RAIRO. RECHERCHE OPÉRATIONNELLE

J. SISKOS

Le traitement des solutions quasi optimales en programmation linéaire continue : une synthèse

RAIRO. Recherche opérationnelle, tome 18, n° 4 (1984), p. 381-401

<http://www.numdam.org/item?id=RO_1984__18_4_381_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 18, n° 4, novembre 1984, p. 381 à 401)

LE TRAITEMENT DES SOLUTIONS QUASI OPTIMALES EN PROGRAMMATION LINÉAIRE CONTINUE : UNE SYNTHÈSE (*)

par J. SISKOS (1)

Résumé. — *Le but de cet article est triple : 1° poser le problème de solutions optimales multiples en programmation linéaire en le formulant dans un contexte plus général, celui de la recherche des solutions quasi-optimales; 2° décrire deux approches représentatives et complémentaires et montrer leurs limites après les avoir illustrées sur le même exemple numérique; 3° présenter finalement deux heuristiques d'ordre pratique qui évitent les complications algorithmiques des méthodes exactes.*

Mots clés : Programmation linéaire; Solutions quasi optimales; Algorithme du simplexe.

Abstract. — *The aim of this paper is: 1. to consider and to discuss the problem of multiple optimal solutions in linear programming within the frame-work of general near-optimality analysis; 2. to describe two representative and complementary approaches and to illustrate them on the same numerical example; 3. and finally to present briefly some practical heuristics which avoid the algorithmic complexity of exact methods.*

Keywords: Linear programming; Near-optimality analysis; Simplex algorithm.

1. INTRODUCTION

Le problème de solutions optimales multiples en programmation linéaire (PL) est très fréquemment rencontré tant dans les applications pratiques de la PL à des problèmes de gestion que dans des problèmes théoriques. Pour ce qui concerne ces derniers, les domaines les plus affectés sont :

— la théorie des jeux à deux personnes et à somme nulle [18] où il existe très souvent non pas une seule stratégie optimale mais plusieurs;

— le « goal programming », domaine nouveau de la PL [13] caractérisé par la recherche de solutions qui doivent être les plus compatibles possible avec un système de contraintes (goals);

(*) Reçu février 1983.
(1) Technical University of Crete, Iroon Polytechniou 37, Chania, Crète, Grèce. L'auteur a écrit cet article lorsqu'il travaillait au Lamsade, Université de Paris-Dauphine.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/84/04 381 21/$ 4.10

– la régression ordinale du type quantitatif et/ou qualitatif [22, 23] où le codage des variables (*cf.* fonction d'utilité) que l'on cherche à ajuster est loin d'être unique;

– la théorie des graphes (le problème de transport par exemple), bien que bon nombre de ses problèmes se formalisent en termes de programmes linéaires en nombres entiers;

– . . .

Le but de cet article est donc triple : 1° poser clairement le problème de solutions optimales multiples en PL en le formulant dans un contexte plus général, celui de la recherche des solutions quasi optimales; 2° décrire de façon originale deux approches représentatives et complémentaires et montrer leurs limites après les avoir illustrées sur le même exemple numérique; 3° présenter succinctement deux heuristiques d'ordre pratique qui évitent les complications algorithmiques des méthodes exactes.

Le contenu de cet article demande des connaissances préalables en PL (algorithme du simplexe). La matière dans [7] ou [8] serait par exemple suffisante.

2. SOLUTIONS OPTIMALES ET QUASI OPTIMALES D'UN PROGRAMME LINÉAIRE

Considérons le programme linéaire continu suivant, écrit sous sa forme canonique :

$$\text{PL1} \begin{cases} [\max]\ F = \mathbf{c}^t\, \mathbf{x}, \\ \mathscr{A}\, \mathbf{x} \leqq \mathbf{b}, \\ \mathbf{x} \geqq 0, \end{cases}$$

avec $\mathscr{A}$, $\mathbf{x}$, $\mathbf{b}$ et $\mathbf{c}$ des matrices respectivement de taille $m \times n$, $n \times 1$, $m \times 1$ et $n \times 1$.

Le problème de *solutions optimales multiples*, se présente à l'issue de l'algorithme du simplexe (tableau-simplexe optimal) lorsqu'il existe au moins un profit marginal associé à une variable hors base qui est nul, c'est-à-dire :

$$\Delta_j = c_j - \sum_{i=1}^{m} c_i\, a_{ij} = 0, \tag{1}$$

i désignant l'indice d'une variable de base; j, l'indice d'une variable hors base. Géométriquement, cela signifie que l'hyperplan représentant la fonction-objectif (sous l'hypothèse qu'il y a plus que trois variables) est parallèle à une arête ou une face de l'hyperpolyèdre des contraintes. Un problème crucial

à résoudre est donc la recherche de toute autre solution optimale de ce programme linéaire.

Dans ce cas particulier, il est bien connu que l'introduction dans la base-simplexe d'une variable j vérifiant (1) entraînera une nouvelle solution optimale (avec $F=F^*$, valeur optimale de la fonction objectif). Pourtant, on ne peut pas prescrire d'avance le nombre et le type de solutions multiples auxquelles on peut ainsi parvenir. La poursuite arbitraire de l'algorithme du simplexe au-delà de l'optimum peut générer d'autres solutions optimales mais cette démarche ne peut, en aucun cas, être considérée comme une méthode d'exploration systématique de toutes les solutions optimales.

Depuis l'apparition de l'algorithme du simplexe en 1948 par Dantzig, il existe une bibliographie très importante sur la PL mais très peu d'auteurs, à notre connaissance, ont essayé de faire face sérieusement à ce problème. D'autre part, les livres de PL qui développent le sujet (*cf.* [3, 12, 25], . . .) ne considèrent pas toutes les approches possibles du problème et, parfois, se limitent à des méthodes qui ont été prouvées inefficaces (exemple : la méthode de Tarry suggérée dans [2] ou [3]; *cf.* section 4 de cet article).

Dans un premier temps, il est important d'observer que l'ensemble des solutions optimales (assurant que $F=F^*$) est donné par le polyèdre Ph1 ci-dessous :

$$\text{Ph1} \begin{cases} \mathscr{A}\,\mathbf{x} \leqq \mathbf{b}, \\ \mathbf{c}^t\,\mathbf{x} = F^*, \\ \mathbf{x} \geqq 0. \end{cases}$$

En cas de solution optimale unique, Ph1 ne contient qu'un seul point. Dans le cas contraire, il existe une infinité de telles solutions et il suffit de déterminer tous les sommets de Ph1, toute autre solution étant une combinaison convexe des solutions représentées par les sommets.

Dans cet article, on se place dans un contexte plus général en s'interrogeant sur l'aspect parfois arbitraire, en pratique, de la solution qualifiée d'optimale. Il suffirait en effet que l'on ait commis une légère erreur dans l'estimation des coefficients d'une contrainte pour qu'une solution, à la limite du respect de cette contrainte, soit en réalité dangereuse ou irréalisable. Même sans cela, le modélisateur ne sera pas toujours intéressé uniquement par la (les) solution(s) optimale(s). Il se peut qu'il désire tenir compte de *solutions quasi optimales.* En effet, il aura souvent été difficile de formuler, de façon précise, les contraintes du modèle qui pourront comporter une plus ou moins grande part d'arbitraire. Par ailleurs, la fonction-objectif à maximiser ou minimiser ne tient compte que d'un seul critère d'optimalité, ce qui peut négliger certains

autres critères d'évaluation des solutions que l'on n'aura pas su ou pu modéliser. D'où l'intérêt d'examiner d'autres solutions que l'optimum qui soient suffisamment proches pour que l'on ne perde pas trop en satisfaction du point de vue de l'objectif du modèle mais qui puissent être considérées comme meilleures ou aussi bonnes d'un certain point de vue non intégré dans la modélisation.

En dehors de l'*analyse postoptimale* classique (*cf.* Gal [10]), Van de Panne [25] propose une analyse particulière qui s'adapte aux situations réalistes évoquées plus haut. En effet, après avoir obtenu l'optimum d'un programme linéaire, il recherche toutes les solutions $\mathbf{x}$ dont la valeur $F=\mathbf{c}^t\mathbf{x}$ ne diffère de la valeur optimale F^* que d'une quantité qui, au pire, sera égale à $k>0$; k représente donc le montant que le décideur est prêt à concéder à titre de perte maximale autorisée sur la fonction-objectif. Comme pour le polyèdre Ph1, le traitement de ce type de solutions s'explicite par la recherche des sommets de l'hyperpolyèdre Ph2 ci-dessous :

$$\text{Ph2} \begin{cases} \mathscr{A}\,\mathbf{x} \leqq \mathbf{b}, \\ \mathbf{c}^t\mathbf{x} \geq F^*-k, \\ \mathbf{x} \geqq 0. \end{cases}$$

La recherche de solutions optimales se présente ici comme un cas particulier en posant $k=0$ (Ph2 $\equiv$ Ph1 car F^* est la valeur maximale de F).

Dans la suite, nous présentons deux approches consistant à rechercher systématiquement les sommets de l'hyperpolyèdre Ph2. La première, due à Van de Panne (section 3) est spécifique à l'analyse des solutions quasi optimales car elle génère les sommets $\mathbf{x}$ dans un ordre hiérarchique des valeurs de F, à savoir dans l'ordre $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^i, \mathbf{x}^{i+1}, \ldots$ avec $\mathbf{c}^t\mathbf{x}^1 \geqq \mathbf{c}^t\mathbf{x}^2 \geqq \ldots \geqq \mathbf{c}^t\mathbf{x}^i \geqq \mathbf{c}^t\mathbf{x}^{i+1} \geqq \ldots$ La deuxième approche consiste en un algorithme de recherche aveugle des sommets d'un ensemble polyédrique (section 4).

Il est important de souligner ici que la mise au point de telle ou telle méthode présuppose que nous avons les moyens d'estimer d'avance le nombre de sommets que contient l'hyperpolyèdre Ph2. Klee [14] donne une borne, soit $\bar{r}$, assez fine pour un polyèdre défini par un système de m équations à l variables avec $l>m$:

$$\bar{r} = \begin{cases} \dfrac{2l}{m+l}\,\mathfrak{C}^m_{1/2\,(m+l)} & \text{si } l-m \text{ pair}, \quad (2) \\ 2\,\mathfrak{C}^m_{1/2\,(m+l-1)} & \text{si } l-m \text{ impair}. \quad (3) \end{cases}$$

Les résultats sont assez décourageants. La valeur de la borne $\bar{r}$ s'accroît avec une vitesse astronomique : pour ($m=10$, $l=16$), $\bar{r}=352$; pour ($m=10$, $l=17$), $\bar{r}=572$ et ce n'est là que des programmes linéaires minuscules. Matheiss et Rubin [17] donnent des estimations statistiques du nombre de sommets d'un hyperpolyèdre basées sur des échantillons générés aléatoirement par ordinateur.

3. LA MÉTHODE DU SIMPLEXE INVERSE

La méthode

Van de Panne ([25], p. 202-231) a remarqué le fait que chaque itération de l'algorithme du simplexe en PL est réversible par un pivotage inversant le rôle de la variable qui entre dans la base avec celui de celle qui en sort. Un pivotage inverse fait décroître l'objectif F (à maximiser) d'une quantité égale à celle de l'accroissement dû au pivotage normal correspondant.

Considérons en effet un pivotage-simplexe normal introduisant dans la base la variable x_j de profit marginal $\Delta_j > 0$. La variable qui sortira de la base sera x_r déterminée par la relation (4) :

$$\frac{b_r}{a_{rj}} = \min_i \left\{ \frac{b_i}{a_{ij}} \text{ avec } a_{ij} > 0 \right\}, \tag{4}$$

i désignant l'indice de variable de base. Cette opération provoquera une augmentation de l'objectif F de $\Delta_j b_r/a_{rj} > 0$. Introduisons maintenant dans la base, en un sens inverse, la variable x_r; à x_r est attaché maintenant un profit marginal négatif : $-\Delta_j/a_{rj} < 0$. L'opération fera donc décroître F de $\Delta_j b_r/a_{rj}$. D'autre part, la variable qui quittera la base sera à son tour x_j du fait qu'elle correspond au ratio (4) minimal $(b_r/a_{rj})/(1/a_{rj}) = b_r$.

La méthode du simplexe inverse est une méthode arborescente qui consiste à rechercher des solutions quasi optimales en les rangeant selon les valeurs décroissantes de F. Elle cherche donc à effectuer des pivotages appropriés inverses respectant cette hiérarchie en allant progressivement de $F=F^*$ à $F=F^*-k$. Pour mieux tenir compte de cette décroissance progressive de F, on introduit une nouvelle variable Y selon la formule (5) :

$$Y = \mathbf{c}^t \mathbf{x} - (F^* - k) \geqq 0, \tag{5}$$

mesurant la déviation de $F=\mathbf{c}^t\mathbf{x}$ de la valeur minimale autorisée F^*-k et variant, bien sûr, de 0 à k. La variable Y est introduite dans la base optimale (initialisée à k) en tant que variable additionnelle. Le tableau optimal aura

donc une ligne supplémentaire composée des coûts marginaux Δ_j ($-$profits marginaux) à l'optimum.

La méthode se déroule en deux phases : en première phase, elle calcule les solutions pour lesquelles $0 < Y \leqq k$; puis, en seconde phase, les solutions les plus écartées de l'optimum avec $Y=0$. Chaque solution générée par cette méthode est munie d'une valeur Y inférieure ou égale à celle de la solution précédemment calculée.

L'arborescence est développée de la façon suivante :

Phase I

Étape 0 (Initialisation) : Mise au point du tableau-simplexe optimal (noté 0) avec une ligne supplémentaire composée des coûts marginaux $\Delta_j^0 > 0$ à l'optimum; $Y=k$ (noté 0) (borne du sommet initial $\kappa_0=k$). Le cas $\Delta_j=0$ n'est pas pris en compte.

Étape 1 : Considérer le tableau-simplexe s. Créer une branche pour chaque variable x_j hors base avec $\Delta_j^s > 0$. Déterminer les pivots a_{rj} (pour chaque j avec $\Delta_j^s > 0$) par la relation :

$$\min_i \{b_i^s/a_{ij}^s;\ a_{ij}^s > 0\} = b_r^s/a_{rj}^s. \tag{6}$$

Étape 2 : Associer à chaque nouveau sommet la borne :

$$\kappa_{sj} = \kappa_s - \Delta_j^s\,(b_r^s/a_{rj}^s). \tag{7}$$

Cette expression mesure la quantité de décroissance de la valeur κ_s par l'introduction de la variable x_j dans la base s.

Étape 3 : Déterminer la variable x_j maximisant la quantité (8) :

$$\kappa_{s+1} = \max_j \{0,\ \kappa_{0j},\ \kappa_{1j},\ \ldots,\ \kappa_{sj}/\kappa_{ij} > 0\} \tag{8}$$

et enlever cette valeur de l'ensemble $\kappa_{ij} > 0$. Si $\kappa_{s+1}=0$, aller en phase II. Sinon :

Étape 4 : Effectuer le pivotage ($x_j \uparrow x_r \downarrow$) au tableau-simplexe correspondant au sommet ayant la borne maximale (8). Aller à l'étape 1 avec $s=s+1$.

Phase II

Cette phase ne calcule que des solutions pour lesquelles $Y=0$ car Y sortira obligatoirement de la base, tous les κ_{ij} étant négatifs. A chaque sommet de

l'arbre (tableau s avec $\Delta_j^s > 0$ et $\kappa_{sj} < 0$), on calcule la solution correspondante en évitant évidemment de reproduire une solution déjà calculée. Il suffit donc de poser :

$$x_j = \kappa_s / \Delta_j^s, \tag{9}$$

$$x_i = b_i^s - a_{ij}^s x_j, \qquad i \neq j. \tag{10}$$

Un exemple numérique ([2])

Considérons le programme linéaire standard (pris dans [25]) :

$$\text{PL2} \begin{cases} [\max]\ F = 3x_1 + 4x_2 + 5x_3 + 6x_4 + 0x_5 + 0x_6, \\ x_1 + x_2 + x_3 + x_4 + x_5 = 18, \\ 2x_3 + 3x_4 + x_6 = 6, \\ x_i \geqq 0, \qquad i = 1, 6. \end{cases}$$

$s = 0$

Le tableau initial du simplexe et le tableau optimal ($F^* = 76$) sont respectivement notés par 00 et 0 dans le tableau I. Prenons $k = 20$; ceci équivaut à l'introduction de la contrainte additionnelle $3x_1 + 4x_2 + 5x_3 + 6x_4 \geqq 56$.

Étape 0 : L'algorithme commence par le remplacement, dans le tableau n° 0, de $F^* = 76$ par $Y = 20$ ($\kappa_0 = 20$); sur cette ligne, les profits marginaux sont transformés en coûts marginaux, en inversant les signes.

Étape 1 : Les variables qui peuvent entrer dans la base sont celles qui correspondent à des coûts marginaux positifs ([3]) (première ligne du tableau-simplexe), à savoir x_1, x_5, x_3, x_6. Les pivots sont déterminés par les relations (6); ils correspondent aux coefficients encadrés de chaque tableau-simplexe.

Étape 2 : Nous calculons les bornes κ_{0j} par les formules (7) :

$$\kappa_{01} = 20 - 1 \times \frac{16}{1} = 4, \qquad \kappa_{05} = -44, \qquad \kappa_{03} = 19, \qquad \kappa_{06} = 16.$$

Ces calculs mettent en évidence les premières branches de l'arborescence (*fig.* 1).

([2]) Un deuxième exemple a été traité dans le mémoire de 3e cycle de G. Leflaive-Groussaud, effectué en 1980 au Lamsade sous notre direction.

([3]) On s'aperçoit par là que le programme linéaire n'admet pas de solutions optimales multiples. La méthode ne servira donc qu'à la recherche de solutions quasi optimales.

TABLEAU I

Exemple d'application de la méthode du simplexe inverse
(1re phase)

Tableau simplexe	Variables de base	Vecteurs **b**	x_1	x_2	x_3	x_4
00	F	0	3	4	5	6
	x_5	18	1	1	1	1
	x_6	6	0	0	2	3
			x_1	x_5	x_3	x_6
0	$F^*[Y]$	76[20]	1	4	1/3	2/3
	x_2	16	[1]	[1]	1/3	−1/3
	x_4	2	0	0	[2/3]	[1/3]
	κ_{0j}		4	−44	19	16
			x_1	x_5	x_4	x_6
1	Y	19	1	4	−1/2	1/2
	x_2	15	[1]	[1]	−1/2	−1/2
	x_3	3	0	0	[3/2]	[1/2]
	κ_{1j}		4	−41	–	16
			x_1	x_5	x_3	x_4
2	Y	16	1	4	−1	−2
	x_2	18	[1]	[1]	1	1
	x_6	6	0	0	2	3
	κ_{2j}		−2	−56	–	–
			x_2	x_5	x_3	x_6
3	Y	4	−1	3	0	1
	x_1	16	1	[1]	1/3	−1/3
	x_4	2	0	0	[2/3]	[1/3]
	κ_{3j}		–	−44	4	−2
			x_2	x_5	x_4	x_6
4	Y	4	−1	3	0	1
	x_1	15	1	[1]	−1/2	−1/2
	x_3	3	0	0	3/2	[1/2]
	κ_{4j}		–	−41	–	−2

Étape 3 : D'après la formule (8), nous avons :

$$\kappa_1 = \max\{\kappa_{01}, \kappa_{03}, \kappa_{06}, 0\} = \kappa_{03} = 19.$$

La variable x_3 entrera dans la nouvelle base et la solution quasi optimale aura une déviation d'une unité de F^* ($F = 76 - 1 = 75$). κ_{03} ne sera pas repris ensuite dans l'ensemble de la formule (8).

Étape 4 : Le pivotage ($x_3 \uparrow$, $x_4 \downarrow$) donne le tableau n° 1 à partir duquel l'algorithme retourne à l'étape 1 avec $\kappa_1 = 19$.

$s = 1$

Étape 1 : Détermination des pivots dans le tableau n° 1 à partir des variables entrantes x_1, x_5 et x_6.

Étape 2 : De la même façon qu'à l'itération 0, nous obtenons $\kappa_{11} = 4$, $\kappa_{15} = -41$, $\kappa_{16} = 16$.

Étape 3 : $\kappa_2 = \max\{\kappa_{01}, \kappa_{06}, \kappa_{11}, \kappa_{16}, 0\} = 16$, ce qui correspond indifféremment à κ_{06} ou κ_{16}. On peut donc faire entrer la variable x_6 dans la base des tableaux n° 0 ou 1.

Étape 4 : Pivotage ($x_6 \uparrow$, $x_4 \downarrow$) dans le tableau n° 0, ce qui donne le tableau n° 2. Retour à l'étape 1.

$s = 2$

Étape 1 : Les variables qui peuvent entrer dans la base du tableau n° 2 sont : x_1 et x_5; deux nouveaux pivots sont donc repérés.

Étape 2 : $\kappa_{21} = -2$, $\kappa_{24} = -56$.

Étape 3 : $\kappa_3 = \max\{\kappa_{01}, \kappa_{11}, 0\} = \kappa_{01} = \kappa_{11} = 4$. Nous ferons donc entrer la variable x_1 dans un des tableaux n^{os} 0 ou 1.

Étape 4 : Pivotage ($x_1 \uparrow$, $x_2 \downarrow$) dans le tableau n° 0 donnant lieu au tableau n° 3. Retour à l'étape 1.

$s = 3$

Étape 1 : Variables pouvant entrer dans la base du tableau n° 3 : x_5, x_3, x_6.

Étape 2 : $\kappa_{35} = -44$, $\kappa_{33} = 4$, $\kappa_{36} = -2$.

Étape 3 : $\kappa_4 = \max\{\kappa_{33}, 0\} = \kappa_{33} = 4$.

Étape 4 : Pivotage ($x_3 \uparrow$, $x_4 \downarrow$) dans le tableau n° 3. Calcul du tableau n° 4. Retour à l'étape 1.

$s = 4$

Étape 1 : Variables entrant dans la base du tableau n° 4 : x_5 et x_6.

Étape 2 : $\kappa_{45} = -41$, $\kappa_{46} = -2$.

Étape 3 : $\kappa_5 = \max\{0\} = 0$. Aller en Phase II.

Phase II

Reprendre les tableaux-simplexe n^{os} 1, 2, 3, 4 construits en Phase I et appliquer les formules (9)-(10) sans reproduire la même solution (*fig.* 1). Ces solutions correspondant à des pivotages du type ($x_j \uparrow$, $Y \downarrow$), nous aurons $Y = 0$, d'où, d'après (5), $\mathbf{c}^t \mathbf{x} = F^* - k = 56$. Huit nouvelles solutions sont calculées de cette façon (*cf.* tableau II, n^{os} 5-12). La solution n° 5, par exemple, est constituée, en appliquant (9)-(10) sur le tableau-simplexe n° 0 (car $\kappa_{05} < 0$), par les composantes non nulles :

$$x_5 = \frac{20}{4} = 5,$$

$$x_2 = 16 - 1 \times 5 = 11,$$

$$x_4 = 2 - 0 \times 5 = 2.$$

Les sept autres solutions du tableau 2 ont été obtenues de façon analogue suivant le schéma de la figure 1.

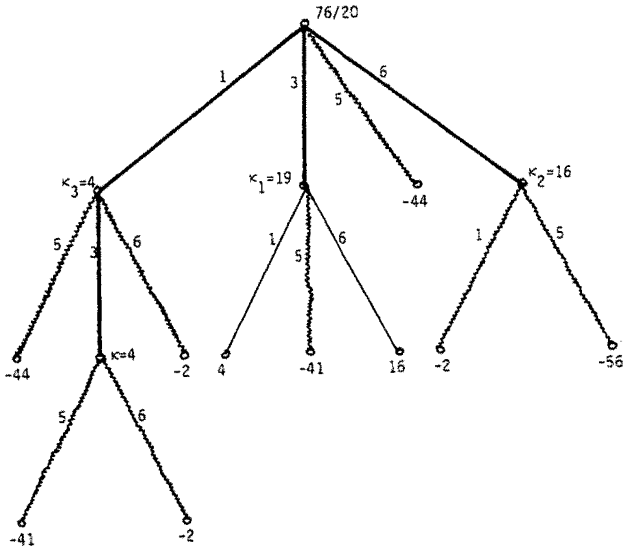

Figure 1. — Arborescence due à l'application de la méthode du simplexe inverse à l'exemple PL2.
Arcs ——— : solutions de la première phase; arcs ——— : solutions de la seconde phase; arcs ——— : solutions multiples déjà calculées en première phase.

TABLEAU II

Solutions quasi optimales du programme PL2 (n^{os} 1-12)

Solution n°	Valeur de F	x_1	x_2	x_3	x_4	x_5	x_6
0	76	0	16	0	2	0	0
1	75	0	15	3	0	0	0
2	72	0	18	0	0	0	6
3	60	16	0	0	2	0	0
4	60	15	0	3	0	0	0
5	56	0	11	0	2	5	0
6	56	0	41/4	3	0	19/4	0
7	56	16	2	0	0	0	6
8	56	0	14	0	0	4	6
9	56	44/3	0	0	2	4/3	0
10	56	52/3	0	0	2/3	0	4
11	56	41/3	0	3	0	4/3	0
12	56	17	0	1	0	0	4

Critique de la méthode

Cette méthode est caractérisée par une rapidité considérable provenant du simple fait qu'elle autorise uniquement les itérations-simplexe qui déterminent chaque fois des nouveaux sommets de l'hyperpolyèdre Ph2. Un autre avantage de la méthode, le plus important, est que la recherche des solutions se fait au fur et à mesure que la fonction-objectif F décroît. Cette propriété permet d'étudier la *sensibilité* de la solution optimale d'un programme linéaire aux variations éventuelles du paramètre k. Il est regrettable que la méthode (sous sa forme actuelle) ne puisse pas garantir la recherche exhaustive des sommets de Ph2. A ce niveau, elle pourrait être très bien complétée par une des méthodes de la section 4 avec application de celle-ci pour chaque palier de F entre F^* et F^*-k.

Une telle recherche permet aux analystes, notamment pour des programmes linéaires de grande taille où le nombre des sommets devient considérable sinon introuvable, d'obtenir en priorité les solutions les moins écartées de l'optimum F^*, ce qui leur donne la possibilité d'arrêter, à un moment donné, la recherche d'autres sommets de l'hyperpolyèdre Ph2.

Le désavantage crucial de la méthode réside dans le fait qu'*elle exige le stockage en mémoire d'ordinateur de tous les tableaux-simplexe obtenus lors de l'exécution de la première phase.* L'exemple précédent, par exemple, nécessiterait le stockage de cinq tableaux-simplexe (*cf.* tableau I). Cependant, certaines techniques spéciales de l'algorithme du simplexe, la méthode PFI (Product Form of Inverse; *voir* [8]) par exemple, pourrait réduire la quantité de l'information stockée étant donné que, pour mémoriser un tableau-simplexe, il

suffirait de stocker un seul vecteur-colonne. Cela, en revanche, alourdirait les calculs dus à la reconstitution des tableaux-simplexe à partir de ces vecteurs.

4. MÉTHODES DE LABYRINTHE

Quasi-optimalité et labyrinthes

Charnes et Cooper (*cf.* [2], [3]) ont remarqué que le problème du traitement de solutions optimales multiples se ramène à un cas particulier du problème classique des labyrinthes en théorie des graphes (*voir* [21] pour un exposé mathématique en la matière). Il suffit de remarquer qu'un hyperpolyèdre du type Ph2 (comme tout ensemble polyédrique convexe d'ailleurs) peut être représenté par un graphe connexe (V, U), V étant l'ensemble de ses sommets et U l'ensemble des arcs de voisinage des sommets qui s'explicitent par des pivotages-simplexe. Ainsi, deux sommets sont voisins s'ils correspondent à des bases-simplexe différant d'une seule variable; chaque arc peut donc être traversé dans les deux sens du fait que chaque pivotage-simplexe est réversible (*cf.* section 2). Le graphe qui correspond à l'hyperpolyèdre Ph2 de l'exemple précédent est donné dans la figure 3.

La recherche de tous les sommets de l'hyperpolyèdre Ph2 se ramène à l'exploration de tous les sommets du graphe (V, U) dont, à l'exception des arcs de voisinage (pivotages à partir d'un tableau-simplexe), la forme explicite n'est pas connue *a priori*. Il s'agit là par conséquent de la recherche d'un parcours dans un labyrinthe où les carrefours sont représentés par les sommets et les allées par les arcs de (V, U).

Pour formaliser un peu ces idées, nous devons donner quelques définitions.

— V (ensemble des sommets du graphe) est composé des m-uples d'entiers (indices de base-simplexe) $v=(i_1, i_2, \ldots, i_m)$.

— Deux sommets différents $v_1=(i_1, i_2, \ldots, i_m)$ et $v_2=(k_1, k_2, \ldots, k_m)$ ont la distance $d \leqq m$ si ces sommets diffèrent de d composantes exactement; ces sommets sont donc voisins si $d=1$.

— Un arc $(v_1, v_2) \in U$ si et seulement si v_1 et v_2 sont voisins [les sommets voisins de v_i forment un ensemble noté $\Gamma(v_i)$].

Le problème de la recherche de tous les sommets d'un graphe (V, U) se formule maintenant différemment : trouver un chemin qui passe par tous les sommets du graphe en ne gardant en mémoire qu'un seul tableau-simplexe pour générer les nouveaux sommets.

L'algorithme de Tarry [24]

Tarry [24] a présenté, en 1895, la première méthode qui permet de résoudre le problème de parcours d'un labyrinthe; sa méthode est également exposée dans [25] et [2, 3]. Pour l'exploration d'un graphe, l'algorithme fonctionne de la façon suivante :

A chaque sommet, aller à un autre sommet suivant un arc qui n'a pas été traversé en cette direction mais ne reprendre l'arc de la première visite à ce sommet que si l'on ne peut pas faire autrement.

L'application de cet algorithme prescrit que chaque arc sera traversé dans les deux sens et que chaque sommet sera visité au moins une fois. Un arc du graphe (V, U) étant désigné par une variable hors base d'un tableau-simplexe, l'algorithme de Tarry nécessitera toujours autant d'itérations-simplexe que le nombre de variables hors base (n pour PL1) multiplié par le nombre de sommets possibles r, c'est-à-dire $n \times r$. Pour notre exemple numérique représenté par le graphe de la figure 3, on aurait dû effectuer $4 \times 12 = 48$ itérations-simplexe! Cet algorithme est donc loin d'être efficace au niveau calcul.

Aujourd'hui, le problème de la recherche explicite des sommets d'un ensemble polyédrique convexe présente une bibliographie déjà très abondante. Nous pouvons citer ici quelques méthodes et quelques auteurs : Balinski [1], Manas et Nedoma [15], Omar [19], Mattheiss [16], Dyer et Proll [6] parmi les méthodes dites de pivotage et, Motzkin *et al.* [18], Chernikova [4] parmi les méthodes géométriques ou sans pivotage-simplexe. Un excellent tour d'horizon de ces méthodes est donné par Mattheiss et Rubin [17]. Ce domaine de recherche n'est pas démuni d'applications; en voici quelques unes : Winkels et Colman [27] utilisent ce genre de techniques pour visualiser, à l'aide de micro-ordinateur, des hyperpolyèdres à cinq dimensions. Rizzi [20] cherche à préciser les valeurs de plausibilité de divers scénarii à partir d'un système d'inégalités linéaires. Greenberg [11], enfin, détecte par cette technique les contraintes redondantes d'un système d'inégalités linéaires.

Dans la suite, nous présentons une des méthodes de pivotage, celle de Manas et Nedoma [15], dont l'efficacité a déjà été reconnue dans deux études comparatives (*cf.* [5, 17]). La méthode a également été intégrée par Gal [10] et Gal et Nedoma [9] dans leur analyse multiparamétrique en PL et par Zeleny [28] dans la recherche des solutions efficaces (non dominées sur tout critère) en PL à plusieurs fonctions-objectifs.

L'algorithme de Manas-Nedoma [15]

Son principe est d'effectuer une « promenade » dans le graphe (V, U) caractérisée par deux propriétés importantes : (1) éviter de reproduire

(si possible) des solutions déjà calculées, (2) n'enregistrer en mémoire qu'un seul tableau-simplexe.

De même que pour les autres méthodes, l'algorithme part du tableau-simplexe optimal (sommet v_0) et construit itérativement deux suites d'ensembles (R_1, W_1), (R_2, W_2), ..., (R_s, W_s), ... contenant respectivement d'une part l'ensemble des sommets déjà rencontrés (ensemble R) et, d'autre part, l'ensemble des sommets non rencontrés accessibles à partir d'au moins un sommet de R par un seul pivotage-simplexe (ensemble W). Ainsi, si $\Gamma(v_s)$ représente les sommets voisins d'un sommet v_s, le déroulement de l'algorithme peut se schématiser par l'organigramme de la figure 2. Ce dernier reprend le formalisme sur les graphes (V, U) donné plus haut; $d(v_s, v_{s+1})$ désigne la distance entre deux bases-simplexe, autrement dit le nombre de pivotages-simplexe nécessaire pour passer du sommet v_s à v_{s+1}.

L'algorithme se termine à une itération s lorsque $W_{s+1} = \varnothing$, à savoir lorsqu'il ne reste plus de sommets à atteindre à partir de la liste R_{s+1}. Le nombre total d'itérations est très réduit par rapport à celui de la méthode de Tarry. Pour un programme linéaire PL1 à r sommets, ce nombre est compris entre r et $m \times r$ à condition qu'il n'y ait pas de dégénérescence (*cf.* [15]). Le type d'hyperpolyèdres où le nombre d'itérations serait r exactement (chemin hamiltonien dans le graphe) n'est pas encore prescrit par les théoriciens (*voir* [17], p. 168).

Nous proposons maintenant d'illustrer cet algorithme sur l'exemple numérique de la section 3. Nous nous reportons pour cela aux tableaux I et II et nous désignons les 13 sommets possibles de l'hyperpolyèdre Ph2 de cet exemple par les numéros portés sur ces solutions, allant de 0 à 12.

Au départ ($s=0$), nous posons $R_0 = \varnothing$, $W_0 = \varnothing$. Pour $s=1$, considérant le tableau-simplexe n° 0 (*cf.* tableau I), nous aurons $R_1 = \{0\}$ et $W_1 = W_0 \cup \Gamma(0)$ où $W_0 = \varnothing$ et $\Gamma(0)$ les solutions n^{os} 1, 2, 3, 5 qui peuvent être acquises par quatre pivotages ayant respectivement comme pivots les éléments : $a_{43} = 2/3$, $a_{46} = 1/3$, $a_{21} = 1$ et $a_{75} = 4$. Comme l'algorithme autorise le choix arbitraire de la variable qui déterminera le sommet suivant, on prend ici systématiquement la solution qui sera la première rencontrée après la solution précédente dans la liste des tableaux I et II, donc la solution n° 1 avec comme pivotage $(x_3 \uparrow, x_4 \downarrow)$.

L'ensemble des itérations se récapitule en détail dans la suite et est également représenté dans la figure 3 par le chemin en pointillés. Il est à remarquer que le chemin d'exploration de l'hyperpolyèdre est curieusement un chemin hamiltonien avec autant d'itérations que de nouveaux sommets à partir du

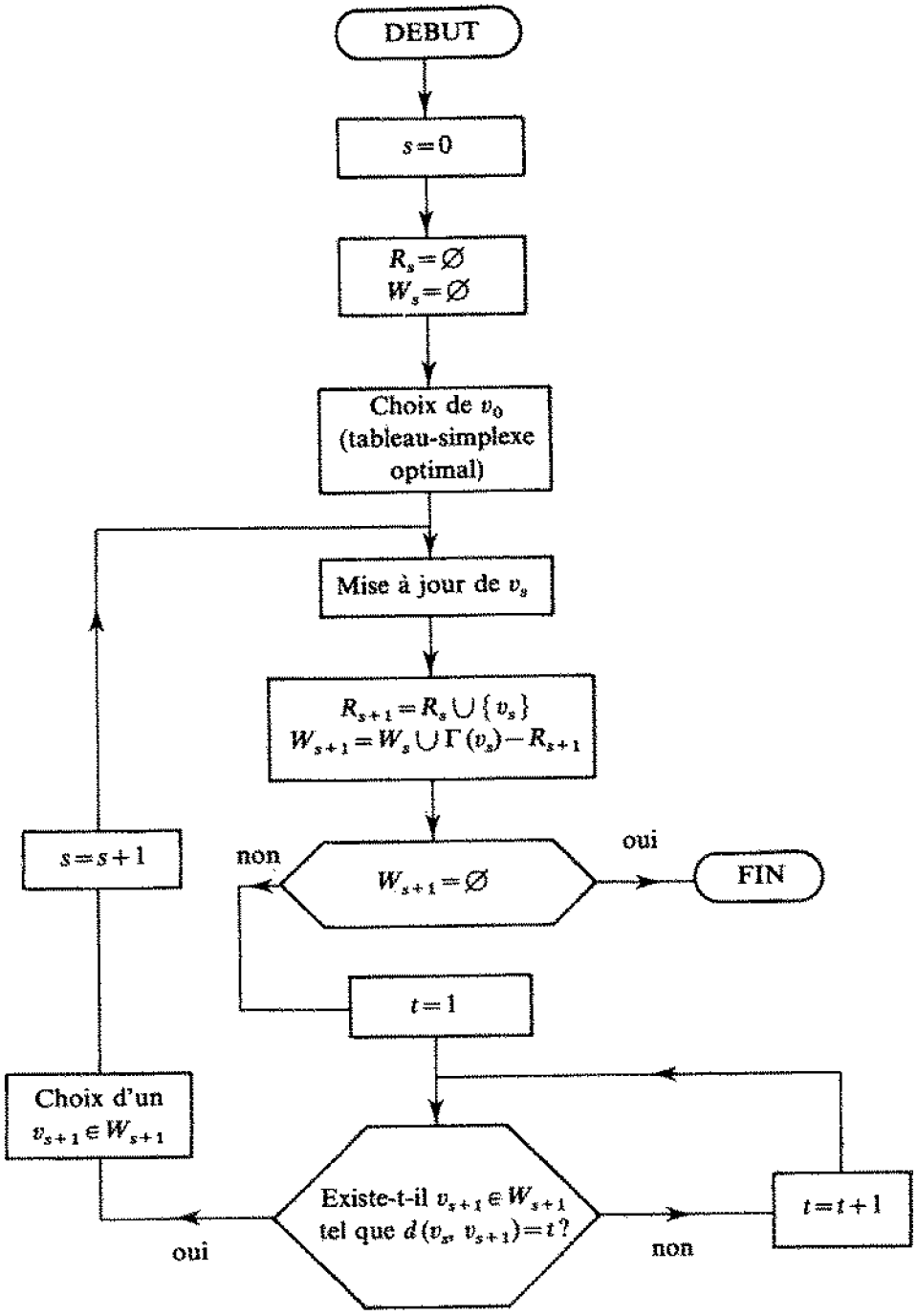

Figure 2. — Organigramme de l'algorithme de Manas et Nedoma.

sommet initial (la distance étant toujours égale à 1) :

$$s=0, \quad R_0=\varnothing, \quad W_0=\varnothing,$$

$$s=1, \quad R_1=\{0\}, \quad W_1=W_0 \cup \Gamma(0)-R_1=\{1, 2, 3, 5\},$$

$$s=2, \quad R_2=\{0, 1\}, \quad W_2=W_1 \cup \Gamma(1)-R_2=\{2, 3, 5, 4, 6\},$$

$$s=3, \quad R_3=\{0, 1, 2\}, \quad W_3=\{3, 4, 5, 6, 7, 8\},$$

$$s=4, \quad R_4=\{0, 1, 2, 7\}, \quad W_4=\{3, 4, 5, 6, 8, 10, 12\},$$

$$s=5, \quad R_5=\{0, 1, 2, 7, 8\}, \quad W_5=\{3, 4, 5, 6, 10, 12\},$$

$$s=6, \quad R_6=\{0, 1, 2, 7, 8, 5\}, \quad W_6=\{3, 4, 6, 10, 12, 9\},$$

$$s=7, \quad R_7=\{0, 1, 2, 7, 8, 5, 6\}, \quad W_7=\{3, 4, 10, 12, 9, 11\},$$

$$s=8, \quad R_8=\{0, 1, 2, 7, 8, 5, 6, 11\}, \quad W_8=\{3, 4, 10, 12, 9\},$$

$$s=9, \quad R_9=\{0, 1, 2, 7, 8, 5, 6, 11, 4\}, \quad W_9=\{3, 10, 12, 9\},$$

$$s=10, \quad R_{10}=R_9 \cup \{3\}, \quad W_{10}=\{9, 10, 12\},$$

$$s=11, \quad R_{11}=R_{10} \cup \{9\}, \quad W_{11}=\{10, 12\},$$

$$s=12, \quad R_{12}=R_{11} \cup \{10\}, \quad W_{12}=\{12\},$$

$$s=13, \quad R_{13}=R_{12} \cup \{12\}, \quad W_{13}=\varnothing, \quad \text{FIN.}$$

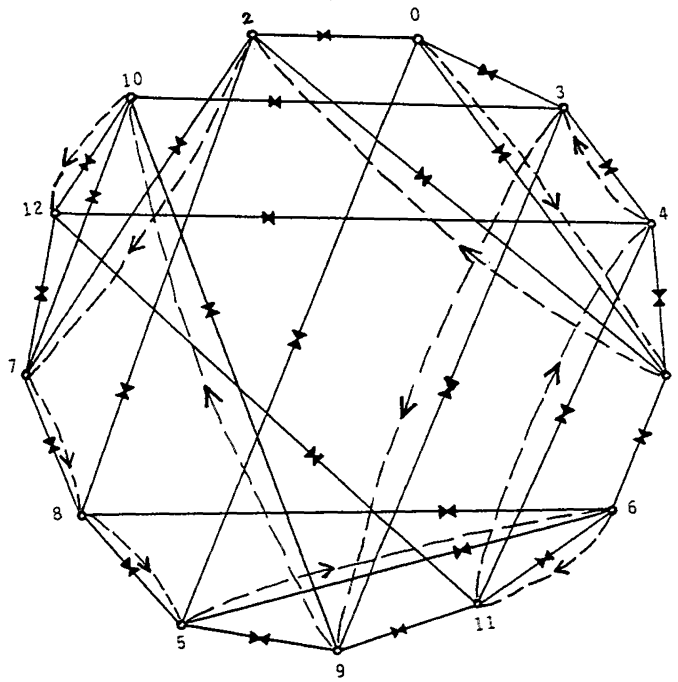

Figure 3. – Le chemin 0-1-2-7-8-5-6-11-4-3-9-10-12 indique le déroulement de l'algorithme de Manas-Nedoma sur l'exemple de la section 3.

La complexité des algorithmes de labyrinthe

Seul l'algorithme de Manas-Nedoma a montré que, en gardant un tableau-simplexe unique pour le besoin des calculs, on peut explorer les sommets d'un hyperpolyèdre de façon très efficace. Il reste à examiner le temps-machine que l'algorithme nécessite pour traiter des programmes linéaires de taille raisonnable.

L'étude comparative de Mattheiss et Rubin [17] peut nous servir de guide. Les auteurs ont comparé quatre algorithmes de recherche des sommets d'un hyperpolyèdre (Balinski [1], noté B; Chernikova [4], noté C; Mattheiss [16], noté M et Manas-Nedoma [15], noté M-N) sur la base d'un grand échantillon d'hyperpolyèdres généré aléatoirement par un ordinateur IBM 370/168. Utilisant une capacité-mémoire de 312 K, la taille maximale de programme linéaire qu'ils ont pu traiter était respectivement 28×8 (B), 25×8 (C), 29×20 (M) et 23×14 (M-N). Le nombre maximal de sommets recherchés dans cette expérimentation était de 22 000 environ. Une première limite est donc due au traitement de grands programmes linéaires. Une deuxième expérimentation, plus modeste que la précédente, a été menée par Dyer et Proll [5]. Les auteurs n'ont pas pu dépasser cette fois-ci 250 sommets. Pourtant, leurs résultats semblent analogues à ceux obtenus dans [17].

Mattheiss et Rubin ont pu tracer une droite de régression linéaire pour chaque méthode comparée, du type (11) :

$$\ln (\text{temps en secondes}) = b_0 + b_1 \ln (\text{nombre des sommets}), \qquad (11)$$

avec ($b_0 = -5{,}345$, $b_1 = 2{,}272$) pour B, ($b_0 = -5{,}589$, $b_1 = 1{,}418$) pour C, ($b_0 = -5{,}386$, $b_1 = 1{,}146$) pour M et ($b_0 = -5{,}046$, $b_1 = 1{,}379$) pour M-N. Ces temps ayant l'air de ne pas être très éloignés varient pourtant de quelques minutes à quelques milliers d'heures ! En effet, pour un hyperpolyèdre comprenant 10 000 sommets, la formule (11) donne, dans l'ordre : 2,93 minutes pour M, 29,28 minutes pour C, 35,19 minutes pour M-N et . . . 1 623,28 heures pour B ! De plus, pour des nombres de sommets plus élevés, les temps pour M-N deviennent meilleurs que pour C.

Les problèmes de la taille-mémoire occupée et du temps exigé dans le calcul de plusieurs milliers de sommets sont inévitables dans ces méthodes. On se demande parfois s'il est nécessaire de se livrer à des tâches si coûteuses pour traiter les solutions optimales ou quasi optimales en PL.

5. DEUX HEURISTIQUES ET CONCLUSION

Le modeste tour d'horizon que nous avons entrepris sur les méthodes exactes de traitement des solutions quasi optimales en PL n'a fait que nous décourager de les appliquer à des problèmes de taille courante.

Le recours à des *techniques heuristiques* est une bonne solution pour sortir de l'impasse, d'autant plus que l'on s'intéresse le plus souvent non pas à connaître des milliers de solutions-sommets mais à un tout autre type d'informations comme, par exemple : la stabilité d'une solution dégénérée, les « fourchettes » de variation des valeurs que prennent les variables dans ce type de solutions, le choix d'un ensemble (voire système) de telles solutions qui soient aussi représentatives que possible de l'hyperpolyèdre Ph2, la corrélation éventuelle (effets de synergie) entre variables, etc.

Siskos [22] propose un type particulier d'analyse où le système des solutions recherchées comprend seulement celles qui maximisent et/ou minimisent chaque fois une ou plusieurs variables x_j. Ces solutions sont détectées à partir de l'optimum par résolution de nouveaux programmes linéaires du type PL3 ci-dessous :

$$\text{PL3} \begin{cases} [\max] \text{ ou } [\min] \sum_j \rho_j x_j, & \rho_j = 0 \quad \text{ou} \quad 1, \qquad \forall j, \\ \text{sur l'hyperpolyèdre Ph2.} \end{cases}$$

Pour $\rho_1 = 1$ et $\rho_j = 0$, $j \neq 1$, on trouve la solution qui contient la valeur maximale (ou minimale) de la variable x_1. Pour maximiser x_1 sur l'exemple numérique de la section 3, on aurait dû calculer les nouveaux profits marginaux sur le tableau-simplexe n° 0 du tableau I, en prenant comme coefficients économiques $c_1 = 1$, $c_\gamma = 0$, $c_j = 0$, $j \neq 1$, et procéder aux pivotages successifs ($x_1 \uparrow$, $x_2 \downarrow$), tableau-simplexe n° 3 (*cf.* tableau I), et ($x_6 \uparrow$, $Y \downarrow$) pour obtenir la solution n° 10 du tableau II, avec $x_1 = 52/3$ maximum. Pour continuer maintenant avec la recherche d'autres solutions, il suffit de continuer de la même façon exactement, non plus sur le tableau-simplexe n° 0 mais à partir du tableau-simplexe de cette dernière solution. Cette recherche se fait actuellement sur l'algorithme-simplexe *LU* (algorithme de factorisation) sur micro-ordinateur et de façon interactive.

Cette méthode s'applique avec beaucoup de succès aux problèmes de régression ordinale multicritère. Les variables que l'on maximise d'avantage dans (PL3) sont les poids ou des combinaisons appropriées de poids des critères. Siskos [23] donne un éventail d'applications fondées sur les principes de cette heuristique. En particulier, pour un problème de sélection de points de vente, trois solutions du type de PL3 ont suffit pour constater la stabilité du modèle d'évaluation des points de vente (*voir* [23]). Il s'agissait de programmes linéaires de dimensions 81×113.

Winkels [26] propose une heuristique consistant à explorer l'hyperpolyèdre Ph2 de façon plus régulière sans se limiter forcément à ses sommets ni se préoccuper des propriétés *a priori* des solutions recherchées. C'est une *méthode*

de discrétisation de Ph2 dont le principe repose sur le choix d'un pas de discrétisation pour chacune des variables x_j (cf. *fig.* 4).

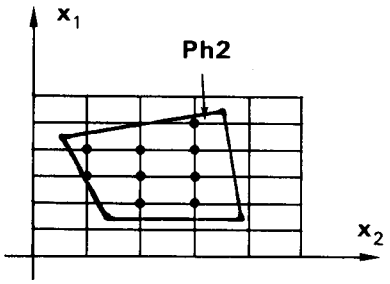

Figure 4. — Schéma de discrétisation d'un polyèdre.

Cette technique pourrait, selon l'auteur, définir automatiquement les pas de discrétisation une fois que le nombre des solutions désirées est fixé d'avance par l'analyste. Pourtant, les expériences sont insuffisantes; il manque notamment des essais avec des programmes linéaires de grande taille; on peut se demander quel type de solutions donnerait la méthode pour la recherche d'une petite dizaine de points d'un tel programme linéaire.

En conclusion, nous soulignons encore une fois l'importance du problème de solutions multiples en programmation linéaire, problème que beaucoup d'auteurs sous-estiment et dont les codes informatiques, malheureusement, ne tiennent pas compte.

REMERCIEMENTS

Mes remerciements s'adressent à Marie-Claude Portmann et un « referee » anonyme dont les remarques et suggestions m'ont permis d'améliorer les versions précédentes de cet article.

BIBLIOGRAPHIE

1. M. L. Balinski, *An Algorithm for Finding all Vertices of Convex Polyhedral Sets*, J. Soc. Indust. Appl. Math., vol. 9, n° 1, 1961, p. 72-88.
2. A. Charnes, *Optimality and Degeneracy in Linear Programming*, Econometrika, vol. 20, 1952, p. 160-170.
3. A. Charnes et W. W. Cooper, *Management Models and Industrial Applications of Linear Programming*, John Wiley and Sons, vol. 1, New York, 1961.
4. N. V. Chernikova, *Algorithm for Finding a General Formula for the Non-Negative Solutions of a System of Linear Inequalities*, U.S.S.R. Computational Mathematics and Mathematical Physics, vol. V, 1965, p. 228-233.

5. M. E. DYER et L. G. PROLL, *Vertex Enumeration in Convex Polyhedra: A Comparative Computational Study*, in T. B. BOFFEY, éd., Proc. CP77 Combinatorial Programming Conference, University of Liverpool, Liverpool, 1977, p. 23-43.

6. M. E. DYER et L. G. PROLL, *An Improved Vertex Enumeration Algorithm*, European Journal of Operational Research, vol. 9, 1982, p. 359-368.

7. R. FAURE, *La programmation linéaire appliquée*, Que sais-je?, n° 1776, P.U.F., Paris, 1979.

8. R. FAURE, *Précis de recherche opérationnelle* (4e éd. entièrement refondue d'Éléments de la Recherche Opérationnelle, 1968), Dunod, Paris, 1979.

9. T. GAL et J. NEDOMA, *Multiparametric Linear Programming*, Management Science, vol. 18, 1972, p. 406-422.

10. T. GAL, *Postoptimal Analyses, Parametric Programming, and Related Topics*, MacGraw Hill, New York, 1979.

11. H. GREENBERG, *An Algorithm for Determining Redundant Inequalities and all Solutions to Convex Polyhedra*, Numerische Mathematik, vol. 24, 1975, p. 19-26.

12. G. HADLEY, *Linear Programming*, Addison-Wesley, Reading, Massachusetts, 1962.

13. J. P. IGNIZIO, *Goal Programming and Extensions*, Lexington Books, D.C. Heath and Company, Massachusetts, 1976.

14. V. KLEE, *On the Number of Vertices of a Convex Polytope*, Canadian Journal of Mathematics, vol. 16, 1964, p. 701-720.

15. M. MANAS et J. NEDOMA, *Finding all Vertices of a Convex Polyhedron*, Numerische Mathematik, vol. 14, 1968, p. 226-229.

16. T. H. MATTHEISS, *An Algorithm for Determining Irrelevant Constraints and all Vertices in Systems of Linear Inequalities*, Operations Research, vol. 21, 1973, p. 247-260.

17. T. H. MATTHEISS et D. S. RUBIN, *A Survey and Comparison of Methods for Finding all Vertices of Convex Polyhedral Sets*, Mathematics of Operations Research, vol. 5, 1980, p. 167-185.

18. T. S. MOTZKIN, H. RAIFFA, G. L. THOMPSON et R. M. THRALL, *The Double Description Method*, in: H. W. KUHN et A. W. TUCKER, éds., Contributions to the Theory of Games, vol. 2, Princeton University Press, Princeton, New Jersey, 1953.

19. A. OMAR, *Finding all Extreme Points and Extreme Rays of a Convex Polyhedral Set*, Ekonomicko-Matematicky, Obzor, vol. 3, 1977, p. 331-342.

20. M. RIZZI, *Une nouvelle méthode d'aide à la décision en avenir incertain*, R.A.I.R.O. Recherche Opérationnelle, vol. 16, n° 4, 1982, p. 391-405.

21. P. ROSENSTIEHL, *Labyrinthologie mathématique*, Mathématiques et Sciences Humaines, 9e année, n° 33, 1971, p. 5-32.

22. J. SISKOS, *Comment modéliser les préférences au moyen de fonctions d'utilité additives*, R.A.I.R.O. Recherche Opérationnelle, vol. 14, n° 1, 1980, p. 53-82.

23. J. SISKOS, *Application de la méthode UTA I à un problème de sélection de points de vente mettant en jeu des critères multiples*, R.A.I.R.O. Recherche Opérationnelle, vol. 17, n° 2, 1983, p. 121-136.

24. G. TARRY, *Le problème des labyrinthes*, Nouvelles Annales de Mathématiques, vol. XIV, 1895, p. 187-190.

25. C. VAN DE PANNE, *Methods for Linear and Quadratic Programming*, North-Holland Publishing Company, Amsterdam, 1975.

26. H. M. WINKELS, *A Flexible Decision Aid Method for Linear Multicriteria Systems*, in: M. GRAUER, A. LEWANDOWSKI et A. P. WIERZBICKI, éds., Multiobjective and Stochastic Optimization, I.I.A.S.A. Collaborative Proceedings Series CP-82-812, Laxenburg (Austria), 1982, p. 377-410.

27. H. M. WINKELS et R. COLMAN, *Visualization of 5 Dimensional Polyhedra*, Working paper on economathematics n° 8209, Ruhr-Universität Bochum, 1982.

28. M. ZELENY, *Linear Multiobjective Programming*, Springer-Verlag, Berlin, 1974.