Algorithme du simplexe

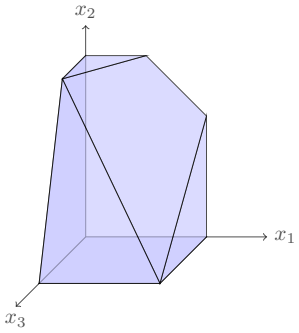
François Schwarzentruber

3 mai 2021

Algorithme pour la programmation linéaire **réelle** qui se balade de sommet en sommet

Exemple 1
Il vend des chocolats simples (1 euro), des pyramides (6 euros) et des pyramides de luxe (13 euros).
Au maximum, il peut vendre 200 chocolats simples, 300 pyramides, pas plus de 400 chocolats en tout et le nombre de pyramides plus trois fois le nombre de pyramides de luxe est au plus 600.

$$\text{maximiser } x_1 + 6x_2 + 13x_3$$
$$\begin{cases} x_1 \leqslant 200 \\ x_2 \leqslant 300 \\ x_1 + x_2 + x_3 \leqslant 400 \\ x_2 + 3x_3 \leqslant 600 \\ x_1, x_2, x_3 \geq 0 \\ x_1, x_2, x_3 \in \mathbb{R} \end{cases}$$



1 Formes normales pour les programmes linéaires

1.1 Programme canonique

Définition 2 (programme canonique) Un programme canonique est de la forme :

$$\text{maximiser } c^t x$$
$$\begin{cases} Ax \leqslant b \\ x \geqslant 0 \end{cases}$$

où $c \in \mathbb{R}^n$, $A \in \mathfrak{M}_{m,n}(\mathbb{R})$ et $b \in \mathbb{R}^m$.

n = nombre de variables
m = nombre d'inégalités linéaires

Exemple 3

$$\text{maximiser } x_1 + 6x_2 + 13x_3$$
$$\begin{cases} x_1 \leqslant 200 \\ x_2 \leqslant 300 \\ x_1 + x_2 + x_3 \leqslant 400 \\ x_2 + 3x_3 \leqslant 600 \end{cases}$$

$$\text{maximiser } \begin{pmatrix} 1 & 6 & 13 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$
$$\begin{cases} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leqslant \begin{pmatrix} 200 \\ 300 \\ 400 \\ 600 \end{pmatrix} \end{cases}$$

Proposition 4 Tout programme linéaire peut s'écrire sous la forme d'un programme canonique équivalent.

Démonstration.

1. Passer de maximiser à minimiser (ou vice et versa) : multiplier la fonction objectif par -1.
2. Passer d'égalités à des inégalités : $a^T x = b$ devient $a^T x \leq b$ et $a^T x \geq b$
3. N'avoir que des variables positives : Remplacer x par $x^+ - x^-$ et ajouter $x^+, x^- \geq 0$.

■

1.2 Programme équationnel

Définition 5 (programme équationnel) Un programme linéaire équationnel est de la forme

$$\begin{array}{l}\text{maximiser } c^t x \\ \begin{cases} Ax = b \\ x \geqslant 0 \end{cases}\end{array}$$

où $c \in \mathbb{R}^n$, $A \in \mathfrak{M}_{m,n}(\mathbb{R})$ et $b \in \mathbb{R}^m$.

Exemple 6 Tout programme canonique se réécrit en un programme équationnel équivalent faisant introduire des **variables d'écart** x_4, x_5, x_6, x_7 de la façon suivante :

$$\begin{array}{l}\text{maximiser } x_1 + 6x_2 + 13x_3 \\ \begin{cases} x_1 + x_4 & = 200 \\ x_2 + x_5 & = 300 \\ x_1 + x_2 + x_3 + x_6 & = 400 \\ x_2 + 3x_3 + x_7 & = 600 \end{cases}\end{array}$$

$$\begin{array}{l}\text{maximiser} \\ \begin{pmatrix} 1 & 6 & 13 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \\ \begin{cases} \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 3 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} 200 \\ 300 \\ 400 \\ 600 \end{pmatrix} \\ \end{cases}\end{array}$$

2 Représentation d'un sommet du polyèdre

Dans cette section, on considère un programme équationnel

$$\begin{array}{l}\text{maximiser } c^t x \\ \begin{cases} Ax = b \\ x \geqslant 0 \end{cases}\end{array}$$

où $c \in \mathbb{R}^n$, $A \in \mathfrak{M}_{m,n}(\mathbb{R})$ et $b \in \mathbb{R}^m$, et où $\underline{A \text{ est de rang } m}$.

2.1 Bases

Exemple 7 L'exemple 6 convient : la matrice A contient la matrice identité Id_m donc est de rang $m = 4$.

Notation 8 Pour un sous-ensemble B de $\{1, \ldots, n\}$, on note A_B la matrice obtenue en sélectionnant les colonnes d'indices dans B.

Définition 9 Une base B est un sous-ensemble de $\{1, \ldots, n\}$ de cardinal m telle que A_B soit de rang m.

Exemple 10 $\{4, 5, 6, 7\}$ est une base car $A_B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ est de rang $m = 4$.

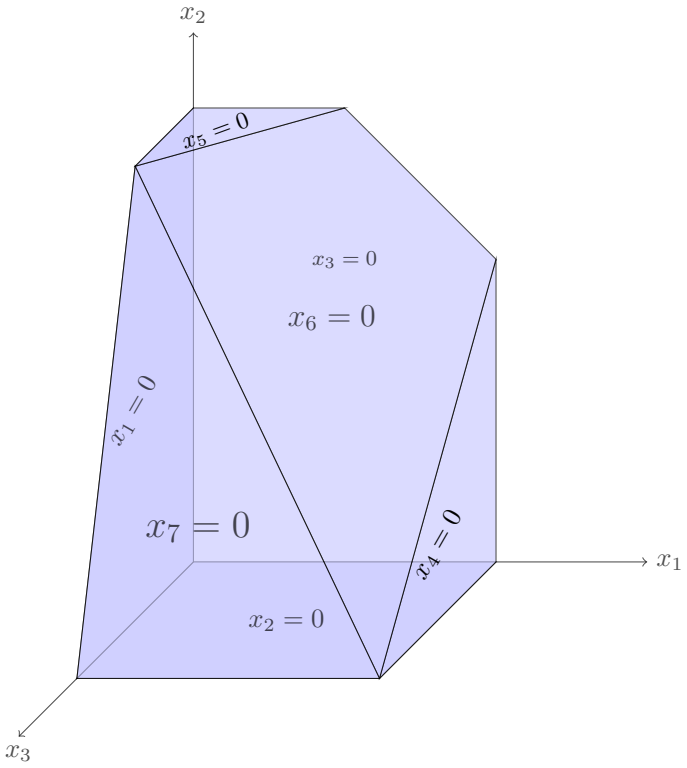
Définition 11 (variables de base) Les variables dans x_B s'appelle les variables de base.

Exemple 12 Dans l'exemple 6, $B = \{4, 5, 6, 7\}$. Les variables de base sont les variables d'écart x_4, x_5, x_6, x_7.

n	=	nombre de faces
$x_1 = 0, \ldots, x_n = 0$	=	les n faces
$n - m$	=	nombre de dimensions dans le problème original
$x_1, \ldots, x_{n-m} = 0$	=	les $n - m$ faces correspondantes aux axes
valeur de x_i	=	'distance' à la face $x_i = 0$
sommet	$\sim$	base (les variables hors-base x_i sont nulles)

maximiser $x_1 + 6x_2 + 13x_3$

$$\begin{cases} x_1 + x_4 = 200 \\ x_2 + x_5 = 300 \\ x_1 + x_2 + x_3 + x_6 = 400 \\ x_2 + 3x_3 + x_7 = 600 \end{cases}$$



Notation 13 On note $N = \{1, \ldots, n\} \setminus B$.

Définition 14 (variables hors base) Les variables dans x_N sont hors base. (et sont vues comme nulles.)

2.2 Tableaux

Définition 15 (tableau) Étant donné une base B, un tableau est un programme linéaire sous la forme

$$\text{maximiser } v + c'^t x_N \quad \begin{cases} x_B = p - A' x_N \\ x \geqslant 0 \end{cases}$$

où $v \in \mathbb{R}$, $c' \in \mathbb{R}^{n-m}$, $p \in \mathbb{R}^m$, $A' \in \mathfrak{M}_{m,n-m}(\mathbb{R})$ et le vecteur $x \in \mathbb{R}^n$ est décomposé en $x_B \in \mathbb{R}^m$ et $x_N \in \mathbb{R}^{n-m}$.

Exemple 16 (tableau) On transforme un programme équationnel en tableau en prenant les variables d'écart pour x_B :

$$\text{maximiser } 0 + x_1 + 6x_2 + 13x_3 \quad \begin{cases} x_4 = 200 - x_1 \\ x_5 = 300 - x_2 \\ x_6 = 400 - x_1 - x_2 - x_3 \\ x_7 = 600 - x_2 - 3x_3 \end{cases}$$

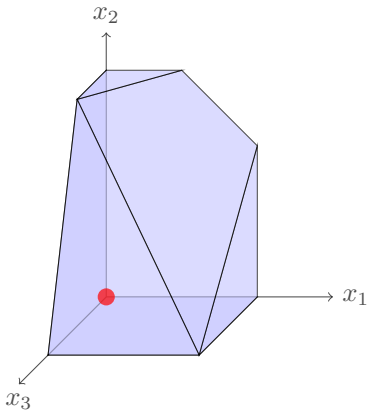
variables hors bases

variables bases

;

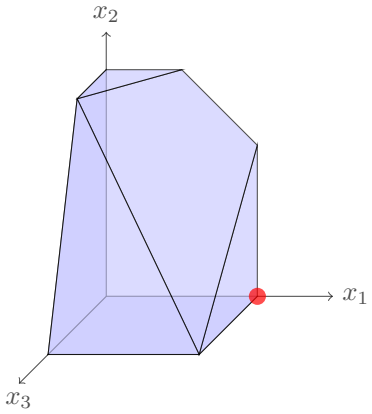
Exemple 17 Le point $(x_1, x_2, x_3) = (0, 0, 0)$ est l'intersection des plans $x_1 = 0$, $x_2 = 0$ et $x_3 = 0$. Il est représenté par le tableau avec la base $B = \{4, 5, 6, 7\}$:

$$\text{maximiser } x_1 + 6x_2 + 13x_3 \quad \begin{cases} x_4 &= 200 - x_1 \\ x_5 &= 300 - x_2 \\ x_6 &= 400 - x_1 - x_2 - x_3 \\ x_7 &= 600 - x_2 - 3x_3 \end{cases}$$



Exemple 18 Le point $(x_1, x_2, x_3) = (200, 0, 0)$ est l'intersection des plans $x_4 = 0$, $x_2 = 0$ et $x_3 = 0$. Il est représenté par le tableau avec la base $B = \{1, 5, 6, 7\}$:

$$\text{maximiser } 200 - x_4 + 6x_2 + 13x_3 \quad \begin{cases} x_1 &= 200 - x_4 \\ x_5 &= 300 - x_2 \\ x_6 &= 200 + x_4 - x_2 - x_3 \\ x_7 &= 600 - x_2 - 3x_3 \end{cases}$$



Définition 19 (solution basique) Un tableau

$$\text{maximiser } v + c'^t x_N \quad \begin{cases} x_B = p + A' x_N \\ x \geqslant 0 \end{cases}$$

admet une solution basique si $p \geq 0$. La solution basique est le vecteur $x \in \mathbb{R}^n$ défini par $x_B = p$ et $x_N = 0$.

Proposition 20 Étant donné une base B, il existe un unique tableau équivalent au programme équationnel. Il s'agit de :

$$\text{maximiser } v + c'^t x_N \quad \begin{cases} x_B = p - A' x_N \\ x \geqslant 0 \end{cases}$$

avec $v = c_B^t A_B^{-1} b$; $c' = c_N - (c_B^t A_B^{-1} A_N)^t$; $p = A_B^{-1} b$; et $A' = A_B^{-1} A_N$.

Démonstration. Le système de contraintes $Ax = b$ s'écrit $A_N x_N + A_B x_B = b$. D'où $x_B = A_B^{-1} b - A_B^{-1} A_N x_N$. L'objectif $c^t x$ s'écrit $c_B^t x_B + c_N^t x_N = c_B^t (A_B^{-1} b - A_B^{-1} A_N x_N) + c_N^t x_N = c_B^t (A_B^{-1} b + (c_N - (c_B^t A_B^{-1} A_N)^t) x_N$. ■

Exemple 21 Considérons le programme équationnel $\text{maximiser } x_1 + 6x_2 + 13x_3$ $\begin{cases} x_1 + x_4 &= 200 \\ x_2 + x_5 &= 300 \\ x_1 + x_2 + x_3 + x_6 &= 400 \\ x_2 + 3x_3 + x_7 &= 600 \end{cases}$

Pour la base $B = \{4, 5, 6, 7\}$, on a $A_B = Id_4$, $c_B^t = (0, 0, 0, 0)$ et $c_N^t = (1, 6, 13)$.

Pour la base $B = \{1, 5, 6, 7\}$, on a $A_B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $c_B = (1, 0, 0, 0)$ et $c_N = (6, 13, 0)$.

3 Algorithme du simplexe

3.1 Principe

fonction simplexe
    sommet = $(0, \dots, 0)$
    tant que sommet non optimal
        trouver un sommet voisin qui améliore l'objectif
    renvoyer sommet optimal

entrée : un tableau τ admettant une solution basique
sortie : le maximum ou alors lève une exception "non borné"
fonction simplexe(τ)
    tant que il y a un coefficient strictement positif dans la fonction objectif de τ
        $\tau = \text{pivot}(\tau)$
    renvoyer la constante dans la fonction objectif de τ

Théorème 22 L'algorithme du simplexe, s'il termine, retourne bien la valeur maximale de l'objectif.

Démonstration.

$$\begin{array}{l}\text{maximiser } v + c'^t x_N \\ \begin{cases} x_B = p - A' x_N \\ x \geqslant 0 \end{cases}\end{array}$$

L'algorithme termine donc les coefficients de c' sont tous négatifs. Sur l'espace des solutions, la valeur v majore l'objectif. La valeur v est atteinte pour la solution basique, qui est donc solution optimale. ■

3.2 Pivot

entrée : un tableau τ admettant une solution basique
sortie : un tableau équivalent admettant aussi une solution basique, et améliorant l'objectif, de même espace de solution, et de même valeur d'objectif ou alors lève une exception "non borné"
fonction pivot(τ)
    renvoyer un nouveau tableau obtenu à partir de τ comme suit.
    1. repérer une variable x_e avec coefficient strictement positif dans la fonction objectif
    2. repérer la contrainte $x_s = \dots$ qui limite le plus la croissance de x_e
       s'il n'y a pas de telle contrainte **lever une exception** "non borné"
    3. augmenter x_e jusqu'à ce que $x_s := 0$ $\quad x_e := 0 \quad x_s \nearrow$

fonction *pivot* $\left(\begin{array}{l}\textbf{maximiser } v + c^t x_N \\ \begin{cases} x_B = b - A x_N \\ x \geq 0 \end{cases}\end{array} \right)$

choisir $j_0 \in N$ tel que $c_{j_0} > 0$
choisir $i_0 \in B$ tel que $-a_{i_0,j_0} < 0$ et $\frac{b_{i_0}}{a_{i_0,j_0}}$ minimal, s'il n'y a pas, **lever une exception** "non borné"

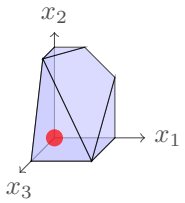
renvoyer $\begin{array}{l}\textbf{maximiser } v' +^t c' x_{N'} \\ \begin{cases} x_{B'} = b' - A' x_{N'} \\ x \geq 0 \end{cases}\end{array}$ où pour tout $i \in B$, $j \in N$:

- $N' = \{i_0\} \cup N \setminus \{j\}$;
- $B' = \{j_0\} \cup B \setminus \{i\}$;
- $v' = v + \frac{b_{i_0} c_{j_0}}{a_{i_0,j_0}}$
- $c'_j = c_j - c_{j_0} \frac{a_{i_0,j}}{a_{i_0,j_0}}$
- $c'_{i_0} = \frac{-c_{j_0}}{a_{i_0,j_0}}$
- $b'_i = b_i - a_{i,j_0} \frac{b_{i_0}}{a_{i_0,j_0}}$
- $b'_{j_0} = \frac{b_{i_0}}{a_{i_0,j_0}}$
- $a'_{i,j} = a_{i,j} - \frac{a_{i,j_0} a_{i_0,j}}{a_{i_0,j_0}}$
- $a'_{i,i_0} = \frac{a_{i,j_0}}{a_{i_0,j}}$
- $a'_{j_0,i_0} = \frac{1}{a_{i_0,j_0}}$
- $a'_{j_0,j} = \frac{-a_{i_0,j}}{a_{i_0,j_0}}$.

Proposition 23 (admis car lourd) Le pivotage produit un tableau équivalent.

Proposition 24 (admis car lourd) Le pivotage augmente la valeur de l'objectif (pas forcément strictement).

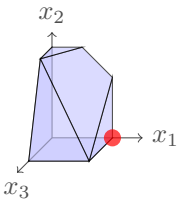
3.3 Exemple d'exécution



maximiser $x_1 + 6x_2 + 13x_3$

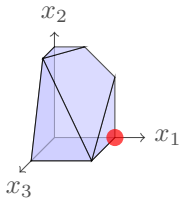
$$\begin{cases} x_4 &= 200 - x_1 \\ x_5 &= 300 - x_2 \\ x_6 &= 400 - x_1 - x_2 - x_3 \\ x_7 &= 600 - x_2 - 3x_3 \end{cases}$$

$x_4 := 0 \qquad x_1 \nearrow$



maximiser $200 - x_4 + 6x_2 + 13x_3$

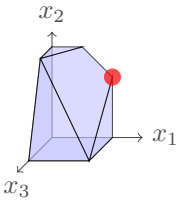
$$\begin{cases} x_1 &= 200 - x_4 \\ x_5 &= 300 - x_2 \\ x_6 &= 200 + x_4 - x_2 - x_3 \\ x_7 &= 600 - x_2 - 3x_3 \end{cases}$$



maximiser $200 - x_4 + 6x_2 + 13x_3$

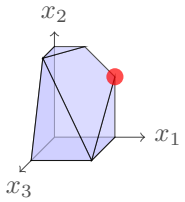
$$\begin{cases} x_1 &= 200 - x_4 \\ x_5 &= 300 - x_2 \\ x_6 &= 200 + x_4 - x_2 - x_3 \\ x_7 &= 600 - x_2 - 3x_3 \end{cases}$$

$x_6 := 0 \qquad x_2 \nearrow$



maximiser $1400 + 5x_4 - 6x_6 + 7x_3$

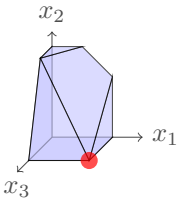
$$\begin{cases} x_1 &= 200 - x_4 \\ x_5 &= 100 - x_4 + x_6 + x_3 \\ x_2 &= 200 + x_4 - x_6 - x_3 \\ x_7 &= 400 - x_4 + x_6 - 2x_3 \end{cases}$$



maximiser $1400 + 5x_4 - 6x_6 + 7x_3$

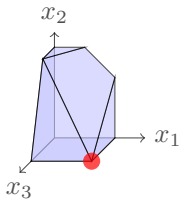
$$\begin{cases} x_1 &= 200 - x_4 \\ x_5 &= 100 - x_4 + x_6 + x_3 \\ x_2 &= 200 + x_4 - x_6 - x_3 \\ x_7 &= 400 - x_4 + x_6 - 2x_3 \end{cases}$$

$x_2 := 0 \qquad x_3 \nearrow$



maximiser $2800 + 12x_4 - 13x_6 - 7x_2$

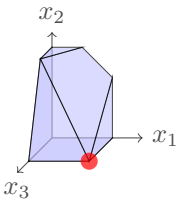
$$\begin{cases} x_1 &= 200 - x_4 \\ x_5 &= 300 - x_2 \\ x_3 &= 200 + x_4 - x_6 - x_2 \\ x_7 &= -3x_4 + 3x_6 + 2x_2 \end{cases}$$



maximiser $2800 + 12x_4 - 13x_6 - 7x_2$

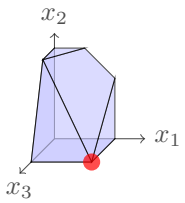
$$\begin{cases} x_1 &= 200 - x_4 \\ x_5 &= 300 - x_2 \\ x_3 &= 200 + x_4 - x_6 - x_2 \\ x_7 &= -3x_4 + 3x_6 + 2x_2 \end{cases}$$

$x_7 := 0 \qquad x_4 \nearrow$



maximiser $2800 - x_6 - 4x_7 + x_2$

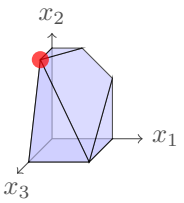
$$\begin{cases} x_1 &= 200 - x_6 + \frac{x_7}{3} - \frac{2x_2}{3} \\ x_5 &= 300 - x_2 \\ x_3 &= 200 - \frac{x_7}{3} - \frac{x_2}{3} \\ x_4 &= x_6 - \frac{x_7}{3} + \frac{2x_2}{3} \end{cases}$$



maximiser $2800 - x_6 - 4x_7 + x_2$

$$\begin{cases} x_1 &= 200 - x_6 + \frac{x_7}{3} - \frac{2x_2}{3} \\ x_5 &= 300 - x_2 \\ x_3 &= 200 - \frac{x_7}{3} - \frac{x_2}{3} \\ x_4 &= x_6 - \frac{x_7}{3} + \frac{2x_2}{3} \end{cases}$$

$x_5 := 0 \qquad x_2 \nearrow$



maximiser $3100 - 4x_7 - x_6 - x_5$

$$\begin{cases} x_1 &= 0 + \ldots \\ x_2 &= 300 - \epsilon_2 \\ x_3 &= 100 + \ldots \\ \epsilon_1 &= \ldots \end{cases}$$

4 Prétraitement : obtenir tableau avec solution basique

entrée : un programme canonique
sortie : "pas de solutions" ou un tableau avec solution basique de même maximum
fonction prétraitement(L)

1. construire un programme auxiliaire L_{aux} en introduisant une variable auxiliaire x_0

   //L_{aux} admet des solutions
   //l'espace des solutions de L est non vide ssi l'optimum de L_{aux} est 0.

2. mettre L_{aux} sous forme équationnelle, puis sous tableau
3. faire un pivot en faisant entrer x_0 dans la base, et en faisant sortir la variable de base dont la valeur est la plus négative

   //le tableau courant admet une solution basique

4. Lancer l'algorithme du simplexe sur ce tableau

si le maximum est 0 **alors**
  5. réécrire la fonction objectif de L avec les variables hors bases et en enlever x_0
  renvoyer le tableau courant, mais en 5.
sinon
  renvoyer "pas de solutions"

Exemple 25 Le programme canonique suivant n'admet pas $(0, 0)$ comme solution :

$$\text{maximiser } 2x_1 - x_2 \quad \begin{cases} 2x_1 - x_2 \leq 2 \\ x_1 - 5x_2 \leq -4 \\ x_1, x_2 \geq 0 \end{cases}$$

1. Programme auxiliaire.

$$\text{maximiser } -x_0 \quad \begin{cases} 2x_1 - x_2 - x_0 \leq 2 \\ x_1 - 5x_2 - x_0 \leq -4 \\ x_1, x_2, x_0 \geq 0 \end{cases}$$

2. Tableau du programme auxiliaire

$$\text{maximiser } -x_0 \quad \begin{cases} x_3 = 2 - 2x_1 + x_2 + x_0 \\ x_4 = -4 - x_1 + 5x_2 + x_0 \\ x_1, x_2, x_0, x_3, x_4 \geq 0 \end{cases}$$

3. Pivot. La variable de valeur la plus petite (-4) est x_4.

$$\text{maximiser } -x_0 \quad \begin{cases} x_3 = 2 - 2x_1 + x_2 + x_0 \\ x_4 = -4 - x_1 + 5x_2 + x_0 \\ x_1, x_2, x_0, x_3, x_4 \geq 0 \end{cases} \xrightarrow{x_4 := 0 \quad x_0 \nearrow} \text{maximiser } -4 - x_1 + 5x_2 + x_4 \quad \begin{cases} x_3 = 6 - x_1 - 4x_2 + x_4 \\ x_0 = 4 + x_1 - 5x_2 - x_4 \\ x_1, x_2, x_0, x_3, x_4 \geq 0 \end{cases}$$

4. Exécution de l'algorithme du simplexe. Le tableau courant admet une solution basique. On lance l'algo du simplexe pour voir si le max = 0. Si ce n'est pas le cas, on répond 'l'espace des solutions de L est vide'.

$$\text{maximiser } -4 - x_1 + 5x_2 + x_4 \quad \begin{cases} x_3 = 6 - x_1 - 4x_2 + x_4 \\ x_0 = 4 + x_1 - 5x_2 - x_4 \\ x_1, x_2, x_0, x_3, x_4 \geq 0 \end{cases} \xrightarrow{x_0 := 0 \quad x_3 \nearrow} \text{maximiser } -x_0 \quad \begin{cases} x_3 = \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5} \\ x_2 = \frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\ x_1, x_2, x_0, x_3, x_4 \geq 0 \end{cases}$$

L'algo du simplexe termine et le max vaut bien 0. L'espace des solution de L est non vide.

5. Réécriture de la fonction objectif. On remet l'objectif initial, que l'on réécrit avec les variables "nulles" (hors base), puis on enlève x_0 qui est nul :

$$\text{maximiser } 2x_1 - x_2 \quad \begin{cases} x_3 = \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5} \\ x_2 = \frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\ x_1, x_2, x_0, x_3, x_4 \geq 0 \end{cases} \qquad \text{maximiser } -\frac{4}{5} + \frac{x_0}{5} + \frac{9x_1}{5} - \frac{x_4}{5} \quad \begin{cases} x_3 = \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5} \\ x_2 = \frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\ x_1, x_2, x_0, x_3, x_4 \geq 0 \end{cases}$$

$$\text{maximiser } -\frac{4}{5} + \frac{9x_1}{5} - \frac{x_4}{5} \quad \begin{cases} x_3 = \frac{14}{5} - \frac{9x_1}{5} + \frac{x_4}{5} \\ x_2 = \frac{4}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

5 Terminaison de l'algorithme du simplexe

Proposition 26 L'algorithme du simplexe peut boucler.

DÉMONSTRATION. Voir chapitre 9 de [CC+83]. ■

Proposition 27 Si la valeur augmente strictement, alors l'algo du simplexe termine en au plus $\binom{m}{n}$ itérations.

DÉMONSTRATION. Car il y a au plus $\binom{m}{n}$ bases possibles. ■

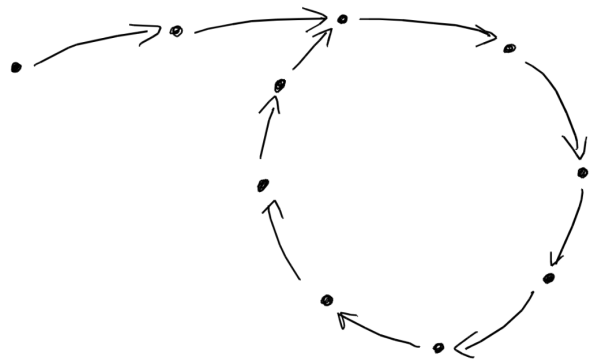
Définition 28 (règle de Bland) La règle de Bland est la stratégie consistant à choisir la variable entrante candidate d'indice minimal, et la variable sortante candidate d'indice minimal.

Proposition 29 (terminaison) En appliquant la règle de Bland, l'algorithme du simplexe termine.

DÉMONSTRATION. Par l'absurde. Supposons que l'algorithme ne termine pas sur le programme équationnelle suivant :

$$\begin{array}{l}\text{maximiser } c^t x \\ \begin{cases} Ax = b \\ x \geq 0 \end{cases}\end{array}$$

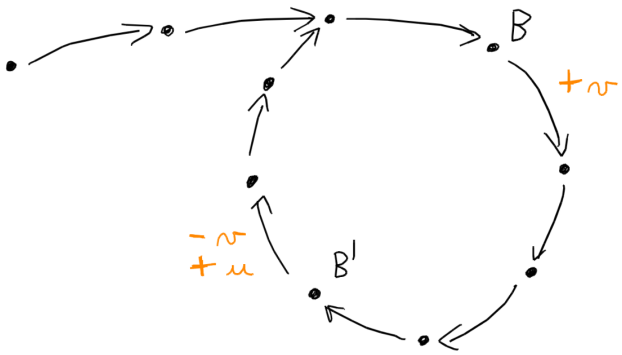
Nous avons donc un cycle. Dans le dessin suivant, chaque point représente une base (et le tableau associé) et une flèche représente un pivotage.



Soit F les indices de variables qui entrent (et aussi sortent, puisqu'on retombe sur la même base !) durant le cycle. Les variables correspondantes sont dites "folles" (mais appelées aussi "capricieuses" dans la littérature).

Fait 30 Les variables "folles" sont nulles dans les solutions basiques du cycle (sinon l'objectif augmenterait !).

Soit v l'indice de F le le plus grand. On considère maintenant une base B du cycle où le pivotage qui suit fait entrer v dans la base. On considère aussi une base B' du cycle où le pivotage qui suit fait sortir v, et qui fait entrer une certaine variable d'indice que l'on note u.



Afin de ne pas alourdir la démonstration, nous ferons des abus de langage. On dira : "en B" pour dire "dans le tableau correspondant à la base B", "dans l'objectif de B" pour dire "dans la fonction objectif écrite comme $z_0 + cx_N$ dans le tableau de la base B, où $N = \{1, \ldots, n\} \setminus B$", "les variables de F" pour dire "les variables dont les indices sont dans F".

En B, il n'y a pas d'autres variables de F candidates à entrer. En effet, s'il y en avait une autre dans F, son indice aurait été plus petit que v. Ainsi, v n'aurait pas été sélectionnée en B car la règle de Bland stipule de choisir celle d'indice minimal. Ainsi :

Fait 31 Dans l'objectif de B, les coefficients devant les variables de F sont ≤ 0 sauf pour v.

Par un raisonnement similaire, en B', pour ce u qui entre, v est le seul candidat sortant dans F (s'il y en avait eu une autre, il aurait été d'indice plus petit et aurait donc été sélectionné, blablabla...). On rappelle que les valeurs des variables dans F sont nulles. Ainsi i candidat sortant ssi $q'_{iu} < 0$, où q'_{iu} est l'élément en ligne i, colonne u dans la matrice du système d'équations du tableau de B' (afin de simuler une "croissance de 0" de la variable u et "mettre à 0" la variable i). Pour résumer :

Fait 32 En B', $q'_{vu} < 0$ et $q'_{iu} \geq 0$ pour tout $i \in F \setminus \{v\}$.

Considérons maintenant le programme linéaire auxiliaire suivant :

$$\text{maximiser } c^t x \begin{cases} Ax = b \\ x_{F\setminus\{v\}} \geq 0 \\ x_v \leq 0 \\ x_{N\setminus F} = 0 \end{cases}$$

où $N = \{1, \ldots, n\} \setminus B$.

Les deux faits suivants sont contradictoires et conclura la démonstration par l'absurde.

Fait 33 Le programme linéaire auxiliaire admet un optimum.

Démonstration. On note $\tilde{x}$ la solution basique en B. Montrons que $\tilde{x}$ est un optimum. Comme elle est basique, on a $\tilde{x}_N = 0$: c'est donc une solution du programme auxiliaire.

Maintenant regardons l'objectif comme écrit dans le tableau de base B : $z_0 + r^t x_N$. Comme il est dit que $x_{N\setminus F} = 0$, les coefficient dans r correspondant à $N \setminus F$ ne sont pas importants. Regardons maintenant les coefficients correspondants à $N \cap F$. Comme les variables dans F sont capricieuses, on a $\tilde{x}_F = 0$.

D'après le fait 31, les coefficients devant $F \setminus \{v\}$ sont négatifs et $x_{F\setminus\{v\}} \geq 0$, et le coefficient devant v est positif mais $x_v \leq 0$. Donc quelque soit la façon de changer les valeurs de $\tilde{x}$, la valeur objectif diminue. Ainsi, $\tilde{x}$ est une solution optimale du programme linéaire auxiliaire.

■

Fait 34 L'objectif du programme linéaire auxiliaire est non borné.

Démonstration. Considérons $\tilde{x}$ solution basique en B. Comme B et B' sont dans le cycle, ils ne différent que par des variables de F qui sont rentrées ou sorties par définition de F. Pareil, N et N' ne différent que par des variables de F. Par le fait , on a $\tilde{x}_{N'} = 0$ et $\tilde{x}$ est aussi solution basique en B'.

On rappelle que l'on a :

$B \xrightarrow{+v}$.

$B' \xrightarrow{+u \quad -v}$.

L'idée de la démonstration est de modifier $\tilde{x}$ en faisant croitre x_u d'une quantité, appelons $\tilde{x}(t)$ la solution de $Ax = b$ pour $x_u = t$, puis de voir que $c^t\tilde{x}(t) \xrightarrow[t\to+\infty]{} +\infty$.

Rentrons dans le vif du sujet. Définissons $\tilde{x}(t)$ par :

$$\begin{cases} \tilde{x}(t)_u = t \\ \tilde{x}(t)_i = 0 \text{ pour tout } i \in N' \setminus \{u\} \\ \tilde{x}(t)_{B'} \text{ est définie par le système d'équation en } B' : x_{B'} = p' + Q'x_{N'}. \end{cases}$$

Montrons que $\tilde{x}(t)$ est dans l'espace de solutions du programme auxiliaire. Tout d'abord, $N' \setminus F = N \setminus F$, par définition F, car seulement les variables de F se promènent. Donc $\tilde{x}(t)_{N\setminus F} = \tilde{x}(t)_{N'\setminus F} = 0$.

Pour le reste, utilisons l'équation $x_{B'} = p' + Q'x_{N'}$. en B'.

Comme v est une variable capricieuse, $\tilde{x}_v = 0$. Comme v sort, $q'_{vu} < 0$. $\tilde{x}(t)_v = \tilde{x}_v + tq'_{vu} < 0$.

Pour tout $i \in F \setminus \{v\}$, on a aussi $\tilde{x}_i = 0$ car i est capricieuse. Si $i \in F \cap N'$, alors $\tilde{x}(t)_i = 0$. Sinon, si $i \in B' \cap F \setminus \{v\}$, par le fait 32, $q'_{iu} \geq 0$. Et donc $\tilde{x}(t)_i = \tilde{x}_i + tq'_{iu} \geq 0$.

Comme u est candidate pour entrer dans la base B', le coefficient devant x_u dans la fonction objectif du tableau B' est strictement positif et donc :

$$c^t\tilde{x}(t) = z'_0 + t \times \text{ce coefficient positif} \xrightarrow[t\to+\infty]{} +\infty.$$

La fonction objectif n'est pas bornée. ■

■

Remarque 35 En pratique, la règle de Bland est lente. On préfère perturber un peu b pour supprimer les dégénérescences. ([DPV08], p. 218)

Exercices

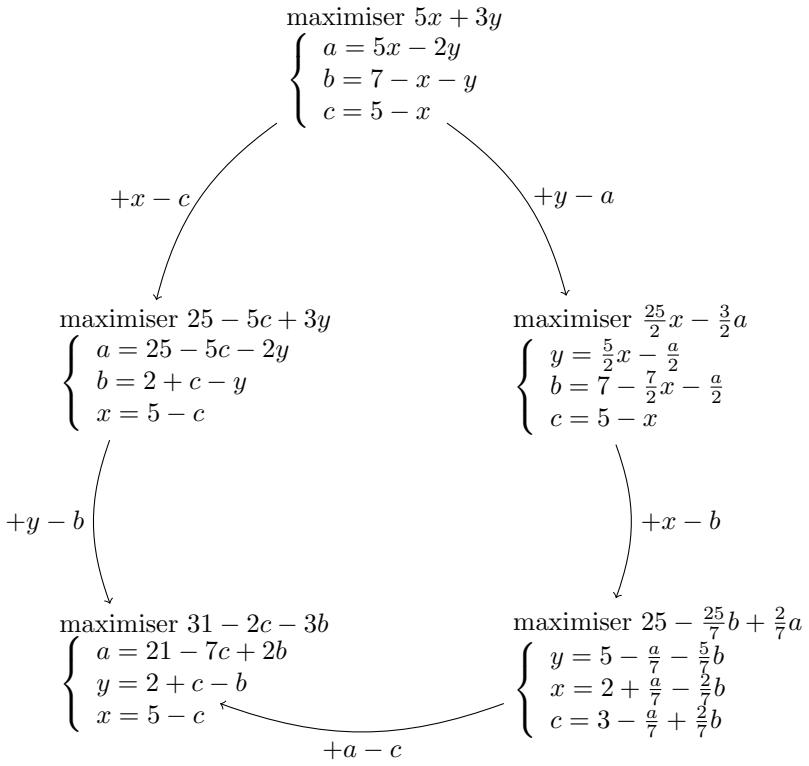
Exercice 36 Considérons le programme

$$\text{maximiser } 5x+3y \quad \begin{cases} 5x-2y \geq 0 \\ x+y \leq 7 \\ x \leq 5 \\ x,y \geq 0 \end{cases}$$

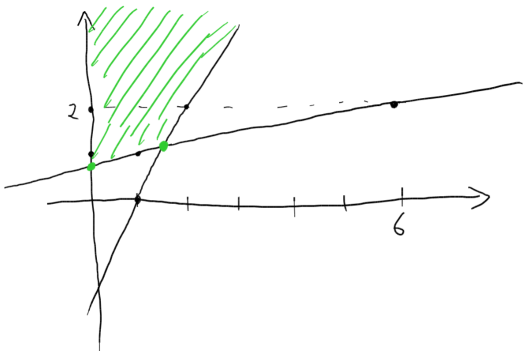
1. Dessiner l'espace des solutions.
2. Mettre sous forme équationnelle puis calculer le tableau initial.
3. Donner toutes les exécutions possibles de l'algorithme du simplexe.

Solution :

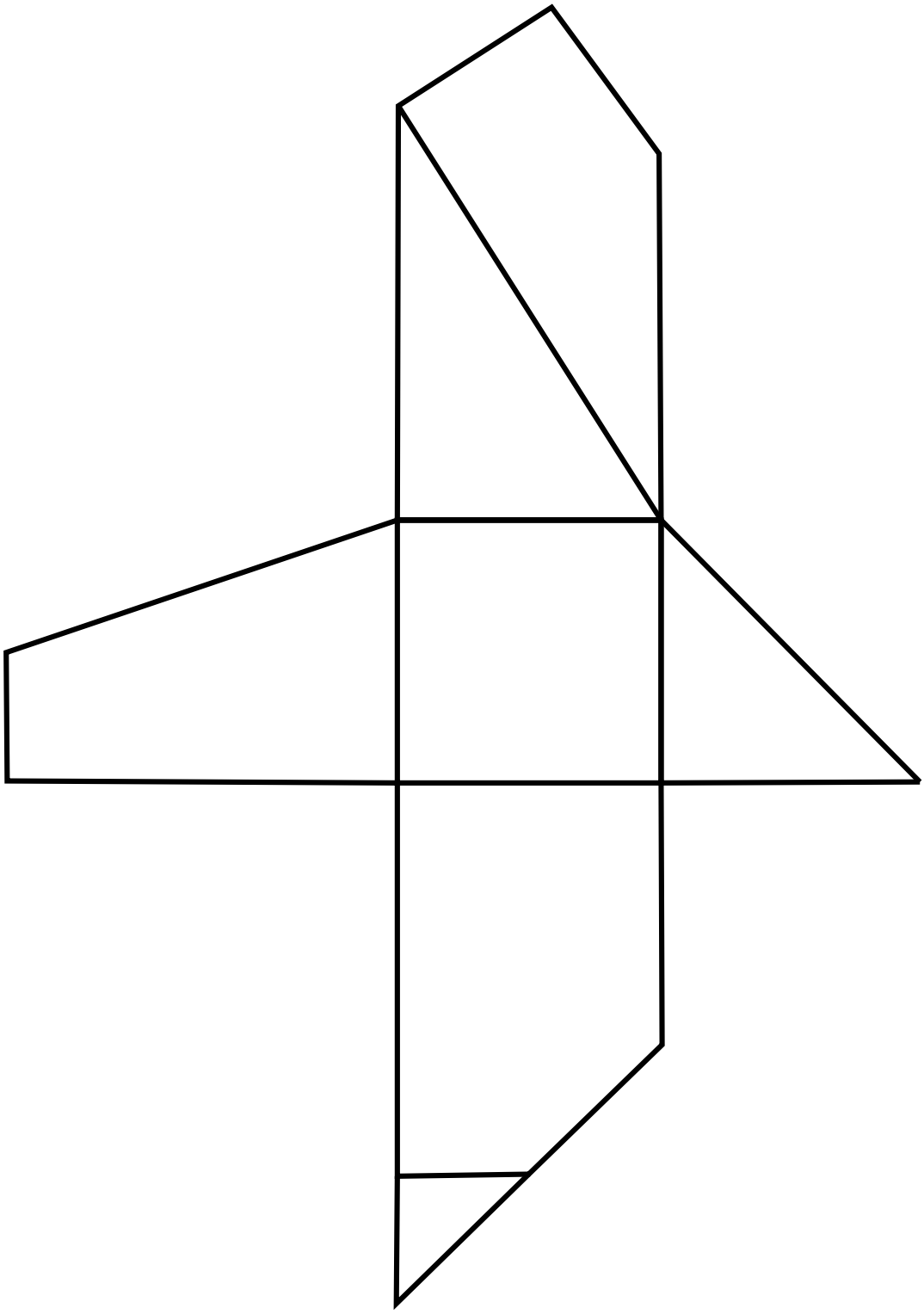
$$\text{maximiser } 5x+3y \quad \begin{cases} -5x+2y \leq 0 \\ x+y \leq 7 \\ x \leq 5 \\ x,y \geq 0 \end{cases} \qquad \text{maximiser } 5x+3y \quad \begin{cases} -5x+2y+a=0 \\ x+y+b=7 \\ x+c=5 \\ x,y,a,b,c \geq 0 \end{cases}$$



Exercice 37 Exécuter l'algorithme du simplexe sur le tableau avec solution basique obtenu dans la section 4. Expliquer ce qu'il se passe sur le dessin suivant :



Exercice 38 Construire le polyèdre exemple du cours.



Exercice 39 Donner le tableau correspondant à ce programme linéaire, puis exécuter l'algorithme du simplexe :

$$\begin{array}{l}\text{maximiser } x_1 + 2x_2 \\ \left\{\begin{array}{ll} -3x_1 + 2x_2 & \leq 2 \\ -x_1 + 2x_2 & \leq 4 \\ x_1 + x_2 & \leq 5 \\ x_1, x_2, x_3 \geq 0 & \end{array}\right.\end{array}$$

Exercice 40 Donner le tableau correspondant à ce programme linéaire, puis exécuter l'algorithme du simplexe :

$$\begin{array}{l}\text{maximiser } 5x_1 + 5x_2 + 3x_3 \\ \left\{\begin{array}{ll} x_1 + 3x_2 + x_3 & \leq 3 \\ -x_1 + 3x_3 & \leq 2 \\ 2x_1 - x_2 + 2x_3 & \leq 4 \\ 2x_1 + 3x_2 - x_3 & \leq 2 \\ x_1, x_2, x_3 \geq 0 & \end{array}\right.\end{array}$$

Exercice 41 (non borné) Donner le tableau correspondant à ce programme linéaire, puis exécuter l'algorithme du simplexe :

$$\begin{array}{l}\text{maximiser } x_1 + 3x_2 - x_3 \\ \left\{\begin{array}{ll} 2x_1 + 2x_2 - x_3 & \leq 10 \\ 3x_1 - 2x_2 + x_3 s & \leq 10 \\ x_1 - 3x_2 + x_3 & \leq 10 \\ x_1, x_2, x_3 \geq 0 & \end{array}\right.\end{array}$$

Exercice 42 (pour s'entrainer avec le prétraitement) Résoudre le programme linéaire :

$$\begin{array}{l}\text{minimiser } x_1 - x_2 + x_3 \\ \left\{\begin{array}{ll} x_1 + 3x_2 & \geq 4 \\ x_1 + x_2 - x_3 & \leq 10 \\ x_1, x_2, x_3 \geq 0 & \end{array}\right.\end{array}$$

Notes bibliographiques

L'exemple initial provient de [DPV08]. Ce livre vulgarise très bien la programmation linéaire. Malheureusement, il ne donne aucune démonstration.

Dans [CLRS09], un programme linéaire équationnel s'appelle un programme standard (mais l'adjectif standard n'a pas trop de sens). Ici, nous empruntons La terminologie de [GM07] : "programme sous forme équationnelle" que nous abrégeons en "programme équationnel". [CLRS09] donne des démonstrations mais parfois les justifications sont lourdes.

Comme mentionné dans [GM07], le plus petit exemple de programme linéaire où le simplexe boucle est donné dans Chvátal's textbook cited in Chapter 9 [CC+83].

Le prétraitement pour avoir un tableau avec solution basique vient de (cf. [CLRS09], p. 886, 29.5). J'ai fait un choix d'écrire explicitement les programmes linéaires et d'éviter d'écrire des appels cryptiques comme $pivot(N, B, A, b, c, v, \ell, 0)$, cf. [CLRS09] !

Les démonstrations dans [GM07] sont pas mal, mais parfois un peu lourdes concernant les indices dans les matrices. Ici, nous avons fait le choix d'être flexible sur les indices une matrice : les indices d'une matrice de 4 lignes peuvent être 1, 3, 5, 6, alors que dans [GM07] (p. 68) les indices sont toujours 1, 2, 3, 4 ce qui forcent d'utiliser des double-indices $\ell_1, \ell_2, \ell_3, \ell_4$.

Références

[CC+83] Vasek Chvatal, Vaclav Chvatal, et al. *Linear programming.* Macmillan, 1983.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009.

[DPV08] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh V. Vazirani. *Algorithms.* McGraw-Hill, 2008.

[GM07] Bernd Gärtner and Jirí Matousek. *Understanding and using linear programming.* Universitext. Springer, 2007.