REVUE FRANÇAISE D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. SÉRIE VERTE

J. TERGNY

Programmation linéaire mixte : conditions de validité dans les méthodes arborescentes

Revue française d'informatique et de recherche opérationnelle. Série verte, tome 5, n° V1 (1971), p. 11-21

<http://www.numdam.org/item?id=RO_1971__5_1_11_0>

R.I.R.O.
(5e année, V-1, 1971, p. 11-21)

PROGRAMMATION LINEAIRE MIXTE : CONDITIONS DE VALIDITE DANS LES METHODES ARBORESCENTES

par J. TERGNY (1)

Résumé. — *Dans les méthodes arborescentes pour la Programmation linéaire Mixte, une analyse simple du dernier tableau de base associé à chaque sous-programme « terminal » permet d'obtenir une condition nécessaire pour que la résolution d'un sous-programme quelconque puisse fournir une solution meilleure que celles déjà connues. Ces conditions nécessaires, ou « conditions de validité » prennent la forme de contraintes linéaires liant les variables entières. S'il s'agit d'un outil classiquement employé dans les méthodes de résolution par décomposition, l'idée de l'appliquer aux méthodes arborescentes est plus récente (voir* [4]).

I. INTRODUCTION

Indirectement ce travail est le fruit de la tentative de classification des méthodes de PL mixte présentée dans [7] et où l'on avait cru pouvoir distinguer trois types « purs » d'algorithmes, à savoir :

Approche 1 : *méthodes de troncations duales* (Gomory, Glover, Hu, etc.).

Approche 2 : *méthodes d'exploration arborescente des stratégies partielles* (2) (Land et Doig, Beale et Small, etc.). Dans ce qui suit on supposera que ce type de méthode, au moins dans ses grandes lignes, est familier au lecteur. C'est l'approche retenue pour l'élaboration de la plupart des gros codes opérationnels : UMPIRE, OPHELIE MIXTE, MPS. On peut en trouver une description claire dans [5] ou [8] ou [3] ou bien encore [6], tous ces papiers contiennent en outre des résultats numériques relatifs à des problèmes concrets.

Approche 3 : *méthodes d'énumération des stratégies globales* (2) (Benders [2], E. Balas [1], etc.).

---

(1) Direction Scientifique, SEMA (Metra International).

(2) Considérant les variables entières comme les variables « stratégiques » du problème, on dira ici qu'on a défini une *stratégie globale* lorsqu'on a assigné une valeur entière à chacune d'elle. On parlera de *stratégie partielle* lorsqu'une partie seulement des variables entières est assignée à une valeur donnée (ou doit figurer dans un intervalle de variation plus étroit que l'intervalle initial).

Les approches 1 et 2

Nous pensons que deux des trois types d'approches qui viennent d'être mentionnés sont en fait parallèles : non pas les types 2 et 3, qui présentent seulement des analogies superficielles, mais bien les types 1 et 2, basés sur l'utilisation de la même propriété élémentaire, qu'on rappelle maintenant :

Si la (ou l'une des) solution optimale du programme continu associé (1) *à un programme mixte se trouve vérifier les conditions d'intégrité cette solution est optimale pour le problème mixte.*

Dans les approches 1 et 2, chaque itération conduit à l'examen de la base optimale d'un problème continu. Lorsque celle-ci n'est pas entière (ce qui est le cas le plus fréquent) il faudra :

— dans l'approche 1 : adjoindre au problème continu une contrainte supplémentaire telle que la base précédente ne soit plus admissible et que toutes les solutions entières vérifient la nouvelle contrainte;

— dans l'approche 2 : construire deux (ou plusieurs) sous-problèmes (chacun défini par l'adjonction d'une contrainte supplémentaire) tels que toute solution entière soit admissible pour l'un de ces sous-problèmes. Le plus souvent les contraintes supplémentaires sont choisies telles que toute solution entière ne soit admissible que pour l'un seulement des sous-programmes, et que la solution optimale du programme précédent ne soit admissible pour aucun d'eux.

Les deux approches diffèrent en fait par la façon dont est mémorisé l'état de l'exploration à une itération donnée. Dans la première il s'agit d'une série de contraintes additionnelles réduisant le domaine d'admissibilité attaché au programme continu associé tandis que dans la seconde on construit une arborescence dont chaque sommet pendant définit un sous-programme. Deux raisons pratiques font que cette dernière représentation nous paraît actuellement préférable :

— les codes de programmation linéaire se prêtent malaisément à l'adjonction de lignes supplémentaires,

— l'approche 1 ne fournit en aucun cas de solutions intermédiaires entières admissibles, handicap rédhibitoire lorsqu'il s'agit de traiter des problèmes de grande taille. Par contre, avec l'approche 2, on peut parfois obtenir des solutions intermédiaires entières suffisamment proches de l'optimum (2) pour qu'on puisse raisonnablement interrompre les calculs à ce stade de l'exploration.

---

(1) On appelle programme continu associé à un programme mixte, le programme continu obtenu en négligeant les conditions d'intégrité.

(2) Plus exactement : proche d'une borne supérieure (respect. inférieure) de cet optimum, s'il s'agit d'un problème de maximisation (respect. minimisation).

L'approche 3

La plus intéressante des méthodes de ce type est certainement l'algorithme de décomposition, dû à Benders. A chaque itération on choisit une stratégie globale afin de se ramener à un programme où figurent uniquement les variables continues (le niveau choisi pour les variables entières est pris en compte par un remaniement convenable du second membre). L'itération est un succès si le programme continu fournit une solution optimale meilleure que celles obtenues lors des itérations précédentes. A notre avis cette approche présente tout à la fois une faiblesse (le fait de choisir *simultanément* une valeur pour chacune des variables entières ne permet pas d'utiliser la propriété énoncée précédemment) et une caractéristique remarquable : de chaque itération, succès ou échec, on tire une condition nécessaire pour qu'une stratégie globale puisse fournir un succès; c'est dire que des *calculs intermédiaires déjà effectués on déduit une information permettant de réduire le champ d'exploration.* Dans le paragraphe qui suit on montre comment construire de telles conditions nécessaires, dites *conditions de validité*, dans le cadre d'une approche de type 2, c'est-à-dire lorsque les variables entières figurent explicitement comme colonnes du programme.

II. LES CONDITIONS DE VALIDITE

Définitions et notations

Dans le seul but d'alléger l'exposé on supposera ici que toutes les variables entières sont bivalentes. Le programme continu associé au programme mixte s'écrira alors :

$$\max x_0 = cx$$

sous les conditions :

$$\begin{aligned} Ax &= b \\ x_j &\geqslant 0 \qquad j \in J_c \\ 0 &\leqslant x_j \leqslant 1 \qquad j \in J_b \end{aligned}$$

où A est une matrice $(m \times n)$ et où $J_c(J_b)$ représente l'ensemble d'indices des variables continues (bivalentes).

Étant donné une base quelconque de ce programme, soit B, (N) l'ensemble d'indices des variables basiques (non basiques); si l'on choisit la formulation du simplexe révisé la forme canonique associée à cette base s'écrira :

$$x_i + \sum_{j \in N_c} y_{ij} x_j + \sum_{j \in N_b} y_{ij} X_j = t_i \qquad (i \in B)$$

$$\text{où } N_c = N \cap J_c \; ; \; N_b = N \cap J_b \; ; \; X_j = \begin{cases} x_j \text{ si } x_j = 0 \\ 1 - x_j \text{ si } x_j = 1 \end{cases}$$

et où y_{ij}, t_i représentent les coefficients « mis à jour » des colonnes des variables non basiques et du second membre (1).

L'approche arborescente conduit à résoudre un sous-programme particulier à chaque itération. Pour ce faire on part de la base optimale du sous-programme qui précède, dans l'arborescence, celui à traiter. Cette base figure parmi les bases admissibles du programme continu associé et elle présente les caractéristiques suivantes :

— l'une au moins des variables de base, soit x_{io}, est une variable bivalente prenant une valeur non entière dans la solution de base,

— les variables entières non basiques sont séparées en deux sous-ensembles : variables « libres » et variables « arbitrées ». On notera N_b' et N_b'' les sous-ensembles d'indices correspondant.

Le sous-programme courant s'écrit alors :

$$\max x_0 = cx$$

sous les contraintes :

$$(1) \qquad \begin{cases} Ax = b \\ x_j \geqslant 0 & j \in J_c \\ 0 \leqslant x_j \leqslant 1 & j \in J_b \end{cases}$$

$$(2) \qquad X_j = 0 \qquad J \in N_b''$$

$$(3) \qquad X_{io} = 0 \text{ (avec soit } X_{io} = x_{io}, \text{ soit } X_{io} = 1 - x_{io})$$

où (1) représente les contraintes initiales, (2) les contraintes supplémentaires définissant le programme qui précède, dans l'arborescence, le programme courant, (3) correspond à une condition d'intégrité supplémentaire non vérifiée par la solution optimale du programme précédent.

Le sous-programme courant sera généralement résolu soit par la méthode duale, soit par une méthode de paramétrisation, ceci puisqu'on dispose d'une base de départ duale-admissible qui est la base optimale du sous-programme précédent.

Dans tout ce qui suit on supposera que les sous-programmes sont effectivement résolus à l'aide de l'une de ces deux méthodes, c'est-à-dire que les bases intermédiaires sont duale-admissibles pour le sous-programme à traiter.

Si la résolution du sous-programme courant conduit à la génération de deux

---

(1) Si $\mathfrak{B}$ représente la matrice de base, a^j une colonne de A et b le second membre on a :

$$y = (y_{0j}, y_{1j}, \ldots, y_{mj})' = \mathfrak{B}^{-1}a^j$$

et

$$t = (t_0, t_1, \ldots, t_m)' = \mathfrak{B}^{-1}b.$$

nouveaux sous-programmes, le sous-programme sera dit *non-terminal.* Il sera dit *terminal* si l'une des trois éventualités suivantes se produit :

1) La solution optimale vérifie toutes les conditions d'intégrité et se trouve être la meilleure des solutions entières obtenues jusqu'à présent.

2) Le sous-programme n'a pas de solution.

3) Le sous-programme ne peut contenir la solution optimale du programme mixte (la fonction économique se dégrade, en cours de résolution, jusqu'à un niveau inférieur à la valeur de la meilleure solution entière connue).

Dans chacun de ces cas, l'examen de la forme canonique associée à la dernière base générée nous permet de formuler l'une des conditions de validité (de type 1, 2 ou 3) qui sont présentées maintenant.

Condition de validité de type 1

Parmi l'ensemble des variables bivalentes X_j hors-base et non libres $(J \in N_b'' + \{io\})$ soit J^- le sous-ensemble de celles qui vérifient :

$$y_{0j} < 0$$

La solution entière obtenue reste optimale tant qu'on ne remet pas en cause le niveau des variables $X_j (j \in J^-)$. Ainsi :

Si $J^- = \varnothing$ cette solution est l'optimum cherché (c'est le cas particulier où le problème continu associé admet une plage d'optimalité dont l'un des sommets vérifie les conditions d'intégrité).

Si $J^- \neq \varnothing$ seuls les sous-programmes pouvant vérifier la condition de validité de type 1 :

(C1) $$\sum_{j \in J^-} X_j \geqslant 1$$

sont susceptibles de contenir une solution meilleure que la solution courante.

De même l'inégalité :

(C1') $$\sum_{j \in N_b} y_{0j} X_j < 0$$

est également une condition de validité. En effet les solutions meilleures que celle qui vient d'être obtenue sont celles dont les composantes :

$$x_j (j \in N_c) \quad \text{et} \quad X_j (j \in N_b)$$

vérifient l'inégalité :

$$\sum_{j \in N_c} y_{0j} x_j + \sum_{j \in N_b} y_{0j} x_j < 0$$

d'où *a fortiori* (C1') puisque l'on a :

$$y_{0j} \geqslant 0 \quad (j \in N_c)$$

Au cours des itérations ultérieures, chaque fois qu'il sera rencontré une solution admissible de valeur supérieure, d'une quantité : $\Delta\bar{f}$, à la valeur de la meilleure des solutions précédemment connues, cette quantité $\Delta\bar{f}$ sera évidemment retranchée au second membre de (C1').

On remarque que si l'on se proposait d'utiliser les conditions de validité comme des contraintes additionnelles, (C1) serait préférable à (C1'). Par contre si on les emploie en tenant compte de l'intégrité des (X_j), (C1') est alors une condition plus fine que (C1).

Condition de validité de type 2

Dans le cas de détection d'un domaine vide, l'équation de la forme canonique qui fournit la ligne de pivot s'écrit :

$$x_r + \sum_{j \in N_c} y_{rj} x_j + \sum_{j \in Nb'} y_{rj} X_j + \sum_{j \in Nb''} y_{rj} X_j = t_r \tag{2}$$

où : $t_r < 0$ (1)

et $y_{rj} \geqslant 0 \qquad j \{ N_c \cup N_b' \}$

Dans ces conditions, considérons le signe des y_{rj} $(J \in N_b'')$.

Si tous ces coefficients sont également non négatifs, le programme mixte ne peut avoir de solutions vérifiant la condition (3) : $X_{io} = 0$. Et l'on posera définitivement : $(X_{io} = 1)$ dans la suite de l'exploration.

Sinon posons :

$$J^- = \{ j / j \in N_b'', y_{rj} < 0 \}$$

L'inégalité

$$\sum_{j \in J^-} X_j \geqslant 1$$

est alors condition nécessaire pour que :

$$X_{io} = 0$$

d'où la condition de validité :

$$X_{i0} + \sum_{j \in J^-} X_j \leqslant 1 \tag{C2}$$

L'intérêt de (C2) réside exclusivement dans la simplicité de sa formulation. Il existe en effet une condition plus fine. L'égalité (2) peut en effet s'écrire :

$$x_r + \sum_{j \in N_c} y_{rj} x_j + y_{rio} x i_o = t_r - \sum_{j \in Nb - \{io\}} y_{rj} X_j$$

de $y_{rj} \geqslant 0 \quad (j \in N_c)$

(1) Ceci si l'on utilise la méthode duale. Si l'on utilise une méthode de paramétrisation, il y a lieu de remplacer dans ce qui suit (t_r) par $(t_r + \theta_{max} t_r')$ où t_r' représente le coefficient mis à jour du second membre différentiel et θ_{max} la valeur du paramètre assurant la condition (3).

on tire l'inégalité :

$$t_r - \sum_{j \in N_b - \{io\}} y_{rj} X_j \geqslant 0$$

condition nécessaire pour qu'une solution admissible puisse vérifier : $X_{io} = 0$.

D'où la condition :

(C2') $$M X_{io} + \sum_{j \in N_b - \{io\}} y_{rj} X_j \leqslant t_r$$

où M est une constante négative suffisamment grande en valeur absolue.

$$\left(M \leqslant t_r - \sum_{j \in N_b} \max (y_{rj}, 0) \right)$$

Condition de validité de type 3

Notons $\bar{f}$ la valeur donnée à la fonction économique par la meilleure solution entière rencontrée. (On posera $\bar{f} = -\infty$ si l'on ne connaît pas encore de solution entière admissible.)

Dans tout sous-programme, la base courante sera telle que :

$$t_0 \geqslant \bar{f}$$

Soit r la ligne choisie comme ligne de pivot. Si le sous-programme admet une solution admissible, il existe au moins un coefficient y_{rj} négatif, avec $j \in \{ N_c \cup N_b' \}$.

La variable devant entrer dans la base (dont l'indice sera noté k) sera celle (ou l'une de celles) dont les coefficients vérifient :

$$\frac{y_{ok}}{y_{rk}} = \max_{j \in \{N_c \cup Nb'\}} \left(\frac{y_{oj}}{y_{rj}} \right)$$

$$y_{rj} < 0$$

Dans le cas où :

$$\frac{y_{ok}}{y_{rk}} t_r > t_0 - \bar{f}$$

le domaine d'admissibilité du sous-programme courant ne peut contenir la solution optimale.

Pour tout $j \in N_b''$ et tel que : $y_{rj} < 0$

calculons alors :

$$\Delta j = \begin{cases} \max \left(y_{oj}, \dfrac{y_{oj}}{y_{rj}} t_r \right) & \text{si } y_{oj} > 0 \\ y_{oj} & \text{si } y_{oj} \leqslant 0 \end{cases}$$

et posons :

$$J^- = \{j/j \in N_b'', y_{rj} < 0, \Delta_j < t_0 - \bar{f}\}$$

Si $J^- = \varnothing$ il n'existe pas de solution entière de valeur supérieure ou égale à $\bar{f}$ et vérifiant la condition (3) : $X_{io} = 0$. On posera définitivement $(X_{io} = 1)$ dans la suite de l'exploration.

Si par contre $J^- \neq \varnothing$, on obtient la condition de validité de type 3 :

(C3) $$X_{io} + \sum_{j \in J^-} X_j \geqslant 1$$

ou la condition plus fine :

$$t_r X_{io} + \sum_{j \in J^-} y_{rj} X_j \leqslant t_r$$

comme tous les coefficients de cette inégalité sont négatifs et comme toutes les variables y figurant doivent être entières, cette condition peut s'écrire :

(C3') $$t_r X_{io} + \sum_{j \in J^-} z_{rj} X_j \leqslant t_r$$

avec $z_{rj} = \max(y_{rj}, t_r)$.

Arbitrages supplémentaires

Terminons en indiquant une autre procédure simple permettant de raccourcir l'exploration en utilisant l'information obtenue lors des calculs intermédiaires.

A une itération quelconque de la résolution du sous-programme courant (sous-programme terminal ou non), il peut exister des variables hors-base libres X_j telles que :

(3) $$y_{oj} > t_0 - \bar{f}$$

Alors les solutions entières du sous-programme qui vérifient $(X_j = 1)$ ne peuvent être optimales.

Ceci n'exclut pas qu'une telle variable puisse devenir basique non nulle dans la solution optimale de ce sous-programme (ou de certains de ceux qu'il engendrera).

On pourra alors être amené à séparer sur cette variable, séparation inutile puisque la branche $(X_j = 1)$ conduit nécessairement soit à une base terminale de type 3 si le domaine d'admissibilité correspondant est non vide, soit à une base terminale de type 2 sinon.

Ce phénomène peut être évité en interdisant l'entrée dans la base de toute variable entière X_j vérifiant (3). L'interdiction est valable lors de toute itération ultérieure du sous-programme courant et de tous ceux qu'il engendrera. Pour ce faire, il suffira de transférer l'indice j de l'ensemble N_b' à l'ensemble N_b''.

III. METHODE ARBORESCENTE AVEC CONDITIONS DE VALIDITE

Dans ce qui suit on montre quelles modifications doivent être apportées à un algorithme arborescent afin d'utiliser l'information que procurent les conditions de validité. On prendra l'exemple de l'algorithme du code OPHELIE MIXTE [8], les modifications apportées étant indiquées en italiques.

Algorithme

0. Faire $\bar{f} = -\infty$. Initialiser la liste des programmes à résoudre en y intégrant le programme continu associé. *Réserver un tableau annexe où figureront les conditions de validité générées ultérieurement.*

1. Résoudre le programme après l'avoir éliminé de la liste. Si ce programme est non-terminal, aller en 2. *Sinon mettre à jour les coefficients convenables dans les colonnes des variables arbitrées afin de déterminer la nouvelle condition de validité. Enregistrer celle-ci dans le tableau annexe.* Si le programme est de type 2 ou 3, aller en 3.

Sinon le sous-programme est terminal de type 1, faire $\bar{f} = t_0$ et enregistrer la solution de base correspondante : $\bar{X}$. Éliminer de la liste les programmes dont l'évaluation (1) est inférieure où égale à $\bar{f}$ (s'il en existe).

Mettre à jour, dans le tableau annexe, les seconds membres des conditions de validité de type 1 *(s'il en existe).* Aller en 3.

2. Choisir la variable de séparation x_{io}. Les conditions supplémentaires : $(x_{io} = 0)$ et $(x_{io} = 1)$ définissent chacune l'un des programmes générés à partir de celui qui vient d'être résolu. Intégrer chacun de ces programmes à la liste, *à la condition que les niveaux des variables arbitrées (x_{io} inclue) puissent être compatibles avec toutes les conditions de validité.*

3. Si la liste des programmes à résoudre est vide, aller en 5. Sinon choisir un programme dans cette liste.

4. *Si le niveau des variables arbitrées dans le programme choisi n'est pas incompatible avec les conditions de validité,* aller en 1. *Sinon éliminer ce programme de la liste et aller en* 3.

5. Si $\bar{f}$ a une valeur finie, la solution associée $\bar{X}$ est l'optimum cherché. Sinon le programme mixte n'admet pas de solution.

---

(1) L'évaluation d'un programme est par définition une borne supérieure de la valeur de l'optimum de celui-ci. Pour obtenir une évaluation sans avoir à résoudre effectivement le programme, il suffira, lors de la génération de celui-ci, de retrancher une pénalité convenable à la valeur optimale du programme précédent (voir [6] ou [8]).

Le tableau annexe est une matrice dont le nombre de colonnes égale le nombre de variables bivalentes du programme, le nombre de lignes augmentant progressivement et étant égal lors d'une itération donnée au nombre de programmes terminaux rencontrés lors des itérations précédentes.

Calculs attachés au pas 4 de l'algorithme

Les conditions de validité étant notées maintenant :

$$\sum_{j \in J_b} p_{kj} x_j \geqslant q_k \qquad (1 \leqslant k \leqslant K)$$

on désignera par v_j^s les valeurs des variables arbitrées dans le sous-programme (s) examiné (variable de séparation x_{io} inclue). Soit J_b^s l'ensemble d'indices de ces variables.

Posons :

$$f_k^s = \sum_{j \in Jbs} p_{kj} v_j^s + \sum_{j \in [J_b - Jbs]} \max(p_{k_1}, 0) - q_k$$

S'il existe au moins un f_k^s négatif, le sous-programme s sera alors éliminé.

Calculs supplémentaires attachés au pas 2

Soit x_i une variable candidate pour la séparation $(i \notin J_b^s)$. Pour tout k on calculera :

$$f_k^{s(i,0)} = \sum_{j \in Jbs} p_{kj} v_j^s + \sum_{\substack{j \in [J_b - Jbs] \\ j \neq i}} \max(p_{kj}, 0) - q_k$$

et :

$$f_k^{s(i,1)} = f_k^{s(i,0)} + p_{k,i}$$

— S'il existe au moins k' tel que : $f_k^{s(i,0)} < 0$ et k'' tel que : $f_k^{s(i,1)} < 0$, le sous-programme (s) sera éliminé sans être séparé et on ira au pas 3.

— Si les inégalités sont vérifiées pour l'un seulement des deux niveaux, on ne génèrera que le sous-programme correspondant.

APPLICATIONS

1) On remarquera tout d'abord qu'une condition de validité sera d'autant plus efficace que l'inégalité la traduisant comprendra peu de coefficients non nuls. Aux limites :

— si un seul coefficient est non nul, la condition s'écrira : $x_i = 0$ (ou $x_i = 1$) et l'exploration est réduite de moitié;

— si par contre l'ensemble N_b'' des variables arbitrées en un sommet terminal figurent avec un coefficient positif dans la condition de validité atta-

chée à ce sommet, l'efficacité sera faible (dans le cas particulier où cette contrainte sera de la forme :

$$\sum_{j \in Nb''} x_j \geqslant 1$$

l'information apportée sera nulle.

Aussi peut-on penser que l'algorithme proposé sera surtout utile pour la résolution des programmes mixtes de grande taille, comportant un nombre élevé de lignes et relativement peu de variables bivalentes, programmes pour lesquels la résolution de chaque sous-programme peut demander un temps considérable. Car le temps nécessaire à la rédaction et à la consultation des conditions de validité étant uniquement fonction du nombre de variables bivalentes, il suffira que les conditions de validité permettent d'éviter la résolution de quelques sous-programmes pour qu'on ait réalisé globalement un gain de temps.

2) Lorsqu'on désire effectuer, avec une fonction économique différente, l'optimisation d'un programme déjà résolu et dans le cas où le domaine d'admissibilité est inchangé (ou contenu dans le précédent) on réduira l'exploration à effectuer en intégrant les conditions de validité de type 2 en tant que contraintes additionnelles du problème.

REFERENCES

[1] E. BALAS, *Duality in discrete programming*, Tech. Report n° 67-5, Dep. of Oper. Research, Stanford University, July 1967.

[2] J. F. BENDERS, « *Partitioning procedures for solving mixed-variables programming problems* », Numerische Math., 1962, 4, 238-252.

[3] M. BENICHOU, J. M. GAUTHIER, P. GIRODET, G. HENTGES, G. RIBIERE et O. VINCENT, *Mixed Integer Linear programming*. Communication présentée au 7e symposium de programmation mathématique, La Haye, 7-14 septembre 1970.

[4] M. GUIGNARD et K. SPIELBERG, *The State Enumeration Method for Mixed Zero-one programming*. Communication présentée au 7e symposium de programmation mathématique, La Haye, 7-14 septembre 1970.

[5] P. HERVE, « *Résolution des programmes linéaires à variables mixtes par la procédure S.E.P.* », Metra, VI, 1967, 77-91.

[6] G. MITRA, *Designing Branch-and-Bound Algorithms for Mathematical Programming*. Communication présentée au 7e symposium de programmation mathématique, La Haye, 7-14 septembre 1970.

[7] B. ROY, R. BENAYOUN, J. TERGNY et J. DE BUCHET, *Sur la programmation linéaire en variables mixtes* (Venise, 23-27 juin 1969), in : John Lawrence, editor Tavistock pub. « Procedings of the Fifth Int. Conference of Oper. Research ».

[8] B. ROY, R. BENAYOUN et J. TERGNY, *From SEP Procedure to the Mixed OPHELIE Program*, in : « Integer and Non Linear programming », J. Abadie, editor (chap. 20). North-Holland, 1970.