REVUE FRANÇAISE D'AUTOMATIQUE, INFORMATIQUE, RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

PHILIPPE CHRÉTIENNE

# Détermination rapide d'une solution de base initiale en programmation linéaire continue

*Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, tome 9, n° V1 (1975), p. 113-118

<http://www.numdam.org/item?id=RO_1975__9_1_113_0>

© AFCET, 1975, tous droits réservés.

L'accès aux archives de la revue « Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

*Article numérisé dans le cadre du programme*
*Numérisation de documents anciens mathématiques*
http://www.numdam.org/

R.A.I.R.O.
(9e année, janvier 1975, V-1, p. 113 à 118)

# DETERMINATION RAPIDE D'UNE SOLUTION DE BASE INITIALE EN PROGRAMMATION LINEAIRE CONTINUE (*)

par Philippe CHRETIENNE (1)

Résumé. — *On effectue initialement une bipartition de l'ensemble des contraintes d'un programme linéaire continu. L'itération k de la méthode consiste à trouver une solution de base réalisable pour l'ensemble des contraintes à second membre positif satisfaisant, au moins, une contrainte à second membre négatif préalablement choisie. Lorsque tous les seconds membres sont positifs ou nuls, la base associée est réalisable pour le programme linéaire initial.*

**La méthode :** elle comporte deux parties,

I) l'initialisation d'une partition ;

II) la recherche itérative de solutions réalisables de base pour les ensembles $C_k^+ \cup \{ p \}$ ; pour $p \in C_k^-$.

$C_k^+$ est l'ensemble des contraintes à second membre positif à l'itération $k$.

$C_k^-$ est l'ensemble des contraintes à second membre négatif à l'itération $k$.

## I. L'initialisation

Si nous formulons le programme linéaire initial sous la forme

$$A \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i & \text{pour} \quad i = 1, 2, ..., m \\ x_j \geqslant 0 & \text{pour} \quad j = 1, 2, ..., n \\ \text{Max } z = \sum_{j=1}^{n} c_j x_j \end{cases}$$

Trois cas sont à priori possibles :

1) $\forall i,\ i = 1, 2, ..., m\ b_i \geqslant 0$ ; alors la solution $(x_j = 0 ; j = 1, 2, ..., n)$ est une solution de base réalisable ; ce cas, bien sûr, ne nous intéresse pas ici.

(1) Université Paris 6. Institut de programmation.
(*) Reçu le 19 mars 1974.

2) $\forall i,\ i = 1, 2, \ldots, m\ \ b_i < 0$; alors nous construirons simplement une contrainte à second membre positif, à l'aide d'un pivotage supplémentaire. Considérons, en effet, une telle contrainte :

$$\sum_{j=1}^{n} a_{ij}x_j + s_i = b_i \qquad b_i < 0 \qquad s_i \geqslant 0$$

Si $\forall j\ \ a_{ij} \geqslant 0$ alors le programme linéaire initial est impossible. Soit donc $j_0$ tel que $a_{ij_0} < 0$, nous choisissons $a_{ij_0}$ comme pivot ; effectuons le pivotage, la contrainte $i$ devenant contrainte de $C_0^+$.

3) $\exists i, \exists j\ b_i \geqslant 0\ b_j < 0$ alors nous pourrons poser :

$$C_0^+ = \{\, i/b_i \geqslant 0 \,\}$$
$$C_0^- = \{\, i/b_i < 0 \,\}$$

Il nous sera possible de considérer le programme dual si nous le jugeons plus intéressant vis-à-vis de la méthode.

En particulier dans le cas 2, l'initialisation peut être plus rapide en considérant le programme dual. (Il peut exister des $c_j \leqslant 0$.)

**II.** Considérons à l'itération $k$, les classes $C_{k-1}^+$ et $C_{k-1}^-$ qui constituent la partition de l'ensemble $C$, résultat de l'itération $(k-1)$.

Il nous faut lors de cette itération

1) Choisir une contrainte $p \in C_{k-1}^-$

2) Trouver dans $C_{k-1}^+$ une solution de base réalisable pour $p$.

Nous reviendrons plus loin sur le choix de $p$.

Par contre 2) peut se traiter *en entreprenant* la résolution du programme linéaire suivant :

B) Max $(s_p)$ sur $C_{k-1}^+$ ($s_p$ est la variable d'écart associée à $p$). En fait, on arrêtera les pivotages dès que l'on aura trouvé une base réalisable de $C_{k-1}^+$ telle que : $s_p \geqslant 0$.

A ce moment :
$$\begin{cases} C_k^- = C_{k-1}^- - \{\, p \,\} \\ C_k^+ = C_{k-1}^+ \cup \{\, p \,\} \end{cases}$$

3 sorties sont possibles à l'issue de 2)

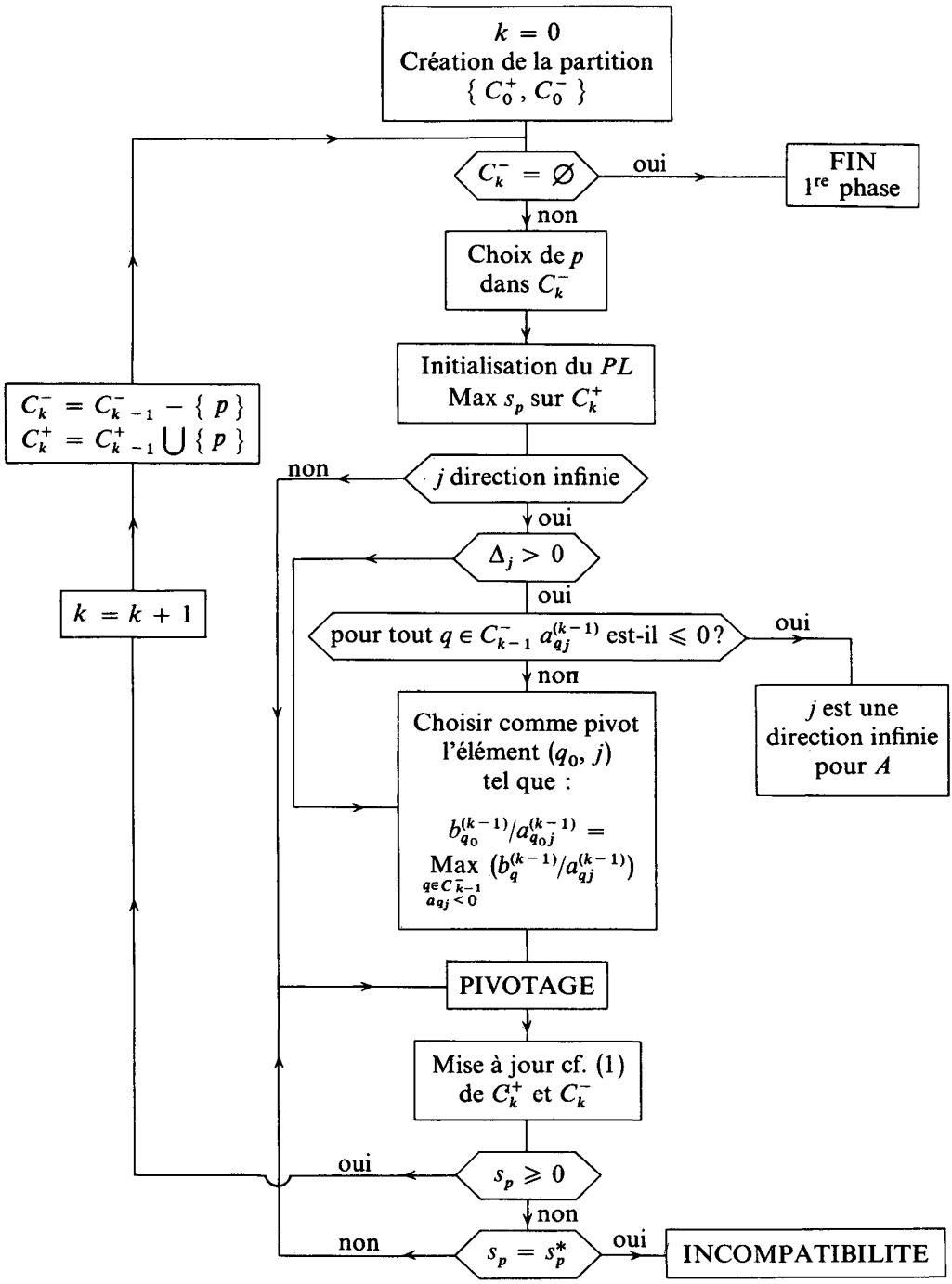
$S_1$) Sortie normale, on trouve une base pour laquelle $s_p \geqslant 0$.

$S_2$) On trouve une direction infinie pour $s_p$.

Soit $j$ cette direction infinie.

Nous allons exploiter cette situation de la manière suivante :

1) Si *a*) pour tout $q \in C_{k-1}^-\ a_{qj}^{k-1} \leqslant 0$
et *b*) le coût marginal $\Delta_j$ associé à $z$ est positif,
alors $j$ est une direction infinie pour le programme linéaire initial.


$k = 0$
Création de la partition
$\{ C_0^+, C_0^- \}$
$C_k^- = \emptyset$
oui
FIN
1re phase
non
Choix de $p$
dans $C_k^-$
Initialisation du $PL$
Max $s_p$ sur $C_k^+$
$C_k^- = C_{k-1}^- - \{ p \}$
$C_k^+ = C_{k-1}^+ \cup \{ p \}$
non
$j$ direction infinie
oui
$\Delta_j > 0$
oui
$k = k + 1$
pour tout $q \in C_{k-1}^-$ $a_{qj}^{(k-1)}$ est-il $\leqslant 0$ ?
oui
non
$j$ est une
direction infinie
pour $A$
Choisir comme pivot
l'élément $(q_0, j)$
tel que :
$b_{q_0}^{(k-1)}/a_{q_0 j}^{(k-1)} =$
Max $(b_q^{(k-1)}/a_{qj}^{(k-1)})$
$q \in C_{k-1}^-$, $a_{qj} < 0$
PIVOTAGE
Mise à jour cf. (1)
de $C_k^+$ et $C_k^-$
oui
$s_p \geqslant 0$
non
non
$s_p = s_p^*$
oui
INCOMPATIBILITE

2) Sinon nous choisirons comme pivot l'élément $a_{q_0 j}^{(k-1)}$ tel que

$$\frac{b_{q_0}^{(k-1)}}{a_{q_0 j}^{(k-1)}} = \underset{q \in C_{k-1}^-}{\text{Max}} \left( \frac{b_q^{(k-1)}}{a_{qj}^{(k-1)}} \middle/ a_{qj}^{(k-1)} < 0 \right)$$

Nous effectuerons le pivotage, qui aura nécessairement comme conséquence :

$$p \in C_k^+$$

*mais aussi :* dans le cas où $q_0 \neq p$, plusieurs autres contraintes de $C_{k-1}^-$ se trouveront du même coup appartenir à $C_k^+$ (accélération du processus de réduction de $C_0^-$).

$S_3$) On a résolu complètement B) et $s_p^*$ ($s_p^*$ valeur optimale de $s_p$) est strictement négatif. Alors la contrainte $p$ est incompatible avec $C_{k-1}^+$, le programme linéaire initial $A$ n'a pas de solution.

L'organigramme général de II est donné ci-devant.

REMARQUES :

(1) Après chaque pivotage, on peut mettre à jour $C_k^+$ et $C_k^-$ si certaines contraintes ont vu leur second membre devenir positif ou nul, c'est une manière de « brûler les étapes », l'itération $k$ ayant comme résultat de supprimer plus d'une contrainte de $C_{k-1}^-$.

(2) Toute contrainte $p$ de $C_0^-$ ne peut être redondante avec $C_0^+$ car la solution de base réalisable de $C_0^+$ ($x_j = 0, j = 1, 2, ..., n$) n'est pas réalisable pour $p$.

**Avantages de cette méthode par rapport aux algorithmes classiques**

Dans la méthode habituelle en 2 phases, la phase I consiste à résoudre *complètement* un programme linéaire continu dont l'objectif est :

$\text{Max}\left( - \sum_{i \in A} x_i \right)$ ($A$ : ensemble des variables artificielles), sur le polyèdre total des contraintes.

La méthode proposée consiste au contraire à *limiter le nombre de pivotages* nécessaires à l'obtention d'une solution de base réalisable. D'autre part chaque pivotage ici concerne $(n + 1)(m + 1)$ coefficients alors qu'avec l'introduction de $k$ variables artificielles un pivotage concerne $(n + k)(m + 1)$ coefficients.

Cette méthode ne prend en considération que le caractère réalisable ou non des bases de $C_k^+$ vis-à-vis de la contrainte $\{ p \}$ en cours, et consiste donc à obtenir « le plus rapidement possible » une solution de base réalisable pour $C$. De la même façon que la méthode proposée, l'algorithme paramétrique auto-dual ne nécessite pas l'introduction de variables artificielles [3]; il

oblige cependant à traiter chaque coût marginal et chaque second membre comme une forme linéaire du paramètre, d'où une gestion plus lourde dont on n'est pas sûr qu'elle sera compensée par une convergence plus rapide vers l'optimum. Cependant, l'avantage de cet algorithme réside dans le fait, que la première solution primale ou duale réalisable trouvée est optimale.

– Retour sur le choix de la contrainte $p$ à introduire lors d'une itération.

Il est clair que l'efficacité de la méthode dépend

1) du critère de choix de la contrainte $p$.

2) de la séquence des pivotages réalisés.

Pour le problème 2), nous pouvons *dynamiquement au cours du déroulement du programme* nous situer entre les 2 politiques extrêmes suivantes :

$P_1$ : choisir de faire entrer dans la base le vecteur dont le coût marginal est le plus grand ; politique qui consiste à ne prendre en compte et n'avantager que la contrainte $p$ en cours.

$P_2$ : choisir parmi les directions *à pente positive* pour $s_p$ celle qui provoque un accroissement pour l'objectif du plus grand nombre de contraintes de $C_k^-$, politique qui favorise le plus grand nombre.

Si par exemple, au départ $|C_0^-|/|C|$ est assez grand, il est bon d'essayer par $P_2$ de se débarrasser d'un maximum de contraintes, puis lorsque $|C_k^-|/|C|$ devient petit (paramètre $\alpha$ à préciser) de favoriser séparément par $P_1$ les contraintes introduites.

Le problème 1 est plus délicat, il est en effet plus difficile de savoir à priori quelle est la contrainte de $C_k^-$ qui impliquera le plus petit nombre de pivotages.

L'emploi systématique de la politique $P_1$ constitue une méthode assez proche de celle de l'algorithme composite, encore appelée méthode des pseudo-coûts marginaux [1]. Rappelons que cette méthode calcule, pour régler le sort d'une contrainte $p$ (telle que $b_p < 0$) des pseudo-coûts marginaux :

$$p\Delta_j = \varepsilon \Delta_j - a_{pj}$$

Cette méthode efficace, bien que peu employée, a cependant 2 inconvénients majeurs.

1) Elle ne tient compte que d'une contrainte à la fois.

2) Elle donne comme résultat une solution de base réalisable qui peut fort bien être très éloignée de l'optimum du programme linéaire initial. Au contraire, la méthode proposée ici peut par application de $P_2$ ou de toute autre politique analogue, tenir compte de l'ensemble des contraintes de $C_k^-$, mais *aussi des coûts marginaux associés à la fonction économique z.* Il suffit en effet de faire intervenir $z$ comme une contrainte de $C_k^-$.

Remarquons, cependant, que le traitement des *égalités* par cette méthode a toute chance d'être moins bon que l'adjonction classique d'une seule variable artificielle.

**Implémentation du programme sur calculateur**

La place mémoire utilisée est de l'ordre de $(n + 1)(m + 1)$ cellules mémoires.

Lorsqu'une solution de base réalisable pour $C$ a été trouvee, on substitue à la dernière contrainte $p$ la fonction objectif $z = \sum_{j=1}^{n} c_j x_j$ dans la base correspondante et l'on enchaîne l'optimisation simpliciale classique, ce qui revient pratiquement à une méthode en une phase, quel que soit le problème.

Un programme Fortran a été écrit à partir de cette méthode. Il utilise les politiques $P_1$ et $P_2$ avec $\alpha = 0{,}1$. Il a été testé sur le programme linéaire (33 × 33) présenté dans [2] pages 335, 374.

La première solution réalisable a été trouvée au bout de 36 itérations et vaut $z_1 = 93\,659{,}996$ alors que l'optimum, atteint au bout de 64 pivotages, vaut 89 853,137.

Ce programme linéaire n'est pourtant pas très favorable à la méthode proposée ici, car il comporte 15 égalités. Cependant, la méthode classique en deux phases trouve la première solution réalisable en 48 itérations et l'optimum en 68 itérations.

[1] F. P. FISCHER, *Speed-up the solution to Linear Programming Problems.* The Journal of Ind. Engineering, Vol. XII, n° 6, nov-dec. 1961.

[2] H. MAURIN, *Programmation linéaire appliquée*, Technip, 1967.

[3] G. B. DANTZIG, *Linear Programming and Extensions*, Ed. Princeton Univ. Press, 1963, pp. 245-247.