Mémento sur la programmation linéaire

François Thomasset
INRIA

14 janvier 1998

Table des matières

1 Programmation linéaire

1.1 Le problème de la programmation linéaire

Objet de la "programmation linéaire" [Dan63] : résolution de problèmes d'optimisation où la fonction à optimiser ainsi que les contraintes sont exprimés sous forme linéaire.

Données :

A matrice m lignes x n colonnes ;
$c = (c_1, c_2, \ldots, c_n)$;
$b = (b_1, b_2, \ldots, b_m)$

$$\begin{array}{ll} \text{Min} & c \cdot x \\ & Ax = b \\ & x \in \mathbf{R}^n \\ & x \geq 0 \end{array} \qquad (1)$$

Un peu de vocabulaire :

- "solution" : vecteur x vérifiant $Ax = b$
- "solution réalisable"[1] : $Ax = b$ et $x \geq 0$
- "solution optimale" : solution réalisable satisfaisant le minimum
- "fonction coût" : $c \cdot x$; c'est la fonction dont recherche l'optimum.

1.2 Ecritures équivalentes

$$\begin{array}{ll} \text{Max} & c \cdot x \\ & Ax \leq b \\ & x \in \mathbf{R}^n \\ & x \geq 0 \end{array} \qquad (2)$$

Exercices : passer d'un problème sous la forme (1) à un problème équivalent (mêmes ensembles de solutions réalisables et optimales) sous la forme la forme (2) et réciproquement.
Mêmes questions en échangeant Min et Max.

Les problèmes (1) ou (2) sont des "programmes linéaires" (en abrégé "PL")

1.3 Solutions "de base"

Hypothèses :

La matrice A est de rang égal à m.
Problème sous la forme (1).

On réordonne les colonnes de A de sorte que $A = (B, N)$ avec :

B = matrice carrée régulière mxm
N = matrice mx$(n-m)$
$x = (x_B, x_N)$

$$Ax = b \iff Bx_B + Nx_N = b \iff x_B = B^{-1}b - B^{-1}Nx_N$$

[1] en anglais "*feasible*"

Une solution particulière : "solution de base" : obtenue en annulant x_N

$$x_B = B^{-1}b, \quad x_N = 0$$

Si en outre $x_B = B^{-1}b \geq 0$, alors (x_B, x_N) est une "solution de base réalisable"[2].

<u>Théorème</u> *Si un PL admet une solution optimale, il admet une solution de base optimale.*

A chaque solution de base correspond un sommet du polyèdre défini par les contraintes :

$$Ax = b, \; x \geq 0$$

1.4 Conditions d'optimalité

Soit $x = (x_B, 0)$ une solution de base ; on décompose de même $c = (c_B, c_N)$

$$c \cdot x = (c_B \cdot x_B + c_N \cdot x_N) = c_B \cdot x_B$$

<u>Théorème</u>
Si $c_N - c_B \cdot B^{-1}N \geq 0$, alors $x = (x_B, 0)$ est une solution optimale.
<u>Démonstration</u> Soit $y = (y_B, y_N)$ une autre solution réalisable

$$c \cdot y = c_B \cdot y_B + c_N \cdot y_N = c_B \cdot B^{-1}b + (c_N - c_B \cdot B^{-1}N) \cdot y_N$$

$$c \cdot y - c \cdot x = (c_N - c_B \cdot B^{-1}N) \cdot y_N \geq 0$$

(par l'hypothèse du théorème et du fait que y est réalisable). Donc x donne la valeur minimale à $c \cdot x$ parmi toutes les solutions réalisables ; autrement dit, x est optimale. $\diamond$

1.5 Comment améliorer une solution de base non optimale

Nous supposons que nous avons obtenu une "base" B, à laquelle correspond une solution de base non optimale. Par suite $c_N - c_B \cdot B^{-1}N$ doit posséder des composantes positives.
Notations :

$$\hat{c} = c_N - c_B \cdot B^{-1}N \qquad \hat{b} = x_B = B^{-1}b \qquad \hat{N} = B^{-1}N$$

$$\pi = c_B \cdot B^{-1}$$

Pour tout vecteur réalisable y :

$$c \cdot y = \pi \cdot b + \hat{c} \cdot y_N \tag{3}$$

[2]Les variables dans x_B sont dites "de base", celles de X_N "hors base".

$$y_B = \hat{b} - \hat{N} y_N \tag{4}$$

Numérotons de 1 à m les colonnes de B, et de $m+1$ à n les colonnes de N.

$$c_B = (c_1, c_2, \ldots, c_m) \qquad c_N = (c_{m+1}, c_{m+2}, \ldots, c_n)$$

$$\hat{c} = (\hat{c}_{m+1}, \hat{c}_{m+2}, \ldots, \hat{c}_n)$$

Supposons $x = (x_B, 0)$ solution de base non optimale ; soit s un indice ($m+1 \leq s \leq n$) tel que $\hat{c_s} < 0$; choisissons un vecteur y de la forme

$$y = (y_B, y_N), \quad y_N = (0, 0, \ldots, \theta, \ldots 0) \tag{5}$$

(y_N est nul sauf pour la composante y_s : $y_i = 0, m+1 \leq i \leq n, i \neq s ; y_s = \theta$). Comme y doit être réalisable, $\theta \geq 0$.

Alors d'après (4) et (5), y_B est donné par

$$y_B = \hat{b} - \theta \hat{N}_s \tag{6}$$

où $\hat{N}_s$ désigne la s-ième colonne de $\hat{N}$.

Il faut alors distinguer 2 cas :

1. $\hat{N}_s \leq 0$:
   on peut donner à θ des valeurs positives arbitrairement grandes sans que y cesse d'être réalisable : le problème est dit <u>non borné</u> : il n'y a pas de solution finie.

2. $\{ \ i \ \mid \ \hat{N}_{i,s} > 0 \ \} \neq \emptyset$
   il est permis d'accroître θ tant que chaque composante de y_B reste positive, de façon à garder une solution réalisable ; d'après (6), on peut donc choisir pour θ :

   $$\theta = \hat{b}_j / \hat{N}_{j,s} \tag{7}$$

   où j est l'indice donnant la valeur maximale à $\hat{b}_i / \hat{N}_{i,s}$:

   $$\hat{b}_j / \hat{N}_{j,s} = \ \max \ \{ \ \hat{b}_i / \hat{N}_{i,s} \ \mid \ \hat{N}_{i,s} > 0 \ \} \tag{8}$$

   Notons qu'alors la j-ième composante de y_B s'annule, et d'autre part que $\theta \geq 0$, puisque x est réalisable et $\hat{b} = x_B$.
   Calculons la valeur de la fonction coût pour y ; d'après (3) et (5),

   $$c \cdot y - c \cdot x = \hat{c} \cdot y_N = \hat{c_s} \theta \leq 0 \tag{9}$$

   Donc la fonction coût décroît si l'on passe de x à y.
   Remarques :
   • Si $\hat{b}_j = 0$, la fonction coût ne change pas de valeur : c'est un cas "dégénéré" assez rare en pratique ; il existe cependane des stratégies permettant

de se prémunir contre une telle dégérescence.
• Si plusieurs choix sont possibles pour l'indice s, on obtient une décroissance maximale du coût en prenant l'indice correspondant à la plus petite valeur négative de $\hat{c_s}$:

$$\hat{c_s} = \max \{ \hat{c'_s} \mid \hat{c'_s} < 0 \} \tag{10}$$

1.6 Algorithme du simplexe

Pour obtenir un algorithme permettant d'arriver à l'optimal, il reste à s'assurer que l'on peut bien considérer y comme une solution de base. Comme la composante y_{N_s} de y_N devient non nulle alors que la composante y_{B_i} s'annule, le changement de base sera consiste en un échange de colonnes : s entre dans la base et i sort de la base. On vérifie aisément que celà est possible en effectuant sur la matrice des opérations semblables au "pivotage" dans la méthode de Gauss pour la résolution de systèmes d'équations.

L'algorithme "du simplexe" se résume ainsi, avec les notations du paragraphe précédent :

1. partir d'une solution de base réalisable.
2. $\hat{c_s} \geq 0$ pour tout s ?
   Alors nous avons atteint l'optimum.
   Fin de l'algorithme.
3. Choisir s tel que $\hat{c_s} < 0$.
   Si $\hat{N}_{i,s} \leq 0$, le problème n'a pas de solution finie.
   Fin de l'algorithme.
4. Choisir j tel que $\hat{N}_{j,s} > 0$ et maximisant $\hat{b}_i/\hat{N}_{i,s}$.
   Former la nouvelle base : x_{B_j} sort de la base, x_{N_s} entre dans la base.
   Retourner en 2. ($\hat{c}$ est bien sûr actualisé pour correspondre à la nouvelle matrice de base.)

Le nombre d'itérations est fini (sauf cas de dégénérescence), car on montre qu'il n'existe qu'un nombre fini (peut-être élevé) de bases distinctes. L'interprétation géométrique de l'algorithme est que l'on se déplace à chaque itération d'un sommet du polyèdre $(A \cdot x = b, x \geq 0)$ à un autre sommet.

Exemple

$$\begin{array}{lllllll}
\text{Min} & -4x_1 - 5x_2 & & & & & \\
 & 2x_1 & +x_2 & +x_3 & & & = 8 \\
 & x_1 & +2x_2 & & +x_4 & & = 7 \\
 & & x_2 & & & +x_5 & = 3 \\
 & x_1, \ldots, x_5 \geq 0 & & & & &
\end{array} \tag{11}$$

Solution de base associée à la base $(3,4,5)$: $x_1 = 0, x_2 = 0, x_3 = 8, x_4 = 7, x_5 = 3$.

Exercice : dérouler l'algorithme du simplexe sur cet exemple.
Donner une interprétation géométrique dans le plan x_1, x_2.

1.7 Initialisation de l'algorithme du simplexe

On peut se ramener au cas précédent où l'on dispose d'une solution de base réalisable par résolution d'un problème auxilliaire, pour lequel nous disposons d'une solution de base réalisable évidente. L'idée est d'introduire des variables w_i "mesurant" l'écart entre Ax et b, et de minimiser la somme de ces variables. On peut supposer que $b \geq 0$, au besoin en changeant de signe certaines équations. Considérons alors le problème :

$$\begin{array}{ll} \text{Min} & w_1 + w_2 + \ldots + w_m \\ & Ax + w = b \\ & x \in \mathbf{R}^n, \ w \in \mathbf{R}^m \\ & x \geq 0, \ w \geq 0 \end{array} \tag{12}$$

Grâce à l'hypothèse sur b, $(x = 0, w = b)$ est une solution de base réalisable, et nous pouvons appliquer l'algorithme du simplexe (on observera que la matrice du problème auxilliaire (12) est de rang m). Après convergence sur le problème (12), nous obtenons une solution de base optimale (x, w) de (12), satisfaisant donc : $Ax + w = b$. Deux situations peuvent se présenter :

1. Le minimum du problème auxilliaire est 0 : ceci n'est possible que si tous les w_i sont nuls, et par conséquent x est une solution réalisable de base du problème d'origine. L'algorithme du simplexe s'applique alors en partant de cette solution.

2. Le minimum du problème auxilliaire est strictement positif ; il ne peut alors exister de vecteur satisfaisant $Ax = b$ et $x \geq 0$; le problème posé n'admet pas de solution.

2 Dualité

2.1 Dual d'un programme linéaire

Considérons le PL suivant :

$$\begin{array}{ll} \text{Max} & c \cdot x \\ & Ax \leq b \\ & x \in \mathbf{R}^n \\ & x \geq 0 \end{array} \tag{13}$$

On appelle *dual* de ce problème[3] le PL :

$$\begin{array}{ll} \text{Min} & b \cdot y \\ & A^T y \geq c \\ & y \in \mathbf{R}^m \\ & y \geq 0 \end{array} \tag{14}$$

[3] Par opposé au dual, (13) est dit "problème primal"

Théorème de la dualité

- *Pour tout couple de solutions réalisables x et y pour (13) et (14), $c \cdot x \leq b \cdot y$.*
- *Si x et y sont deux solutions réalisables de (13) et (14), $c \cdot x = b \cdot y$ si et seulement si x et y sont optimales.*
- *Etant donnés un PL et son dual, seulement quatre cas sont possibles :*
  1. *le primal (13) et le dual (14) n'ont pas de solution réalisable ;*
  2. *le primal (13) admet des solutions réalisables non bornées, et le dual (14) n'admet pas de solution réalisable ;*
  3. *le dual (14) admet des solutions réalisables non bornées, et le primal (13) n'admet pas de solution réalisable ;*
  4. *le primal (13) et le dual (14) admettent des solutions réalisables ; (13) et (14) admettent chacun une solution optimale, respectivement x et y, et les valeurs des optima sont égales : $c \cdot x = b \cdot y$.*

Tout PL (le "primal") peut se mettre sous la forme (13) ; on peut donc son problème dual par cet intermdíaire. On peut également appliquer les quelques règles simples suivantes :

- Si le primal est un problème de maximisation, le dual sera un problème de minimisation.
- Si le primal a n variables, son dual a m contraintes.
- La matrice du dual est la transposée A^T de la matrice du primal.
- La correspondance entre les contraintes sur les variables x_i du primal et celles y_j du dual sont résumées dans le tableau ci-dessous :

Primal	Dual
j-ème contrainte : $\sum_i a_{j,i}x_i \geq 0$	j-ème variable : $y_j \leq 0$
j-ème contrainte : $\sum_i a_{j,i}x_i \leq 0$	j-ème variable : $y_j \geq 0$
j-ème contrainte : $\sum_i a_{j,i}x_i = 0$	j-ème variable : $y_j \in \mathbf{R}^m$
i-ème variable : $x_i \geq 0$	i-ème contrainte : $\sum_j a_{i,j}y_j \geq 0$
i-ème variable : $x_i \leq 0$	i-ème contrainte : $\sum_j a_{i,j}y_j \leq 0$
i-ème variable : $x_i \in \mathbf{R}^n$	i-ème contrainte : $\sum_j a_{i,j}y_j = 0$

Exemple

Le dual de (11) est :

$$\begin{array}{ll} \text{Max} & 8y_1 + 7y_2 + 3y_3 \\ & 2y_1 + y_2 \leq -4 \\ & y_1 + 2y_2 + y_3 \leq 5 \\ & y_1 \leq 0, y_2 \leq 0, y_3 \leq 0 \end{array} \tag{15}$$

2.2 Lemme de Farkas

Lemme de Farkas *Soit une matrice A et un vecteur b. Il existe un vecteur $x \geq 0$ tel que $Ax = b$ si et seulement si*

$$A^T y \geq 0 \quad \Longrightarrow \quad y \cdot b \geq 0$$

Interprétation géométrique : b appartient au cône généré par les vecteurs colonnes de A ssi on ne peut pas le séparer du cône par un hyperplan[4].
Lemme de Farkas affine *Soit une matrice A et 2 vecteurs b et c. S'il existe x tel que :*

$$Ax \leq b$$

et

$$Ay \leq b \implies c \cdot y \leq \delta$$

Alors il existe $y \geq 0$ tel que $A^T y = c$ et $y \cdot b \leq \delta$.

2.3 Algorithme dual du simplexe

L'algorithme du simplexe peut s'appliquer au problème dual, ce qui est équivalent à la résolution du primal par le théorème de la dualité. Cette approche présente entre autres avantages la possibilité de permettre une solution incrémentale : si l'on a obtenu la solution du dual (14), et que l'on ajoute des contraintes au problème primal, la solution déjà obtenue fournit un point de départ dual-réalisable pour le problème augmenté.

3 Problèmes en nombres entiers

On ajoute au PL (2) des contraintes d'intégrité sur la solution[5] :

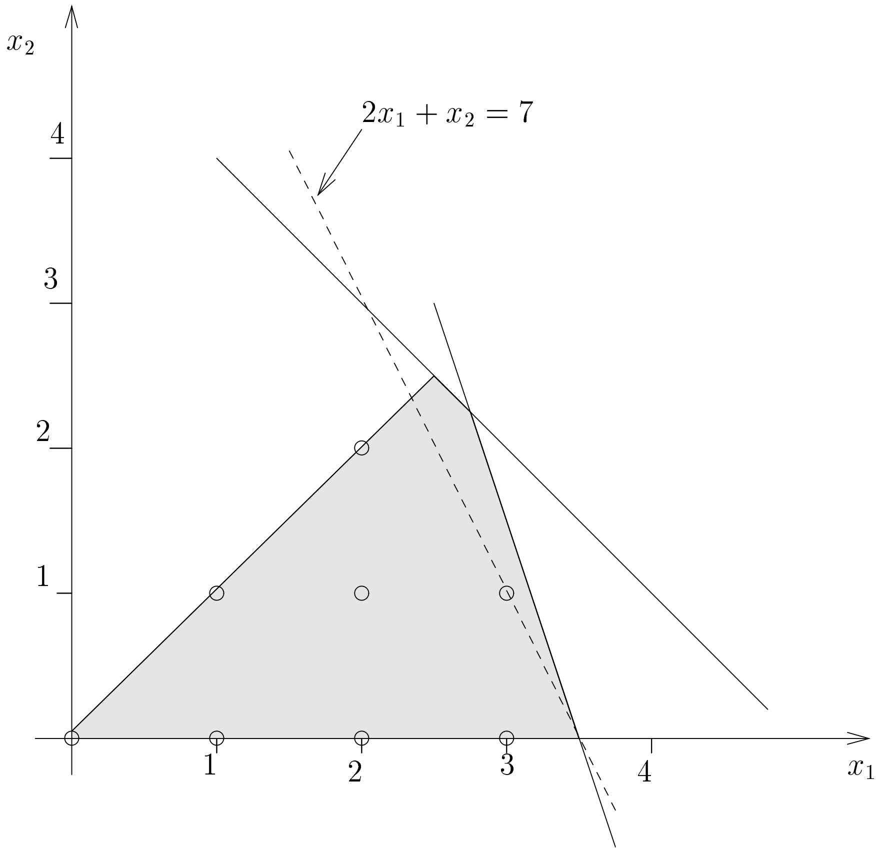
$$\begin{array}{ll} \text{Min} & c \cdot x \\ & Ax \leq b, \quad x \in \mathbb{Z}^n, \quad x \geq 0 \end{array} \tag{16}$$

3.1 Principe des méthodes de "coupe"

Observons tout d'abord que si x est solution optimale de (2) et que tous les coefficients de x sont entiers, alors x est également solution optimale de (16).
Le principe général de la *méthode des coupes* est de se ramener à une succession de programmes linéaires en variables réelles, que l'on résoud par l'algorithme du simplexe tel qu'il est décrit ci-dessus. Les *coupes* sont des contraintes que l'on ajoute au problème de manière à éliminer les solutions optimales non entières.

[4] Le cône généré par des vecteurs $x_1, x_2, \ldots, x_m$ est l'ensemble des combinaisons linéaires positives des x_i : $\{\lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_m x_m \mid \lambda_1, \ldots, \lambda_m \geq 0\}$

[5] On suppose que les coefficients de A et b sont entiers.



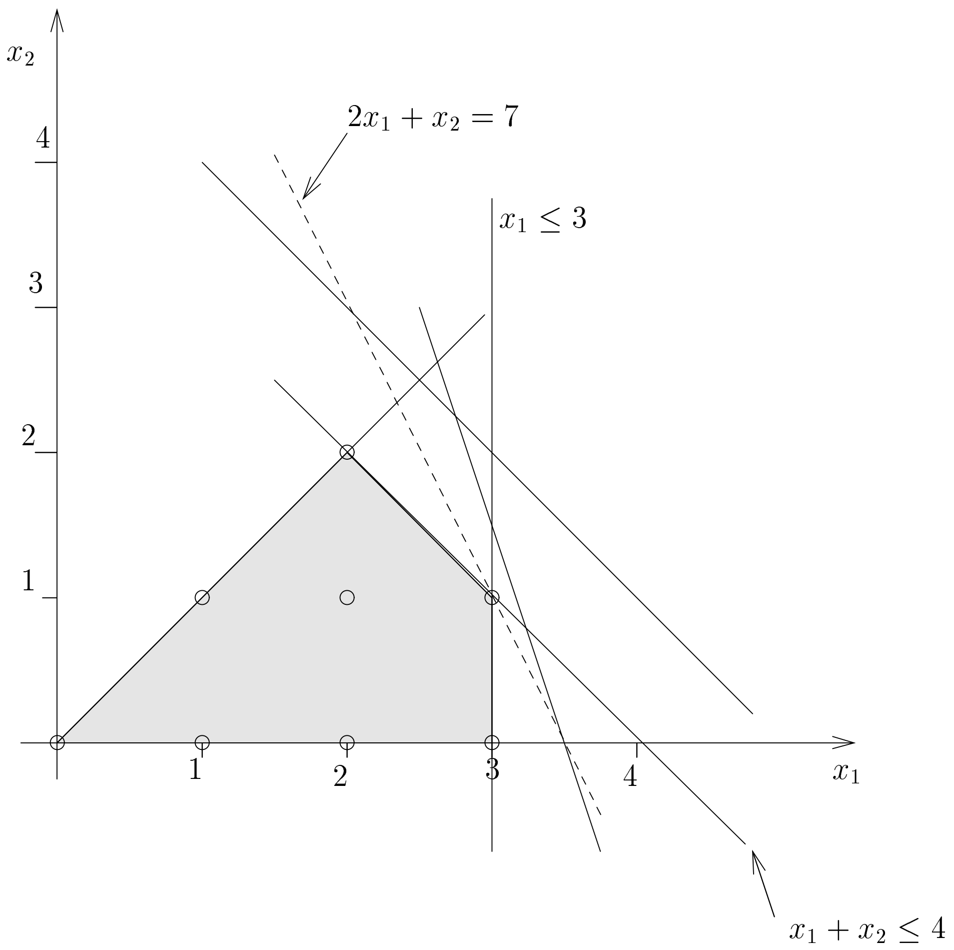
<u>Exemple</u>

$$
\begin{array}{ll}
\text{Max} & 2x_1 + x_2 \\
& x_1 + x_2 \leq 5 \\
& -x_1 + x_2 \leq 0 \\
& 6x_1 + 2x_2 \leq 21 \\
& x_1 \geq 0, x_2 \geq 0, x_1 \text{ et } x_2 \text{ entiers}
\end{array}
\tag{17}
$$

Sans les contraintes d'intégrité, le problème admet la solution $(11/4, 9/4)$, mais ce point n'est pas entier. En ajoutant les contraintes

$$
\left\{
\begin{array}{l}
x_1 + x_2 \leq 4 \\
x_1 \leq 3
\end{array}
\right.
$$

on obtient un problème dont la solution en variables réelles est $(3, 1)$, donc identique à la solution du problème en variables entières (17).



3.2 Coupe de Gomory

Un certain nombre de variantes de la méthode des coupes existe ; elles sont basées sur le théorème ci-dessous du à Gomory (1958), qui permet de découvrir de nouvelles inégalités impliquées par un programme linéaire avec contraintes d'intégrité.

Définition Etant donné un programme linéaire en nombres entiers (P), notons (P') le problème *relaxé*, i.e. obtenu en omettant les contraintes d'intégrité. L'inéquation $a \cdot x \leq \alpha$ est dite *valide* si elle est satisfaite par toute solution réalisable de (P). Une *coupe* est une inéquation valide qui n'est pas satisfaite par toute solution réalisable de (P').

Exemple Dans l'exemple (17) $x_1 + x_2 \leq 4$ et $x_1 \leq 3$ sont des coupes.

Supposons le PL (P) écrit dans une base de matrice B :

$$Ax = b \iff Bx_B + Nx_N = b \iff x_B + \hat{N}x_N = \hat{b} \tag{18}$$

$$x_B = (x_1, \ldots, x_m), \quad x_N = (x_{m+1}, \ldots, x_n)$$

<u>Notation :</u> Pour tout nombre α on note $\lfloor \alpha \rfloor$ la partie entière de α et $\langle \alpha \rangle = \alpha - \lfloor \alpha \rfloor$ sa “partie fractionnaire” ; on a : $0 \leq \langle \alpha \rangle < 1$.
<u>Théorème</u> *Pour tout nombre $\alpha \in \mathbf{R}$, l'inéquation suivante est valide :*

$$\sum_{s=m+1}^{s=n} (\lfloor \alpha \rfloor \hat{N}_{i,s} - \lfloor \alpha \hat{N}_{i,s} \rfloor) x_s \geq \lfloor \alpha \rfloor \hat{b}_i - \lfloor \alpha \hat{b}_i \rfloor \tag{19}$$

<u>Démonstration</u> Multiplions la i-ème contrainte de (18) par α $(1 \leq i \leq m)$.

$$\alpha x_i + \sum_{s=m+1}^{s=n} \alpha \hat{N}_{i,s} x_s = \alpha \hat{b}_i$$

Décomposons chaque coefficient en partie entière et partie fractionnaire :

$$\lfloor \alpha \rfloor x_i + \sum_{s=m+1}^{s=n} \lfloor \alpha \hat{N}_{i,s} \rfloor x_s + \langle \alpha \rangle x_i + \sum_{s=m+1}^{s=n} \langle \alpha \hat{N}_{i,s} \rangle x_s = \lfloor \alpha \hat{b}_i \rfloor + \langle \alpha \hat{b}_i \rangle$$

Comme $0 \leq \langle \alpha \rangle < 1$ et $x_s \geq 0$:

$$\lfloor \alpha \rfloor x_i + \sum_{s=m+1}^{s=n} \lfloor \alpha \hat{N}_{i,s} \rfloor x_s \leq \lfloor \alpha \hat{b}_i \rfloor + \langle \alpha \hat{b}_i \rangle$$

Le membre de gauche est entier et $\langle \alpha \hat{b}_i \rangle < 1$; par suite :

$$\lfloor \alpha \rfloor x_i + \sum_{s=m+1}^{s=n} \lfloor \alpha \hat{N}_{i,s} \rfloor x_s \leq \lfloor \alpha \hat{b}_i \rfloor \tag{20}$$

Multiplions la i-ème contrainte de (18) par $\lfloor \alpha \rfloor$.

$$\lfloor \alpha \rfloor x_i + \sum_{s=m+1}^{s=n} \lfloor \alpha \rfloor \hat{N}_{i,s} x_s = \lfloor \alpha \rfloor \hat{b}_i \tag{21}$$

En soustrayant (21) et (20), on obtient (19). ◇

Si $\hat{b}_i$ n'est pas entier, la solution de base associée à B n'est pas entière ; alors, pour certaines valeurs de α, (19) est une coupe, en particulier pour $\alpha = 1$. En effet, pour $\alpha = 1$, (19) s'écrit :

$$\sum_{s=m+1}^{s=n} \langle \hat{N}_{i,s} \rangle x_s \geq \langle \hat{b}_i \rangle > 0 \tag{22}$$

Et (22) n'est pas satisfait par la solution de base associée à B (qui est telle que $x_s = 0, m+1 \leq s \leq n$).

Forme générale d'un algorithme de coupes

1. Résoudre (P') (le problème relaxé).
2. Si la solution optimale obtenue est entière, c'est aussi la solution optimale de (P).
   Fin de l'algorithme.
3. Choisir une ligne i telle que $\hat{b}_i$ ne soit pas entier. Ajouter la contrainte (22) et retourner à l'étape 2.

Voir les références ci-dessous pour les critères de choix de i assurant la convergence de l'algorithme.

4 Parametric Integer Programming (P. Feautrier)

La méthode "*PIP*" développée par Paul Feautrier est consacrée à la solution de systèmes d'inéquations linéaires, où le critère est donné par l'ordre lexicographique $\ll$: on recherche le vecteur x satisfaisant les contraintes et tel que $x \ll y$ pour tout autre vecteur satisfaisant les mêmes contraintes.
Définition **Ordre lexicographique** strict : soit $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n), x \neq y$;

$$(x_1, \ldots, x_n) \ll (y_1, \ldots, y_n) \iff (\ x_1 < y_1\ ) \text{ ou } (x_2, \ldots, x_n) \ll (y_2, \ldots, y_n)$$

On notera également $\ll$ la relation d'ordre partiel associée ($x = y \Longrightarrow x \ll y$).

D'autre part le second membre des inégalités n'est pas supposé constant, mais dépendant linéairement de *paramètres* ; soit z le vecteur de ces paramètres ; on considère donc la famille de polyèdres paramétrés par z :

$$\mathcal{F}(z) = \{\ x \ \mid\ x \in \mathbf{R}^n,\ Ax + Mz + c \geq 0,\ x \geq 0\ \} \tag{23}$$

Par ailleurs, les *paramètres* doivent vérifier un certain nombre de contraintes appelées le *contexte* du problème :

$$Mz + h \geq 0 \qquad (24)$$

Il s'agit de déterminer si le polyèdre $\mathcal{F}(z)$ de $\mathbf{R}^n$ est vide (dans le contexte (24), et s'il ne l'est pas, d'en donner le minimum lexicographique en fonction de z.
La méthode est basée sur l'algorithme *dual* du simplexe ; comme les seconds membres sont des formes linéaires en z, comment un tel algorithme peut-il fonctionner ?

- Le changement de matrice de base dans les méthodes de simplexe est, comme on l'a vu, calculé à l'aide de formules de pivotage de la forme :

  $$b'_i(z) = \hat{b}_i(z) - \hat{b}_j(z)\beta/\beta'$$

  où β et β' sont des nombres ; il est clair que b'_i est encore une forme linéaire en z après cette opération ; de même, la matrice reste constituée de constantes numériques après pivotage.

- La sélection du "pivot", i.e. de la variable $x_{B\,i}$ quittant la base, et de la variable $x_{N\,s}$ entrant dans la base, est conditionnée par le signe du second membre $\hat{b}(z)$ dans le contexte (24) pour l'algorithme dual[6] ;

  - Si tous les $\hat{b}_j(z)$ sont positifs, la solution est donnée par les n premières composantes de $\hat{b}$.
  - S'il existe j tel que $\hat{b}_j(z) < 0$:
    * Si tous les coefficients de la ligne j, $B_{j,i}$, sont négatifs, il n'y a pas de solution.
    * S'il existe un j tel que $B_{j,i} > 0$, on exécute une étape de pivotage ; on montre que la solution de base correspondante décroît pour l'ordre lexicographique.
  - Sinon choisir un $\hat{b}_j(z)$ dont le signe n'est pas connu.
    * Soit le nouveau contexte : $\{\ z \mid Kz{+}h, \hat{b}_j(z) \geq 0\ \}$. On poursuit la résolution dans ce nouveau contexte : soit x^+ la solution obtenue dans ce contexte. On pose $x^+ = \perp$ si cette solution n'existe pas[7].
    * Soit le nouveau contexte : $\{\ z \mid Kz + h, \hat{b}_j(z) < 0\ \}$, et x^- la solution obtenue dans ce contexte.

  La solution s'exprime sous la forme d'une condition :

  $$\textbf{si } \hat{b}_j(z) \geq 0 \textbf{ alors } x^+ \textbf{ sinon } x^-$$

[6] Contrairement à l'algorithme "primal", exposé dans le paragraphe 1, où ce choix est déterminé par le signe des coefficients de coût $\hat{c}_s$.

[7] $\perp$ est traditionnellement un symbole signifiant "pas d'information" ou "pas de solution"

Notons qu'éventuellement les solutions x^+ et x^- s'expriment elles-mêmes sous forme conditionnelle, de sorte que finalement la solution du problème d'origine se présente sous la forme d'un arbre dont les branches sont labellés par **alors** ou **sinon** . Feautrier dénomme "*quasts*" ("quasi affine selection trees") de telles expressions.

Exemple Trouver le minimum lexicographique de :

$$\{ \ (i,j) \ \mid \ 0 \leq i \leq m, 0 \leq j \leq n, 2i+j=k \ \}$$

dans le contexte $m \geq 0, n \geq 0, k \in Z$ (z est donc un triplet (m,n,k)).
La méthode PIP donne le résultat suivant :

$$\begin{array}{rl} (i,j) = & \textbf{si } (-k+2m+n) \geq 0 \\ & \quad \textbf{alors si } (k-2m) \geq 0 \\ & \quad\quad\quad \textbf{alors } (m, k-2m) \\ & \quad\quad\quad \textbf{sinon } (k/2, 0) \\ & \quad \textbf{sinon } \perp \end{array}$$

Voir [Fea88] pour les détails de l'algorithme et les preuves.

Références

[Dan63] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New-Jersey, 1963.

[Fea88] Paul Feautrier. Parametric integer programming. *RAIRO, Recherche Opérationnelle*, 22(3):243–268, September 1988.

[GN72] R. S. Garfinkel and G. N. Nemhauser. *Integer Programming.* Wiley-Interscience, 1972.

[Min83] M. Minoux. *Programmation Mathématique.* Dunod, 1983.

[New72] Morris Newman. *Integral matrices.* Pure and applied mathematics. Academic press, 1972.

[Pug92] William Pugh. The omega test: a fast and practical integer programming method for dependence analysis. *Communications of the ACM*, 35(8):102–114, August 1992.

[Sak84a] Michel Sakarovitch. *Optimisation combinatoire : méthodes mathématiques et algorithmiques : graphes et programmation lineaire.* Enseignement des sciences. Hermann, 1984.

[Sak84b] Michel Sakarovitch. *Optimisation combinatoire : méthodes mathématiques et algorithmiques : programmation discrete.* Enseignement des sciences. Hermann, 1984.

[Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley-Interscience, 1986.

[SM89] Harvey M. Salkin and Kamlesh Mathur. *Foundations of Integer Programming.* North-Holland, 1989.