Seminaire Sogesci — Sogesci Seminarie
(fin - einde)

PROGRAMMATION LINEAIRE PAR L'ALGORITHME DE DICKSON ET FREDERICK SUR L'UNIVAC SOLID STATE COMPUTER

par R. BROUCKE
Remington Rand Bruxelles.

1. — Introduction.

La programmation linéaire continue toujours à jouer un rôle d'avant-garde dans la recherche opérationnelle. Elle ne peut jouer ce rôle que grâce à l'aide des calculateurs électroniques. C'est pourquoi de grands efforts ont déjà été faits dans le but de réduire le temps de résolution des programmes linéaires sur les calculateurs électroniques. D'un côté, les firmes s'attachent à produire des calculateurs de plus en plus rapides et d'un autre côté, des spécialistes cherchent des algorithmes mathématiques nouveaux permettant de réduire au minimum le nombre d'opérations élémentaires à effectuer pour atteindre la solution cherchée.

Le jeu de programmes qui a été préparé pour la Remington Rand par Bonner and Moore Engineering Associates à Houston, Texas, est basé sur une variante de l'algorithme simplexe classique de Dantzig. C'est une variante qui a été mise au point par J.C. Dickson et F.P. Frederick, tous deux de Bonner and Moore Engineering Associates. Ces programmes permettent d'obtenir des performances très bonnes sur le calculateur de grandeur moyenne qu'est l'Univac Solid State Computer, aussi bien du point de vue de la vitesse de calcul que du point de vue de la précision des calculs.

2. — Rappels sur l'algorithme du simplexe de Dantzig.

Nous donnons ici quelques brefs rappels sur l'algorithme de Dantzig, en utilisant au maximum des notations qui sont usuelles dans ce domaine. Pour un exposé systématique, le lecteur peut, par exemple, consulter le livre de Saul I. Gass [1].

Soit un programme linéaire exprimé par sa fonction économique linéaire en les n variables x_j et m égalités linéaires

$$\sum_{j=1}^{j=n} a_{0j}\ x_j = \text{MINIMUM} \qquad (1)$$

$$\sum_{j=1}^{s=n} a_{ij}\ x_j = a_{i0}\,, \qquad (i = 1, 2, \ldots, m) \qquad (2)$$

Les méthodes dites de Dantzig résolvent le problème d'une façon itérative et chaque itération consiste fondamentalement en trois étapes. L'ensemble des itérations part d'une solution de base admissible formée de m variables et aboutit à une autre solution de base également formée de m variables, cette dernière étant la solution optimum. Chaque itération consiste à échanger deux variables sans jamais s'éloigner de l'optimum à atteindre. Les trois étapes qui constituent chaque itération sont dès lors :

— Choix de la nouvelle variable x_j à entrer dans la base.
— Choix de la variable x_i qui quitte la base.
— Changement de base proprement dit.

Si $Z(0)$ est l'ancienne valeur de la fonction économique, c'est-à-dire celle avant une itération, la nouvelle valeur $Z(1)$ est donnée par la formule

$$Z(1) = Z(0) - a_{0j} \frac{a_{i0}}{a_{ij}}. \tag{3}$$

La variable x_j qui vient de rentrer dans la nouvelle base prend la valeur

$$x_j = \frac{a_{i0}}{a_{ij}}, \tag{4}$$

et si on ne considère que des solutions « admissibles » a_{i0} est non négatif et a_{ij} est positif.

La relation (3) montre qu'au cours de l'itération la fonction économique diminue ou est stationnaire à condition que le « coût marginal »

$$Z_j - C_j = a_{0j} \tag{5}$$

soit non négatif.

Il existe une grande variété d'algorithmes dits de Dantzig, mais tous ont la propriété de choisir d'une façon ou d'une autre i et j tels que

$$a_{0j} \frac{a_{i0}}{a_{ij}} \geqslant 0 \tag{6}$$

(dans le cas de la recherche du minimum).

Tous ces algorithmes sont quelque peu empiriques, en ce sens qu'on ne connaît pas le moyen de choisir à chaque itération une variable entrante x_j dont on sera certain de la trouver dans la solution finale optimale. La propriété directrice exprimée dans (6) est, en termes de la géométrie du polyèdre des solutions de base, une propriété locale liée à un sommet, mais les propriétés globales qui définiraient à priori les variables présentes dans

la base finale sont inconnues. De là l'existence d'un grand nombre d'algorithmes qui servent tous à se frayer un chemin le long des arêtes du polyèdre des solutions de base, vers la solution optimale. Tous ces chemins ont la propriété commune de ne jamais faire marche arrière du point de vue de la valeur de la fonction économique.

3. — Interprétation géométrique de l'algorithme de Dantzig.

L'algorithme de Dickson et Frederick rentre dans la catégorie des algorithmes que nous dirons de Dantzig, décrits ci-dessus. Il se base toujours sur l'inégalité (6), mais avant de le décrire il nous faut dire quelques mots d'une interprétation géométrique intéressante du programme linéaire.

On peut considérer un espace euclidien E_{m+1}, à $(m + 1)$ dimensions, et associer à chacune des n variables x_j un vecteur $\overrightarrow{P_j}$ dont les composantes sont

$$(a_{0j}\ ,\ a_{1j}\ ,\ a_{2j}\ ,\ \ldots,\ a_{mj})\ . \tag{7}$$

L'ensemble de ces vecteurs $\overrightarrow{P_j}$ de E_{m+1} forme un cône, et les composantes de ces vecteurs suivant le zéroième axe ont la valeur

$$Z_j - C_j = a_{0j} = |\overrightarrow{P_j}|\ .\ \cos\ \beta_j\ , \tag{8}$$

β_j étant l'angle du vecteur $\overrightarrow{P_j}$ avec l'axe d'indice zéro.

On peut encore écrire pour cet angle β_j :

$$\cos^2 \beta_j = a^2_{0j} \Big/ \sum_{i=0}^{m} a^2_{ij}\ . \tag{9}$$

4. — Variantes de l'algorithme de Dantzig .

Enumérons maintenant quelques variantes d'algorithmes de Dantzig.

1. On vérifie (6) en choisissant j au hasard, mais tel que l'expression (8) soit non négative. Cet algorithme de sélection, qui est très simple, a été employé, mais on constate qu'il conduit généralement à un nombre d'itérations élevé.
2. Il est possible aussi de choisir j et i simultanément : pour chaque candidat x_j, c'est-à-dire tel que $Z_j - C_j$ est non négatif, on cherche i suivant la règle classique, et on prend finalement le couple (j, i) tel que le premier membre de (6) soit maximum.
   Ce procédé conduit à un nombre d'itérations beaucoup plus petit que le procédé 1, mais ici la recherche du pivot a_{ij} nécessite un grand

nombre de multiplications et de divisions, et allonge donc les temps totaux.

3. L'algorithme de choix de x_j le plus employé est celui qui consiste à prendre j tel que (8) soit non négatif et maximum. Le nombre d'itérations, avec cet algorithme, est de l'ordre de $2\,n$.
4. Dickson et Frederick ont expérimenté avec un algorithme de sélection qui consiste à prendre j tel que $(Z_j - C_j)$ soit non négatif et que β_j soit minimum, c'est-à-dire que $\cos^2 \beta_j$ dans (9) soit maximum.
   Il s'est avéré que cette technique donne des résultats aussi bons qu'au 3, du point de vue du nombre d'itérations. Du point de vue de la précision, il y a peut-être une amélioration pour la raison suivante :

Les changements de base effectués sont géométriquement des rotations où l'angle β_j joue un rôle, et les erreurs d'arrondi introduites dans ces rotations sont plus ou moins proportionnelles à l'angle β_j. L'algorithme 4, utilisant des angles β_j minima, introduira donc aussi des erreurs d'arrondi minimales.

5. — L'algorithme de Dickson et Frederick.

Le véritable algorithme de Dickson et Frederick est une modification de l'algorithme précédent. Il consiste à se servir d'un faux angle β_j, soit $\overline{\beta_j}$, défini par

$$\cos^2 \overline{\beta_j} = a^2_{oj} \,/\, (a^2_{oj} + \sum_{i=1}^{m}{}^{(+)}\ a^2_{ij}) , \tag{10}$$

L'indice $(+)$ du sigma sommatoire indiquant qu'on ne prend dans la sommation que les termes a^2_{ij} tels que $a_{ij} > 0$. Cette modification a le grand avantage de réduire de moitié le nombre total d'itérations à effectuer. Et ce fait peut se comprendre de la façon suivante :

A chaque itération, les variables dans la base x_i sont exprimées en fonction des variables hors base x_j par des relations de la forme

$$x_i = a_{i0} - \sum_{j=1}^{j=n} a_{ij}\ x_j . \tag{11}$$

Comme x_i ne peut pas devenir négatif, chaque x_j candidat à rentrer dans la base est limité seulement par les a_{ij} positifs. Les a_{ij} négatifs n'imposent pas de restrictions aux augmentations de x_j. Dès lors, l'introduction de l'angle $\overline{\beta_j}$ minimum, par (10), consiste à favoriser les variables x_j, candidates à entrer dans la base, qui ont peu de limitations de la forme (11). Ces variables x_j ont ainsi des chances de croître très fort et, par là, de faire décroître très fort la fonction économique.

Nous gardons toujours ici l'avantage d'introduire des erreurs d'arrondi relativement petites, tout en ayant un nombre faible d'itérations.

On peut à juste titre objecter que le calcul de l'indice j en se basant sur la formule (10) est assez long. En fait, la formule (10) contient un grand nombre d'additions et de multiplications, en plus d'une seule division. Elle sera donc indiquée pour les machines électroniques qui ont la division assez longue, mais les autres opérations élémentaires plus rapides. Toujours est-il que le temps moyen d'une itération est allongé de 10 % par l'emploi de la formule (10), comparativement aux autres algorithmes classiques. D'autre part, selon les auteurs de la méthode, le nombre total d'itérations à effectuer est diminué de 30 à 70 %. Globalement, le temps de calcul est donc toujours diminué. Dickson et Frederick citent les deux exemples pratiques suivants, se rapportant à des problèmes de raffinerie :

1) *34 équations, 44 variables :*
   43 itérations au lieu de 64.
2) *62 équations, 70 variables :*
   62 itérations au lieu de 85.

6. — Utilisation de l'algorithme de Dickson et Frederick sur l'Univac Solid State Computer.

Rappelons que l'Univac Solid State Computer est un calculateur de type moyen. Il possède une mémoire générale de 5000 mots de 10 chiffres, sur un tambour tournant à 17.667 tours par minute. Pour 4000 mots de ce tambour, il y a une seule tête de lecture par piste, et pour les 1000 mots restants, il y a 4 têtes par piste. Il y a ainsi 4000 mots à temps d'accès moyen de 1700 microsecondes et 1000 mots à temps d'accès moyen de 425 microsecondes. Dans les deux zones de mémoire, on loge indifféremment des données ou des instructions de programme.

Pour l'application aux programmes linéaires, l'organisation de mémoires utilisée est la suivante :

1) de 4800 à 5000 se trouve en permanence une routine de chargement.
2) de 4300 à 4800 se trouvent tous les programmes et zones de calcul.
3) de 0 à 4300 se trouve la matrice du programme linéaire. Les dimensions maxima de la matrice sont exprimées par

$$m < 100\,, \qquad n < 100\,, \qquad m \,.\, n < 4.300 \tag{12}$$

La matrice unité correspondant aux variables d'écart positives et aux variables artificielles n'est jamais rangée dans ces 4300 mémoires, étant donné qu'elle est simulée par le programme.

Le programme convient pour des problèmes directs, duaux ou composites, les problèmes composites étant d'abord ramenés à des problèmes duaux par le programme.

Etant donné le degré de perfectionnement du programme, il est impossible de le loger dans les 500 instructions de 4300 à 4800. Aussi se compose-t-il de plusieurs paquets de cartes qui sont alternativement lues et exécutées. Le programme complet se compose d'environ 500 cartes de 5 instructions et il est lu par la machine en un temps total d'une minute et 10 secondes. La résolution complète d'un programme linéaire se fait en les étapes suivantes, qui s'enchaînent automatiquement :

— lecture de la routine de chargement.
— lecture du programme de chargement de la matrice.
— lecture de la matrice.
— lecture du premier programme de calcul. Ce programme fait quelques calculs initiaux et détermine le genre du programme linéaire.
— lecture avec exécution éventuelle des programmes de calcul 2 3 et 4; avec une ligne d'impression à chaque itération.
— lecture du programme « Vecteur solution ».
— interprétation et impression de la solution optimale.
— lecture du programme « Coûts marginaux ».
— interprétation et impression des Coûts marginaux.
— lecture du programme de perforation de la matrice.
— perforation de la matrice optimale.

Tous les calculs sont faits en virgule fixe, avec toujours 4 entiers et 6 décimales. Ce fait contribue encore à augmenter la vitesse de résolution, mais, par contre, exige que les données soient d'abord convenablement normalisées.

L'ensemble des caractéristiques que nous venons de décrire a comme conséquence que l'Univac Solid State Computer est capable de traiter les programmes linéaires avec un rendement élevé : par exemple, il faut de quatre à cinq minutes pour résoudre un programme linéaire à 30 équations, 40 variables et 30 itérations.

REFERENCES

[1] S.I. GASS, *Linear Programming, Methods and Applications.* McGraw Hill Book Company, New-York, Toronto, London, 1958.
[2] J.C. DICKSON, F.P. FREDERICK, A Decision Rule for Improved Efficiency in solving Linear Programming Problems with the Simplex Algorithm. *Communications of A.C.M.* (Volume 3). N° 9, September 1960, pages 509 à 512.