$

Notice that $\mathbb{1}_{i\in S_1} = \dfrac{1+\sigma_i}{2}$ *and* $\mathbb{1}_{i\in S_2} = \dfrac{1-\sigma_i}{2}$. *Hence,*

$$
\begin{aligned}
C(\sigma) &= \frac{1}{2} \sum_{(i,j)\in E} J_{i,j} - \frac{1}{2} \sum_{(i,j)\in E} J_{i,j}\sigma_i\sigma_j + \sum_{i\in V} \frac{h_i}{2} \left(\frac{1-\sigma_i}{2} - \frac{1+\sigma_i}{2}\right) \\
&= \frac{1}{2} \sum_{(i,j)\in E} J_{i,j} - \frac{1}{2} \sum_{(i,j)\in E} J_{i,j}\sigma_i\sigma_j - \frac{1}{2} \sum_{i\in V} h_i\sigma_i
\end{aligned}
$$

i.e.

$$
\sum_{(i,j)\in E} J_{i,j}\sigma_i\sigma_j + \sum_{i\in V} h_i\sigma_i = \sum_{(i,j)\in E} J_{i,j} - 2 \times C(\sigma)
$$

and maximizing the left-hand side is equivalent to solving MIN-CUT.