Université d'Alger 1

Département de Mathématiques et Informatique

L3-Systèmes Informatiques

RAPPORT DE RECHERCHE

DÉGÉNÉRESCENCE DE L'ALGORITHME DU SIMPLEXE :
CRITÈRE DE BLAND

FAIT PAR :

LOGOVI Tété Elom Mike Norbert

Groupe 2

21 Avril 2021

Table des matières

Table des figures

Liste des algorithmes

Introduction

L'algorithme du simplexe a été initié depuis les années 1947 par le mathématicien américain Georges Bernard Dantzig.En effet, cet algorithme a conduit à des applications scientifiques et techniques à grande échelle pour des problèmes importants de logistique, d'ordonnancement et d'optimisation de réseaux, ainsi qu'à l'utilisation des ordinateurs pour exploiter efficacement la théorie mathématique.Il a fait également ses preuves en économie d'entreprises pour la gestion des ressources matérielles ou humaines pour maximiser les profits.En pratique, l'algorithme du simplexe, pour pouvoir être exécuté, nécessite que le programme linéaire soit écrit sous forme canonique par rapport à une base réalisable.Un programme linéaire ne possédant pas de base réalisable évidente, ne peut pas être optimisé via la méthode du simplexe standard.On utilise dans ces cas de figure, d'autres méthodes telles que : *la méthode des deux phases* ou *la méthode des pénalités*.L'algorithme du simplexe, quand il est applicable, est une boucle qui suit des étapes jusqu'à une certaine condition d'arrêt.Il arrive parfois que sous certaines contraintes, l'algorithme du simplexe boucle à l'infini sans pour autant optimiser la fonction objetif.C'est ainsi qu'on parle de dégénérescence. Plusieurs solutions existent pour éviter le problème de dégénerescence de l'algorithme du simplexe notamment : la méthode inductive de Dantzig, la règle de Wolfe, la règle de Krishna, **le critère de Bland**.Dans ce rapport, nous allons de prime abord expliquer correctement le problème de dégénérescence de l'algorithme du simplexe, ensuite on exposera le critère de Bland puis finalement on illustrera la solution de Bland via quelques exemples exhaustifs.

Chapitre 1

Dégénérescence de l'algorithme du simplexe

1.1 L'algorithme du simplexe

Voici de manière brève comment l'algorithme fonctionne :

0-Mettre le programme sous forme canonique par rapport à une base réalisable

1-Former le tableau du simplexe

2-Bien choisir la variable entrante

3-Bien choisir la variable sortante

4-Échelonner le programme linéaire autour d'un pivot(pivot de GAUSS)

5-Itérer le processus jusqu'à l'optimum

Soit un programme linéaire (PL) écrit sous forme canonique par rapport à une base réalisable J.

$$(PL)\begin{cases} AX = b \\ CX = maxZ(X) \end{cases} \quad avec \quad \begin{cases} X \text{ un vecteur colonne de } n \text{ lignes} \\ C \text{ un vecteur ligne de } n \text{ colonnes} \\ b \text{ un vecteur colonne de } m \text{ lignes} \\ A \text{ une matrice de } m \text{ lignes et } n \text{ colonnes} \end{cases}$$

$$\begin{cases} X \geq 0 \\ b \geq 0 \\ A^J \text{ matrice identité à une permutation près et inversible} \end{cases}$$

Algorithme 1 Algorithme du simplexe

Étape 1 : Former le tableau du simplexe

Il existe plusieurs manières pour former ce tableau.Voici une méthode :

1. L'en-tête est constituée d'une première colonne contenant J, suivie de n colonnes contenant successivement les variables x_i pour $i = 1 \ldots n$ (variables de base et variables hors base), puis une dernière colonne contenant b.
2. On crée autant de lignes que d'équations dans le programme linéaire écrit sous forme standard ou sous forme canonique par rapport à une base réalisable. Chaque ligne étant associé à une équation.
3. Pour chaque ligne $j(j = 1 \ldots m)$, la première colonne contient le nom de la variable de base se trouvant dans l'équation correspondante. Les autres colonnes contiennent successivement les A_j^i pour $i = 1 \ldots n$, puis on rajoute en dernière colonne b_j.
4. On rajoute finalement une dernière ligne dont la première colonne contient Z, suivie de n colonnes contenant successivement les C^i pour $i = 1 \ldots n$.

Étape 2 : Bien choisir la variable entrante

— **Pour un problème de maximisation**
  Choisir **e** tel que $C^e = max(C^j > 0 \text{ tel que } j \notin J)$

— **Pour un problème de minimisation**
  Choisir **e** tel que $C^e = min(C^j < 0 \text{ tel que } j \notin J)$

Si plusieurs valeurs sont possibles pour **e**, alors on choisit une valeur au hasard et on travaille avec elle. La variable x_e rentre ainsi en base.
Si aucune valeur n'est possible pour **e**, le programme est optimal. **Terminer l'algorithme**.

Étape 3 : Bien choisir la variable sortante

$S \leftarrow \{j \text{ tel que } A_j^e > 0 \text{ pour } j = 1 \ldots m\}$
On trouve ensuite **s** $\in S$ tel que le rapport $b_s/A_s^e = min(\{b_j/A_j^e, j = 1 \ldots m\})$

— Si aucune valeur n'est possible pour **s**, il n'y a pas de solution optimale car Z est non bornée. **Terminer l'algorithme**.

— Si plusieurs valeurs sont possibles pour **s**, survient alors le problème de **dégénérescence de l'algorithme de simplexe** (1.2 détails dans la section suivante). On choisit une valeur au hasard parmi celles qui sont possibles pour **s** (on peut choisir cette valeur selon 2 **le critère de Bland**).La variable se trouvant dans la ligne **s** du tableau du simplexe sortira de la base.

— Si une seule valeur est possible pour s, **pas de dégénérescence**, on choisit s puis on continue l'algorithme. La variable se trouvant dans la ligne **s** du tableau du simplexe sortira de la base.

Étape 4 : Pivot de GAUSS

Après avoir mis à jour la base J en sortant la variable de base se trouvant dans la ligne **s** du tableau et en faisant entrer x_e , on applique la procédure d' élimination de Gauss Jordan autour du pivot A_s^e et on divise la ligne **s** par A_s^e. Ce qui nous permet d'écrire le programme linéaire sous forme canonique par rapport à la nouvelle base.

Étape 5 : Retourner à l'étape 2

1.2 Dégénérescence

En supposant qu'on utilise l'algorithme du simplexe standard uniquement lorsqu' on a une base réalisable évidente, ce dernier précédemment illustré, peut avoir des solutions dégénérées dans certains cas.On parle de **dégénérescence de l'algorithme du simplexe** lorsque lors du déroulement de l'algorithme du simplexe, au cours d'une itération, une ou plusieurs variables de base se retrouvent avec une valeur nulle.

Dans un algorithme de simplexe, si à une itération **i**, lors de l'étape 3 (1 voir étape), pour choisir la variable de base sortante, le rapport b_j/A_j^e est minimal pour au moins deux valeurs de $j(j = 1 \ldots m)$,alors dans ce cas, après le choix (au hasard) d'une variable sortante et échelonnement par la méthode du pivot de Gauss, on se retrouvera dans un problème de dégénérescence de l'algorithme du simplexe dans l'itération **i+1**.

1.2.1 Exemple

Soit le programme linéaire suivant :

$$(\mathrm{PL1}) \begin{cases} x_1 + 4x_2 \leq 8 \\ x_1 + 2x_2 \leq 4 \\ 3x_1 + 9x_2 = maxZ(X) \\ x_1, x_2 \geq 0 \end{cases}$$

On commence par écrire le programme sous forme canonique par rapport à une base réalisable évidente. Pour cela on va rajouter deux variables d'écart x_3 et x_4.
Le programme linéaire devient :

$$(\mathrm{PL1}) \begin{cases} x_1 + 4x_2 + x_3 = 8 \\ x_1 + 2x_2 + x_4 = 4 \\ 3x_1 + 9x_2 = maxZ(X) \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

On forme le tableau du simplexe comme expliqué à l'étape 1 de l'algorithme du simplexe(itération 0)

J	x_1	x_2	x_3	x_4	b
x_3	1	4	1	0	8
x_4	1	2	0	1	4
Z	3	9	0	0	0

Comme on peut le voir, en suivant l'algorithme du simplexe, x_2 est la variable entrante puisque $x_e = x_2$ car e=2 étant donné que $max(\{C^1 = 3, C^2 = 9\}) = 9$ par conséquent $C^e = C^2$. Pour la variable sortante par contre, on peut remarquer qu'il y a deux cas : x_3 et x_4 sont de potentielles variables sortantes car pour notre exemple, $S = \{1, 2\}$ car $A_1^2 = 4 > 0$ et $A_2^2 = 2 > 0.s = \{1, 2\}$ car $b_1/A_1^2 = 8/4 = 2$ et $b_2/A_2^2 = 4/2 = 2$.
Il y a donc plusieurs variables sortantes possibles.

— Si on choisit x_3 comme variable sortante, on obtient successivement les itérations suivantes :

1. Itération 1

J	x_1	x_2	x_3	x_4	b
x_2	$\frac{1}{4}$	1	$\frac{1}{4}$	0	2
x_4	$\frac{1}{2}$	0	-$\frac{1}{2}$	1	0
Z	$\frac{3}{4}$	0	-$\frac{9}{4}$	0	-18

2. Itération 2

J	x_1	x_2	x_3	x_4	b
x_2	0	1	$\frac{1}{2}$	-$\frac{1}{2}$	2
x_1	1	0	-1	2	0
Z	0	0	-$\frac{3}{2}$	$-\frac{3}{2}$	-18

   — On peut remarquer que pour chaque itération : itération1 et itération 2, il y a au moins une variable de base qui prend la valeur nulle, il s'agit de x_4 dans l'itération 1 et de x_1 dans l'itération 2.

   — On remarque aussi que la valeur optimale (18) ne change pas.On a fait une itération inutilement sans réellement optimiser la fonction objectif. On dit que **la solution est dégénérée**.

— Si on choisit par contre x_4 comme variable sortante au début, on arrive plus vite à l'optimum. En voici la preuve :

J	x_1	x_2	x_3	x_4	b
x_3	-1	0	1	-2	0
x_2	$\frac{1}{2}$	1	0	$\frac{1}{2}$	2
Z	-$\frac{3}{2}$	0	0	-$\frac{9}{2}$	-18

Après une seule itération on est arrivé au résultat. Et on a pas cyclé inutilement. Dans certains cas, on peut itérer un nombre infini de fois sans optimiser la fonction objectif.

— Une autre solution serait de choisir x_1 comme variable entrante et x_4 comme variable sortante au départ(on ne suit pas la règle générale de l'algorithme du simplexe pour la variable entrante), on obtient alors :

1. Itération 0

J	x_1	x_2	x_3	x_4	b
x_3	1	4	1	0	8
x_4	1	2	0	1	4
Z	3	9	0	0	0

2. Itération 1

J	x_1	x_2	x_3	x_4	b
x_3	0	2	1	-1	4
x_1	1	2	0	1	4
Z	0	3	0	-3	-12

3. Itération 2(x_2 entre en base et x_1 sort de la base)

J	x_1	x_2	x_3	x_4	b
x_3	-1	0	1	-2	0
x_2	$\frac{1}{2}$	1	0	$\frac{1}{2}$	2
Z	-$\frac{3}{2}$	0	0	-$\frac{9}{2}$	-18

On peut remarquer que cette fois-ci, on assure qu'à chaque itération, la fonction objectif augmente et on arrive à l'optimum.

$$Z^* = 18 \text{ et } X = \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \end{pmatrix}$$

1.2.2 Illustration graphique

Voici la représentation graphique du programme linéaire précédemment résolu :

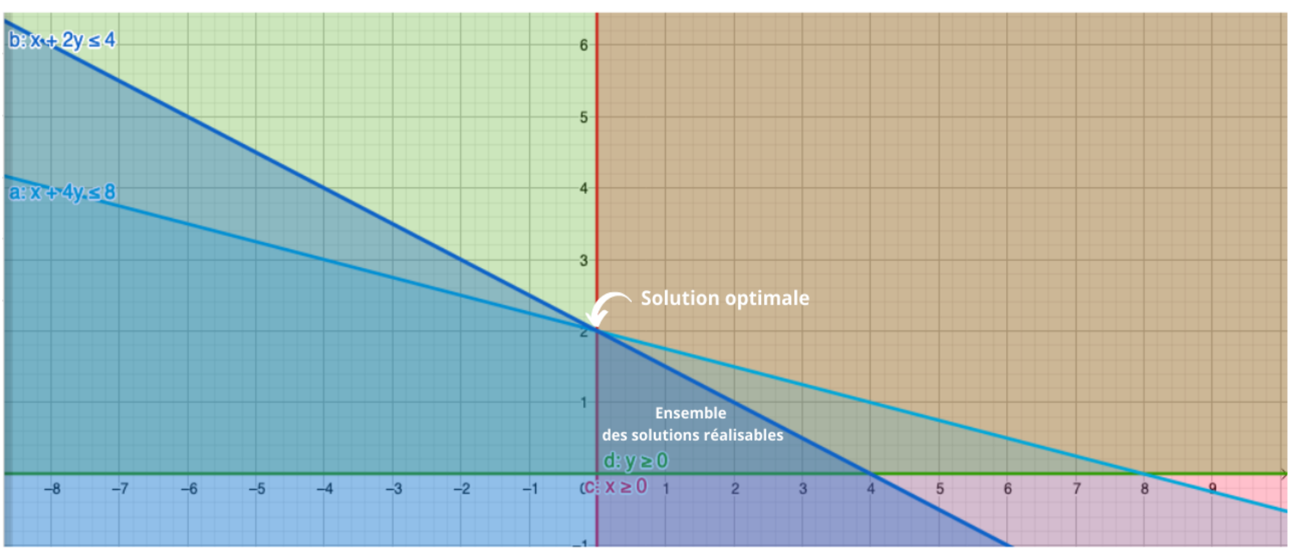

Figure 1.1 – Résolution graphique de PL1

On peut remarquer que le point A$\begin{pmatrix}0\\2\end{pmatrix}$ est le point d'intersection de 3 droites à savoir : (Δ_1) :$x_1 + 4x_2 = 8$,(Δ_2) :$x_1 + 2x_2 = 4$,(Δ_3) :$x_1 = 0$. En réalité, deux droites suffisent amplement pour trouver leur intersection, mais ici plus de deux droites se coupent en un même point, ici le point A. Cela veut dire qu' en prenant deux droites parmi les 3 droites, on retrouve le point A, ce qui veut dire qu'il existe $C_3^2 = 3$ manières de retrouver le point A qui est la solution optimale. On pourrait cycler probablement de trois manières dans le tableau du simplexe et au moins une variable de base aura la valeur 0. Dans ce genre de cas, **la solution est dégénérée**.
Grâce au **critère de Bland**, on peut éviter de cycler inutilement dans l'algorithme du simplexe.

Chapitre 2

Critère de Bland

2.1 Explication

Lorsque plusieurs possibilités s'offrent à nous lors du choix de la variable sortante, on a vu que la dégénérescence de l'algorithme du simplexe se produit.

Toutefois, comment bien choisir la variable sortante parmi toutes les possibilités afin de réduire les itérations et éviter le cyclage ? Il suffit d'utiliser **le critère de Bland**.

Le critère de Bland, connu également sous le nom d'algorithme de Bland ou règle anti-cyclage de Bland ou règle du pivot de Bland, est une amélioration algorithmique de la méthode du simplexe pour l'optimisation linéaire.

L'algorithme du simplexe impose de commencer par une solution de base réalisable.On modifie la base afin d'optimiser la fonction objectif et trouver une solution optimale. La fonction objectif augmente à chaque étape, et après un nombre fini d'étapes, une solution optimale est trouvée. Cependant, il existe des exemples de programmes linéaires (comme on l'a vu dans le chapitre précédent), pour lesquels l'algorithme du simplexe change de base de manière cyclique sans optimiser la fonction objectif.

De tels cycles sont évités par le critère de Bland permettant de bien choisir **la variable entrante** et **la variable sortante**.

Le critère de Bland garantit **la non-cyclicité des bases réalisables produites**. Si la fonction objectif est majorée, on peut atteindre la solution optimale en un nombre fini d'itérations.

De plus, à l'exception peut-être de **la règle du choix aléatoire**, le critère de Bland est le plus simple à mettre en œuvre (Dantzig & Thapa,).

2.2 Utilisation

Considérons le programme linéaire comme on l'a vu sous sa forme canonique par rapport à une base réalisable J, dans le chapitre 1 (voir 1.1). À chaque fois que l'on appliquera l'algorithme du simplexe et que l'on se retrouvera dans une situation dans laquelle la fonction objectif n'augmente plus, il faudra suivre cette démarche ci dessous :

Algorithme 2 Critère de Bland

Étape 1 : Bien choisir la variable entrante

- **Pour un problème de maximisation**
  Choisir **e** tel que $C^e = max(C^j > 0 \text{ tel que } j \notin J)$
- **Pour un problème de minimisation**
  Choisir **e** tel que $C^e = min(C^j < 0 \text{ tel que } j \notin J)$

Si plusieurs valeurs sont possibles pour **e** : Choisir parmi elles, la plus petite valeur e_{min}. Ainsi, la variable x_{emin} entre dans la base. Autrement dit, il faut choisir la toute première variable candidate pour entrer en base.
Si aucune valeur n'est possible pour **e**, le programme est optimal.Terminer l'algorithme.

Étape 2 : Bien choisir la variable sortante

Parmi toutes les variables de base sortantes x_i qui sont candidates(selon l'étape 3 de l' algorithme du simplexe(1)), choisir la variable de base avec le plus petit indice **i** comme étant **la variable sortante**.

La dernière solution proposée (1.2.1) pour (PL1) vu au chapitre 1, illustre bien le critère de Bland pour la variable entrante et la variable sortante.

2.2.1 Exemple 1

- Considérons cet exemple en appliquant le critère de Bland

$$(\text{PL2})\begin{cases} x_1 + 7x_2 \leq 1 \\ 3x_1 + 14x_2 \leq 3 \\ 3x_1 + 3x_2 = maxZ(X) \\ x_1, x_2 \geq 0 \end{cases}$$

- Passons sous forme standard

$$(\text{PL2})\begin{cases} x_1 + 7x_2 + x_3 = 1 \\ 3x_1 + 14x_2 + x_4 = 3 \\ 3x_1 + 3x_2 = maxZ(X) \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

- Tableau du simplexe :

J	x_1	x_2	x_3	x_4	b
x_3	1	7	1	0	1
x_4	3	14	0	1	3
Z	3	3	0	0	0

— Il y a deux variables potentielles pour entrer en base soit x_1 et x_2 on choisit x_1 selon le critère de Bland car il a le plus petit indice qui est 1

— Il y a deux variables potentielles pour sortir de la base soit x_3 et x_4 on choisit x_3 selon le critère de Bland car il a le plus petit indice qui est 3

— Tableau du simplexe(avec le pivot)

J	x_1	x_2	x_3	x_4	b
x_3	1	7	1	0	1
x_4	3	14	0	1	3
Z	3	3	0	0	0

— Après la première itération

J	x_1	x_2	x_3	x_4	b
x_1	1	7	1	0	1
x_4	0	-7	-3	1	0
Z	0	-18	-3	0	-3

On est arrivé à l'optimum : X= $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ et $Z^* = 3$

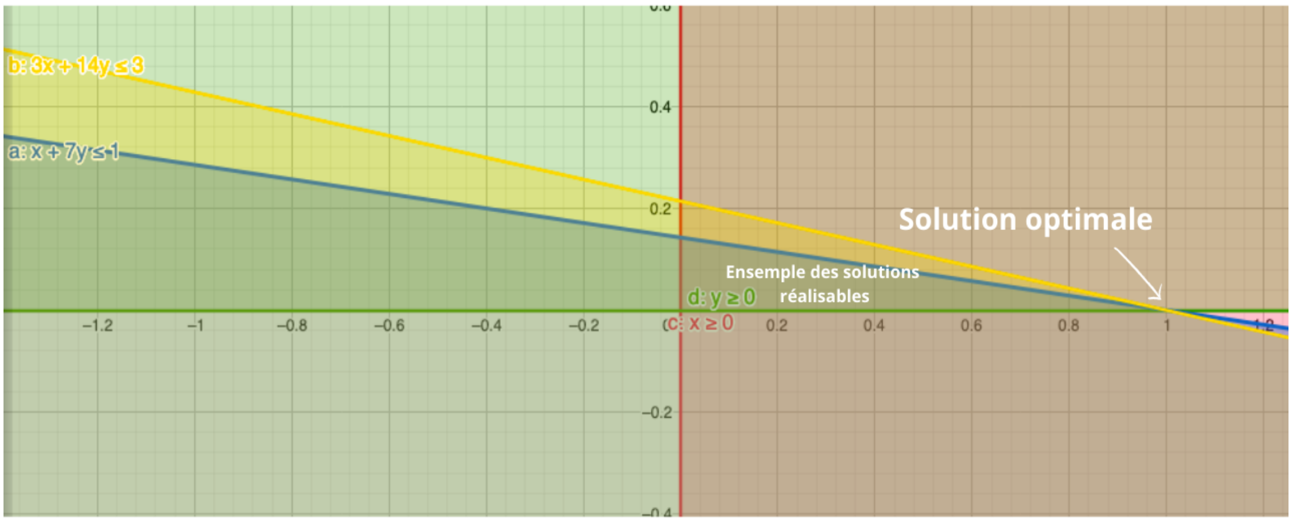

FIGURE 2.1 – Résolution graphique de PL2

2.2.2 Exemple 2

— Considérons ce nouvel exemple en appliquant le critère de Bland

$$(PL3)\begin{cases} x_1 + 2x_2 + 3x_3 \leq 4 \\ 2x_1 + 5x_2 + 6x_3 \leq 8 \\ 3x_1 + 4x_2 + 9x_3 \leq 12 \\ 3x_1 - 5x_2 + 4x_3 = maxZ(X) \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

— Passons sous forme standard

$$(PL3)\begin{cases} x_1 + 2x_2 + 3x_3 + x_4 = 4 \\ 2x_1 + 5x_2 + 6x_3 + x_5 = 8 \\ 3x_1 + 4x_2 + 9x_3 + x_6 = 12 \\ 3x_1 - 5x_2 + 4x_3 = maxZ(X) \\ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{cases}$$

— Tableau du simplexe :

J	x_1	x_2	x_3	x_4	x_5	x_6	b
x_4	1	2	3	1	0	0	4
x_5	2	5	6	0	1	0	8
x_6	3	4	9	0	0	1	12
Z	3	-5	4	0	0	0	0

- On choisit la variable x_1 comme variable entrante
- Il y a trois variables potentielles pour sortir de la base soit x_4, x_5,x_6 et on choisit x_4 selon le critère de Bland car il a le plus petit indice qui est 4
- Tableau du simplexe(avec le pivot)

J	x_1	x_2	x_3	x_4	x_5	x_6	b
x_4	1	2	3	1	0	0	4
x_5	2	5	6	0	1	0	8
x_6	3	4	9	0	0	1	12
Z	3	-5	4	0	0	0	0

- Après la première itération

J	x_1	x_2	x_3	x_4	x_5	x_6	b
x_1	1	2	3	1	0	0	4
x_5	0	1	0	-2	1	0	0
x_6	0	-2	0	-3	0	1	0
Z	0	-11	-5	-3	0	0	-12

On est arrivé à l'optimum : X= $\begin{pmatrix} 4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ et $Z^* = 12$

2.3 Implémentation

Pour implémenter le critère de Bland, nous allons utiliser le langage de programmation Python.

Voir les documents correspondant comme ressources attachées.

Conclusion

Somme toute, dans ce rapport nous avons parlé de l'algorithme de simplexe.Nous avons vu que sous certaines contraintes, on peut avoir des problèmes de dégénérescence. Cela peut donc entraîner un cyclage ou encore une boucle infinie dans l'algorithme du simplexe sans pour autant optimiser la fonction objectif.On a même vu que lorsqu'une variable de base s'annule ou que lorsque plusieurs droites de contraintes se coupent en un certains point, cela est un signe de dégénérescence.Plusieurs solutions sont possibles pour éviter la dégénérescence de l'algorithme du simplexe dont le critère de Bland. Ce critère simple est réputé pour être le plus facile à utiliser en cas de dégénérescence. Bien qu'il garantisse l'anticyclage, comment pourrait-on améliorer le critère de Bland pour aboutir encore plus rapidement à l'optimum lorsque plusieurs contraintes et plusieurs variables sont présentes dans le programme linéaire ?

Références

Dantzig, G. B., & Thapa, M. N. (2006).