Programmation linéaire réelle

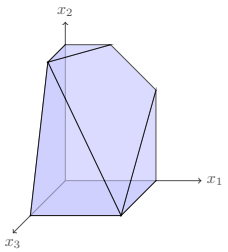
François Schwarzentruber

Définition 1 (programme linéaire) Un programme linéaire réel est la donnée d'une fonction linéaire à maximiser ou minimiser, sous des contraintes linéaires (in.égalités), où les variables prennent leurs valeurs dans $\mathbb{R}$.

Exemple 2

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.

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



Définition 3 **Programmation linéaire réelle**

entrée : un programme linéaire réel P
sortie : une solution optimale de P, non borné s'il n'y a pas d'optimum, ou impossible si les contraintes sont inconsistantes.

1 Formes normales pour les programmes linéaires

1.1 Programme canonique

Définition 4 (programme canonique) Un programme canonique est de la forme :

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

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

d = dimension
m = nombre d'inégalités linéaires

Proposition 5 Tout programme linéaire peut s'écrire sous la forme d'un programme canonique équivalent.

1.2 Programme équationnel

Définition 6 (programme équationnel) Un programme équationnel est de la forme

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

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

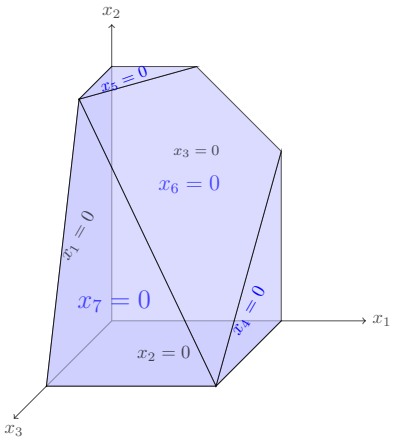
Proposition 7 Tout programme canonique se réécrit en un programme équationnel équivalent en introduisant des **variables d'écart** $x_{d+1}, \ldots, x_{d+m}$.

Exemple 8 On introduit des **variables d'écart** x_4, x_5, x_6, x_7 de la façon suivante :

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

$\underbrace{x_1 \ldots x_d}_{d \text{ variables dans l'espace de départ}}$ $\underbrace{x_{d+1} \ldots x_{d+m}}_{m \text{ variables d'écart} = m \text{ contraintes}}$

Face = une équation $x_i \geq 0$



2 Algorithme du simplexe

2.1 Principe

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

2.2 Représentation d'un sommet du polyèdre

Sommet = intersection de d faces = la donnée de d variables nulles appelées variables hors base

Définition 9 (variables de base) Une base B est un sous-ensemble de $\{1, \dots, n\}$ de cardinal m telle que A_B soit de rang m. Les variables dans x_B sont les variables de base.

Définition 10 (variables hors base) On note $N = \{1, \dots, n\} \setminus B$. Les variables dans x_N sont hors base.

Définition 11 (tableau) Étant donné une base B, un tableau est un programme linéaire sous la forme

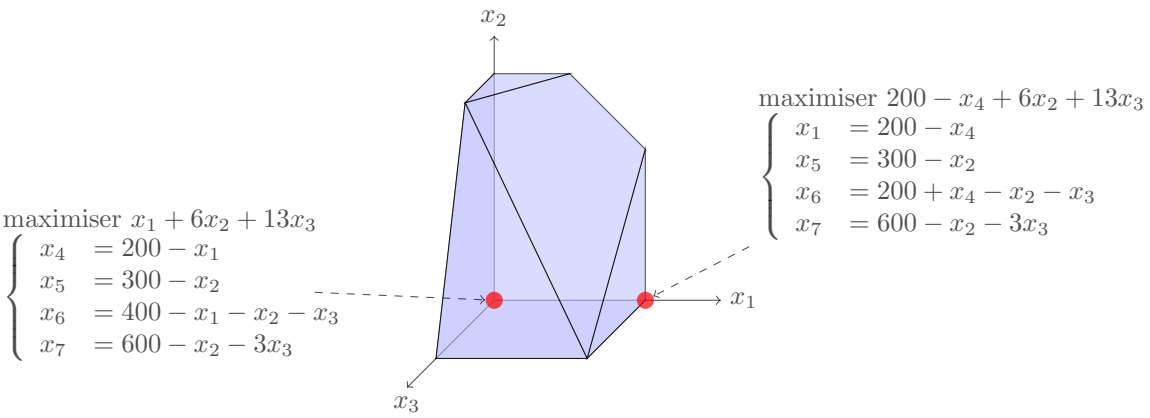
$$\begin{array}{l} \text{maximiser } v + c'^t x_N \\ \begin{cases} x_B = p - A' x_N \\ x \geqslant 0 \end{cases} \end{array}$$

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 $x_N \in \mathbb{R}^{n-m}$.

tableau	$=$ sommet à l'intersection des $x_i = 0$ où $i \in N$
v	$=$ valeur de la fonction objectif en ce sommet
p	$=$ valeurs des variables de base en ce sommet

Définition 12 (solution basique) Un tableau 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 $x_N = 0$.

Exemple 13



2.3 Pseudo-code

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 solution basique de τ

entrée : un tableau τ admettant une solution basique
sortie : un tableau équivalent, admettant une solution basique, améliorant non strictement l'objectif, de même espace de solutions, et ou alors lève une exception non borné
fonction pivot(τ)
renvoyer le 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 $x_s := 0$ $\qquad x_e := 0 \qquad x_s \nearrow$

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 $i \in B$, $j \in N$:

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

2.4 Propriétés de l'algorithme

Proposition 14 (admis car lourd) La fonction pivot est correct.

Théorème 15 L'algorithme du simplexe, s'il termine, retourne bien la valeur maximale de l'objectif.

2.5 A toute base, son unique tableau

Proposition 16 Étant donné une base B, il y a un unique tableau équivalent au programme équationnel. C'est :

$$\begin{array}{l} \text{maximiser } v + c'^t x_N \\ \begin{cases} x_B = p - A' x_N \\ x \geqslant 0 \end{cases} \end{array} \qquad \text{avec } v = c_B^t A_B^{-1} b \,;\; c' = c_N - (c_B^t A_B^{-1} A_N)^t \,;\; p = A_B^{-1} b \,; \text{ et } A' = A_B^{-1} A_N.$$

3 Terminaison de l'algorithme du simplexe

Définition 17 (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 18 (terminaison) En appliquant la règle de Bland, l'algorithme du simplexe termine.

Remarque 19 En pratique, la règle de Bland est lente. On préfère perturber un peu b pour supprimer les dégénérescences. ([DPV08], p. 218)

4 Prétraitement : obtenir tableau avec solution basique

entrée : un programme canonique
sortie : impossible ou un tableau avec solution basique de même maximum

fonction prétraitement $\left(L : \begin{cases} \text{maximiser } c^t x \\ Ax \leqslant b \\ x \geqslant 0 \end{cases} \right)$

  si $b \geq 0$ **alors**
    renvoyer L

  soit le programme $L_{aux} : \begin{cases} \text{maximiser } -x_0 \\ Ax - x_0 \leqslant b \\ x \geqslant 0 \end{cases}$ où x_0 est une variable auxiliaire

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

  1. mettre L_{aux} sous forme équationnelle, puis sous tableau
  2. faire entrer x_0 dans la base, et sortir la variable x_k où b_k minimum

  //le tableau courant admet une solution basique

  3. Lancer l'algorithme du simplexe sur ce tableau
  si le maximum est 0 **alors**
    renvoyer le tableau obtenu depuis celui à la fin de 4., en réécrivant la fonction objectif de L avec les variables hors bases et en enlever x_0
  sinon
    lever une exception impossible

Proposition 20 Le programme L_{aux} admet une solution.

Proposition 21 L'espace des solutions de L est non vide ssi l'optimum de L_{aux} est 0.

Proposition 22 Après l'étape 2, le tableau admet une solution basique.

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.