Prof. F. Eisenbrand EPFL - DISOPT

Optimisation Discrète

Adrian Bock Semestre de printemps 2011

Série 5 **24 mars 2011**

Exercice 1 $(*), (\Delta)$

Considérer l'algorithme du simplexe appliqué au programme linéaire

$$\max\{c^T x \colon Ax \leq b\}.$$

Soit la matrice A de plein rang-colonne. Donner une preuve ou un contre-exemple aux affirmations suivantes :

(i) Le sommet d'un toit est toujours admissible pour les itérations précédentes de l'algorithme.

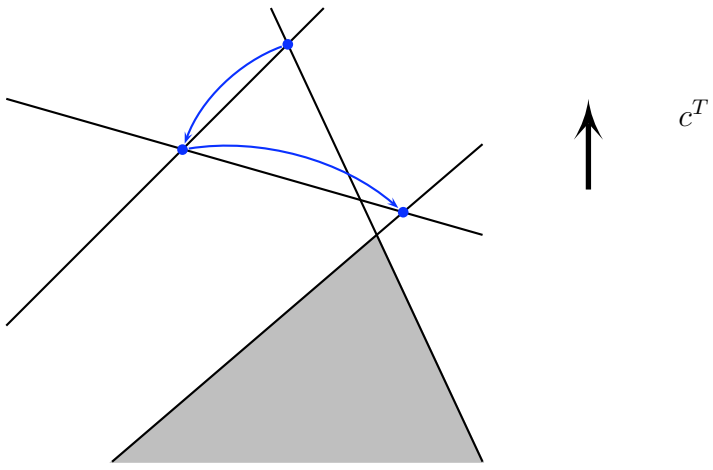
(ii) Un index qui est juste sorti du toit ne peut pas rentrer dans le toit à l'itération suivante.

(iii) Un index qui est juste entré dans le toit ne peut pas quitter le toit à l'itération suivante.

(iv) Dans l'étape iii), on trouve $\lambda^* = 0$ seulement si le toit courant est dégénéré.

Solution

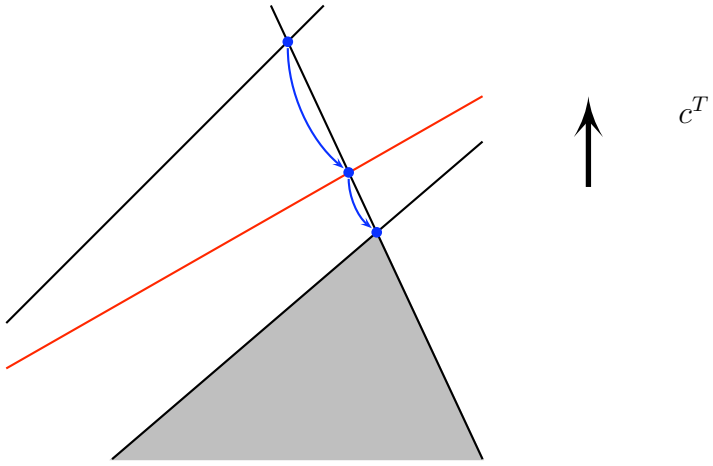
(i) Non. Contre-exemple :



(ii) Oui.
Vu que le sommet du toit de l'itération suivante est admissible pour le toit précédent, en particulier la contrainte correspondante à l'index juste sorti est satisfaite. Donc l'index ne peut pas rentrer à l'itération suivante car on cherche à l'étape ii) une contrainte violée.

Noter que l'index peut bien sûr rentrer dans le toit à la deuxième itération après que l'index est sorti du toit.

(iii) Non. contre-exemple :



(iv) Oui.
Si on trouve $\lambda^* = 0$, on sait qu'il existe un index tel que la contribution de la ligne correspondante à la combinaison conique de l'objectif est zéro. Aprés l'exercice 2 de la série 4, le toit courant doit être dégénéré.

Noter que la réciproque est juste seulement s'il y a un index k tel que $z_k = 0$ dans la combinaison conique de l'objectif et s'il y a un coefficient strictement négatif dans la combinaison linéaire de $-a_i$ où i est l'index à entrer dans le toit.

Exercice 2

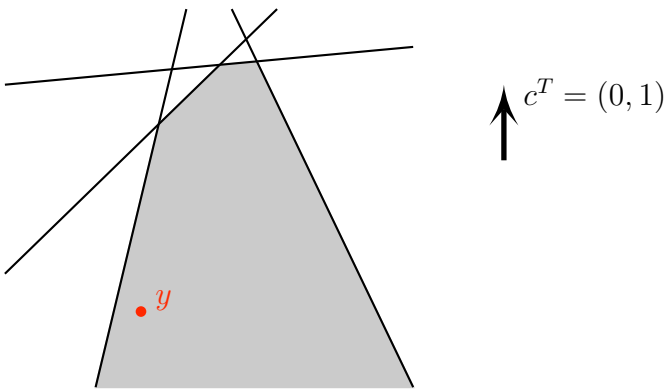
Considérer le programme linéaire

$$\max\{c^T x \colon Ax \leq b\} \qquad \text{avec } A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n.$$

Soit y une solution admissible telle que $Ay < b$. Donner l'esquisse d'un algorithme pour trouver un sommet x du polyèdre $P = \{x \in \mathbb{R}^n \colon Ax \leq b\}$ tel que $c^T x \geq c^T y$.

L'algorithme doit terminer après n étapes au plus et à chaque itération, l'élimination de Gauss-Jordan peut être utilisée, mais pas l'algorithme du simplexe.

(i) D'abord, trouver une stratégie pour trouver le sommet optimal dans le dessin ci-dessous.

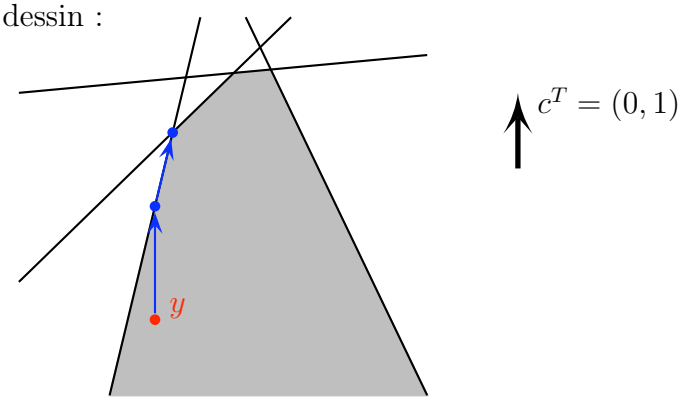


(ii) En déduire une stratégie générale en formalisant l'étape précédente.

Solution

Idée : Déplacer y en direction de l'objectif sans sortir du polyèdre P. Augmenter le nombre des contraintes satisfaites avec égalité à chaque iteration.

(i) La solution sur le dessin :



Noter que le sommet trouvé n'est pas nécessairement optimal par rapport à l'objectif.

(ii) On va implémenter la stratégie décrite ci-dessus :

1. D'abord on augmente y en direction de l'objectif jusqu'au bord du polyèdre.
   Soit L l'ensemble des lignes de la matrice A qui sont satisfaites avec égalité. De plus, soit d le maximal nombre des vecteurs linéairement indépendantes dans L.
2. **Tant que** $d < n$, il existe un vecteur v non-nul dans le complément orthogonal $L^\perp$ de L. Donc on a aussi $-v \in L^T$. Alors soit v, soit $-v$ a un produit scalaire positif avec l'objectif c. On déplace y dans cette direction jusqu'au bord du polyèdre. Mettre à jour y,L et d.

On trouve que le résultat de cet algorithme est un sommet du polyèdre P. Il termine car chaque contrainte qui est satisfaite une fois avec égalité reste dans l'ensemble L par définition de v et à chaque itération, au moins une contrainte supplémentaire entre dans L. Il reste à donner une idée de comment implémenter les étapes.
L'augmentation du vecteur y est faite en trouvant le plus petit $\lambda \in \mathbb{R}$ tel que $A(y + \lambda v) \leq b$ (respectivement $A(y - \lambda v) \leq b$). On applique l'élimination de Gauss-Jordan aux lignes satisfaites avec égalité pour facilement déterminer le nombre d. Il ne reste que le calcul du complément orthogonal de L. Or, c'est le noyau de l'application linéaire avec la matrice formée par les éléments de L en colonnes. Donc, l'élimination de Gauss-Jordan peut être utilisée encore une fois pour obtenir $L^\perp$.

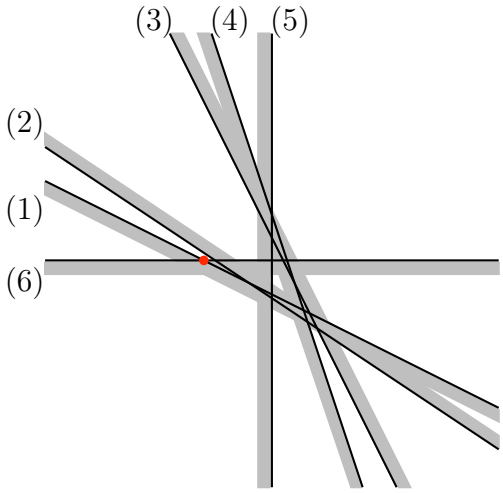
Finalement, on trouve que la valeur de l'objectif du sommet obtenu par l'algorithme est au moins aussi grande que celle du point initial, car on choisit toujours la direction qui donne un produit scalaire positif avec l'objectif.

Exercice 3

Considérer le programme linéaire

$$\begin{array}{rcll} \max & z & & \\ \text{sous contraintes} & x+2y & \leq -3 & (1)\\ & -2x-3y & \leq 5 & (2)\\ & -2x-y+2z & \leq -1 & (3)\\ & 3x+y & \leq 2 & (4)\\ & x & \leq 0 & (5)\\ & y & \leq 0 & (6)\\ & z & \leq 0. & (7) \end{array}$$

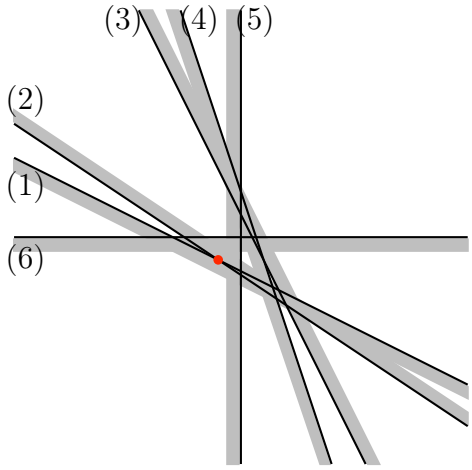
Pour le résoudre, on applique l'algorithme du simplexe et on commence avec le toit défini par les indices des lignes (1), (6) et (7). La situation en dimension 2 grâce à l'hyperplan $z = 0$ est figurée ci-dessous.
Démontrer que l'algorithme du simplexe pourrait ne pas se terminer en donnant une suite périodique de toits qui peuvent être choisis au cours de l'algorithme du simplexe. Expliquer pourquoi votre suite est un choix valable.
(Il est suffisant d'indiquer la suite sur un dessin. Il ne faut pas calculer tous les sommets de façon explicite.).
Indication : Faire rester l'index (7) *dans le toit. Donc on peut se limiter à deux dimensions comme figuré ci-dessous.*



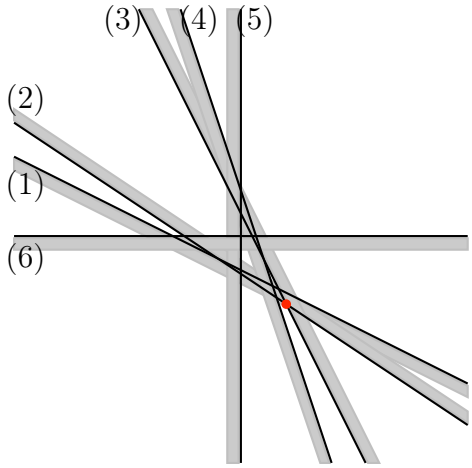
Solution

Rappeler le principe général d'une itération de l'algorithme du simplexe :
À chaque itération, on choisit une contrainte violée par le sommet du toit courant et l'index correspondant entre dans le toit. Une autre contrainte doit sortir du toit de telle sorte que le nouveau sommet est admissible pour le toit précédent.
On va démontrer que l'algorithme peut ne pas terminer sur ce problème en donnant une suite ultimement périodique de toits.

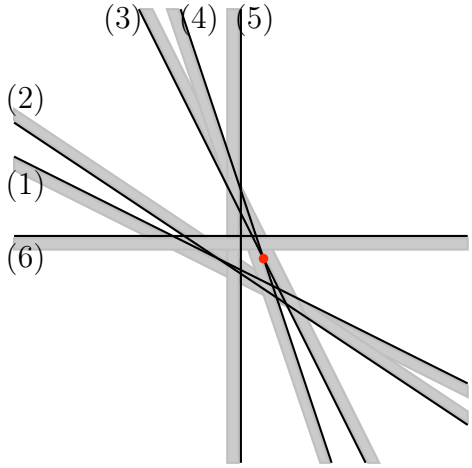
À la première itération, on trouve $\{(1), (2), (7)\}$ comme nouveau toit avec le sommet suivant :



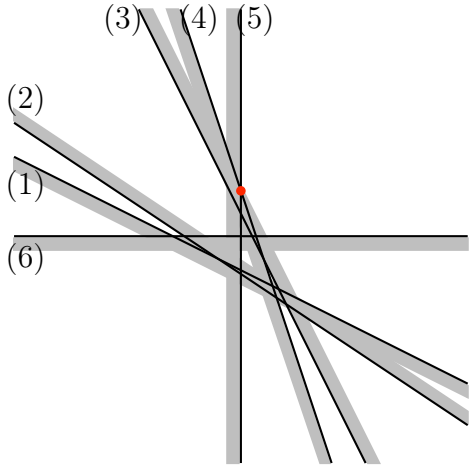
Ensuite, on trouve $\{(2), (3), (7)\}$ et la situation suivante :



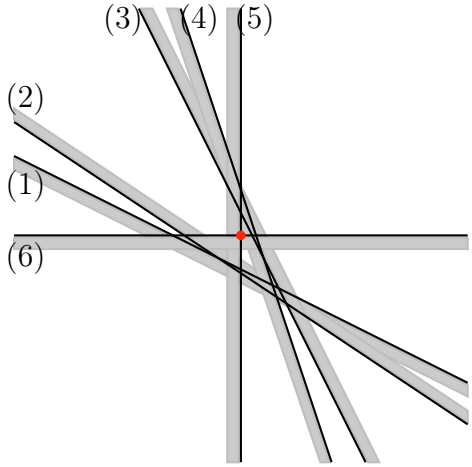
Après, $\{(3), (4), (7)\}$ est le nouveau toit :



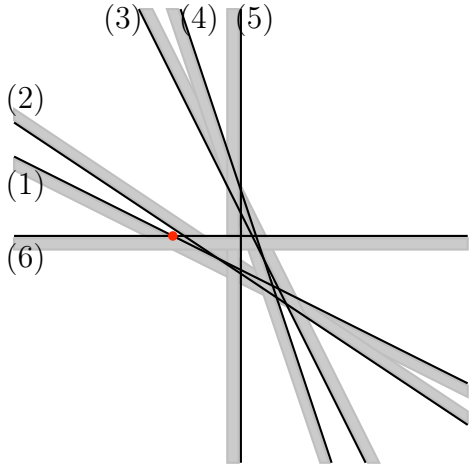
Puis, on a $\{(4), (5), (7)\}$ comme nouveau toit :



On trouve alors le toit $\{(5), (6), (7)\}$ et la situation :



Finalement, on obtient $\{(1), (6), (7)\}$:



Comme le toit obtenu est le toit initial, l'algorithme du simplexe recommence et donc ne termine pas.

Le rendu peut être fait en groupe de trois personnes au plus.