LES PROCEDURES D'OPTIMISATION PAR SEPARATION : PRESENTATION GENERALE

Pierre HANSEN
Institut d'Economie scientifique et de Gestion, Lille

1. Introduction.

1.1. Programmes mathématiques.

Une étape importante de la plupart des études de Recherche Opérationnelle consiste en la résolution d'un ou plusieurs problèmes d'optimisation; de tels problèmes, qui consistent à chercher l'extremum d'une fonction soumise à des contraintes, sont appelés *programmes mathématiques*.

Les programmes mathématiques peuvent être répartis en deux classes, selon que l'information disponible est plus ou moins complète :

a) *Problèmes posés de manière quantitative.*

L'information est suffisante pour que le problème soit posé de manière univoque; on dispose

- d'un critère unique, exprimé par une *fonction économique*;
- de *contraintes*, exprimées sous forme d'égalités ou d'inégalités.

On cherche la solution optimale, ou éventuellement toutes les solutions optimales, si elles sont plusieurs.

La résolution d'un problème de cette classe se fait à l'aide d'une *procédure d'optimisation* ou, autrement dit, d'un *algorithme*.

b) *Problèmes posés en partie de manière qualitative.*

L'information ne permet pas d'exprimer tous les aspects du problème sous forme mathématique; on dispose

- d'un ou de plusieurs critères, exprimés par des fonctions économiques; si les critères sont multiples, il est fréquent qu'ils ne puissent être agrégés en un seul de manière réaliste; parmi ces critères, on distingue éventuellement un critère principal et des critères secondaires;
- de contraintes, exprimées sous forme d'égalités ou d'inégalités, éventuellement après un codage.

On cherche un ensemble de *solutions satisfaisantes* des points de vue des critères et vérifiant les contraintes, ensemble parmi lequel un choix sera

fait ultérieurement, en tenant compte des aspects du problème qui n'ont pu être exprimés sous forme mathématique.

La résolution d'un problème de cette classe se fait à l'aide d'une *procédure d'exploration.*

Les procédures de séparation (P.S. en abrégé), apparues il y a une dizaine d'années, se sont révélées très utiles, tant pour résoudre des problèmes de la classe a) pour lesquels des techniques efficaces de résolution n'étaient pas disponibles, que pour traiter les problèmes de la classe b), difficiles à résoudre autrement. Selon la classe de problème envisagée, on parle de procédure d'optimisation ou de procédure d'exploration par séparation; les modifications à apporter à une procédure d'optimisation par séparation pour la transformer en procédure d'exploration par séparation sont minimes et l'étude des P.S. peut donc se faire globalement.

1.2. **Problèmes structurés et non structurés.**

Certains problèmes de programmation mathématique possèdent une structure susceptible d'être exploitée par un algorithme donnant rapidement la solution optimale; les programmes linéaires en sont un exemple : comme la fonction économique est linéaire et l'ensemble des solutions admissibles convexe, l'extremum est atteint en un point extrême de cet ensemble; cette propriété est exploitée par l'algorithme du simplexe qui permet de résoudre aisément de tels problèmes.

D'autres problèmes ne possèdent pas de structure et leur résolution nécessite des procédures plus complexes. C'est notamment le cas des programmes linéaires ou non linéaires pour lesquels une partie ou toutes les variables sont astreintes à prendre des valeurs entières; ces problèmes, qui portent le nom de *programmes mixtes* et de *programmes en entiers*, ont de très nombreuses applications.

1.3. **Approches des problèmes non structurés.**

A l'heure actuelle, trois approches permettent de résoudre de nombreuses classes de problèmes non structurés, pourvu que le nombre de variables et de contraintes ne soit pas trop élevé; on peut :

- construire un problème structuré dont la solution optimale coïncide avec celle du problème non structuré à résoudre (méthode des plans de coupure);
- énumérer toutes les solutions (énumération complète);
- séparer l'ensemble des solutions pour obtenir des sous-problèmes plus simples auxquels seront appliqués des tests (P.S.).

Les algorithmes de R. Gomory [10], [11], [12] illustrent la première approche dans les cas des programmes linéaires mixtes ou en entiers : on résout d'abord le programme linéaire obtenu en ne tenant pas compte des contraintes d'intégralité des variables; si la solution obtenue vérifie ces contraintes (ce qui serait dû au hasard, sauf dans le cas de programmes jouissant d'une structure très particulière), le problème est résolu; si ce n'est pas le cas, une contrainte additionnelle — le plan de coupure — est engendrée; cette contrainte exclut la solution optimale fractionnaire précédemment obtenue mais n'exclut pas de solution admissible qui vérifie les contraintes d'intégralité des variables; le programme linéaire modifié est résolu, et ainsi de suite. R. Gomory a montré qu'une telle procédure fournit en un nombre fini d'étapes un programme linéaire dont la solution optimale est aussi celle du programme mixte ou en entiers considéré. Les performances sur ordinateur des algorithmes utilisant cette approche laissent encore à désirer; le nombre de plans de coupure nécessaire peut être très grand, même pour des problèmes avec une dizaine de variables, et varie fortement d'un problème à l'autre. De nombreux travaux récents sur la manière optimale d'engendrer les plans de coupure [13], [14], [16], [3] permettent d'espérer des progrès dans cette voie.

Pour les problèmes en nombres entiers de petite taille, l'énumération complète est souvent la méthode de résolution la plus rapide; dès que le nombre de variables astreintes à être entières croît, cette approche exige un temps de calcul prohibitif.

Pour la plupart des problèmes non structurés, la troisième approche est actuellement la plus efficace, soit que les performances des P.S. soient supérieures à celles des méthodes des plans de coupure (cas des programmes mixtes), soit que les P.S. constituent la seule approche qui semble praticable (cas des programmes non linéaires et non convexes).

1.4. **Travaux théoriques sur les P.S.**

Dans le développement de la programmation linéaire comme dans celui de la programmation dynamique, un corps important de théorie a précédé ou accompagné le foisonnement des applications; le contraire s'est produit pour les P.S. : des algorithmes ont été proposés pour plusieurs classes particulières de problèmes avant que la généralité de l'approche ne soit reconnue et que les travaux théoriques ne débutent. Le premier exemple développé de P.S. est l'algorithme proposé en 1960 par A. Land et A. Doig [17] pour les programmes mixtes; en 1963, l'article de J.D.C. Little, K.G. Murty, D.W. Sweeney et C. Karel [19], utilisant une P.S. pour résoudre le problème

du voyageur de commerce, eut un grand retentissement; dès lors, les principes des P.S. étant bien dégagés, le nombre d'applications a crû rapidement. P. Bertier et B. Roy [8] ont donné en 1964 une première axiomatique des P.S., applicable à des problèmes de nature combinatoire; suivant cette axiomatique, on utilise une règle particulière pour l'enchaînement des séparations à laquelle correspond une classe de P.S. : les procédures d'optimisation ou d'exploration par séparation et évaluation progressive (P.S.E.P.). Récemment, B. Roy [21], [22] a modifié un des axiomes proposés, permettant ainsi de prendre en compte les P.S. pour les programmes mixtes. E. Balas [2] a proposé en 1968 une autre axiomatique, formalisant une manière de construire les fonctions d'évaluation et ce travail a été développé par L.G. Mitten [20]; P. Hervé [15] a proposé un cadre général pour toutes les P.S. D'autre part, des présentations générales des P.S. ont été données par N. Agin [1] et par E. Lawler et D. Wood [18], ce dernier travail comprenant également une revue des applications jusqu'en 1966. Les P.S. sont longuement décrites dans les articles de revue de la programmation en nombres entiers de M. Balinski [4], [5] et de M. Balinski et K. Spielberg [6].

La suite de cet article est consacrée à une présentation générale des P.S.; les axiomes proposés pour définir un *principe de séparation* sont assez proches de ceux de Bertier et Roy; les séparations sont définies sur l'ensemble des *solutions candidates* (voir ci-dessous) et non sur l'ensemble des solutions admissibles, ce qui paraît plus naturel; la démonstration du théorème de convergence est différente de celle donnée par Bertier et Roy et couvre toutes les classes de P.S. (procédures d'optimisation par séparation et évaluation progressive, par séparation et évaluation séquentielle et procédures mixtes). La définition des *fonctions d'évaluation* est plus générale que celle de Balas et Mitten; par contre, on ne considère par les P.S. infinies introduites par ce dernier auteur. On propose également une classification des tests utilisés dans les P.S., sujet non encore traité dans les travaux théoriques.

2. Définition des programmes mathématiques.

2.1. Forme canonique.

Par définition, résoudre un programme mathématique consiste à chercher le maximum ou le minimum d'une fonction économique soumise à des contraintes, exprimées sous forme d'égalités ou d'inégalités. Les programmes mathématiques peuvent être représentés sous la forme générale suivante, appelée forme canonique :

Minimiser

$$f(x_1, x_2, ..., x_n) \tag{1}$$

sous les contraintes

$$\begin{aligned} g_1(x_1, x_2, ..., x_n) &\geqslant 0 \\ g_2(x_1, x_2, ..., x_n) &\geqslant 0 \\ &\cdots\cdots\cdots\cdots \\ g_m(x_1, x_2, ..., x_m) &\geqslant 0 \end{aligned} \tag{2}$$

et

$$\begin{aligned} x_1 &\in D_1 \\ x_2 &\in D_2 \\ &\cdots\cdots \\ x_n &\in D_n \end{aligned} \tag{3}$$

En désignant par X le vecteur-colonne à n composantes $(x_1, x_2, ..., x_n)^T$, par G le vecteur-colonne à m composantes $(g_1, g_2, ..., g_m)^T$ et par D^n le produit cartésien $D_1 \times D_2 \times ... \times D_n$, le programme (1), (2), (3) peut s'écrire sous forme vectorielle :

Minimiser

$$f(X) \tag{4}$$

sous les contraintes

$$\begin{aligned} g(X) &\geqslant 0 \\ X &\in D^n \end{aligned} \tag{5}$$

où $X \in D^n$ est un vecteur d'inconnues ;

$D^n \in R_+^n$ est un sous-ensemble de l'ensemble des réels positifs à n dimensions, produit cartésien des ensembles D_j, domaines des variables x_j ;

$f : D^n \rightarrow R$ est une application de l'ensemble D^n dans les réels ;

$G : D^n \rightarrow R^m$ est une application de l'ensemble D^n dans les réels à m dimensions.

2.2. **Passage à la forme canonique.**

Un programme qui ne se présente pas sous la forme canonique peut aisément être ramené à cette forme : s'il s'agit d'un problème de maximisation, on note que :

$$\max f(X) = -\min -f(X) \tag{7}$$

si les contraintes ne sont pas sous la forme requise, on utilise les relations :

$$g_i\ (x_1\ ,\ x_2\ ,\ ...,\ x_m) \leqslant 0 \ <=> \ -g_i\ (x_1\ ,\ x_2\ ,\ ...,\ x_m) \geqslant 0 \qquad (8)$$

$$g_i\ (x_1\ ,\ x_2\ ,\ ...,\ x_m) = 0 \ <=>$$

$$g_1\ (x_1\ ,\ x_2\ ,\ ...,\ x_m) \geqslant 0 \quad \text{et} \quad -g_i\ (x_1\ ,\ x_2\ ,\ ...,\ x_m) \geqslant 0 \qquad (9)$$

et

$$g_i\ (x_1\ ,\ x_2\ ,\ ...,\ x_m) > 0 \ <=> \ g_i\ (x_1\ ,\ x_2\ ,\ ...,\ x_m) - \varepsilon \geqslant 0 \qquad (10)$$

où $<=>$ désigne l'équivalence logique et ε une quantité très petite.

La relation (10) n'est pas rigoureuse du point de vue mathématique, mais acceptable en pratique et appliquée automatiquement sur ordinateur.

2.3. **Définitions.**

Appelons les contraintes de la forme (2), qui portent simultanément sur plusieurs variables, *contraintes globales*, et les contraintes de la forme (3), qui portent chacune sur une seule variable *contraintes individuelles*; appelons *solution candidate* tout vecteur X vérifiant les contraintes individuelles (3) et *solution admissible* tout vecteur X vérifiant les contraintes individuelles (3) et les contraintes globales (2); enfin, appelons *solution optimale* tout vecteur X vérifiant les contraintes individuelles (3) ainsi que les contraintes globales (2) et réalisant, sous ces conditions, le minimum de la fonction économique (1).

La distinction entre les contraintes individuelles et globales et la notion de solution candidate ainsi définie s'avère très utile dans l'exposé des P.S. La terminologie utilisée est en accord avec celle en usage en programmation linéaire. Lorsque, par abus de langage, le terme solution est utilisé sans qualificatif dans la suite, il signifie toujours solution candidate.

2.4. **Exemples.**

a) Programme linéaire :

Minimiser

$$3\,x_1 + 2\,x_2 + 4\,x_3 + 5\,x_4 \qquad (11)$$

sous les contraintes

$$x_1 + x_2 + x_3 + x_4 - 3 \geqslant 0 \qquad (12)$$

$$-x_1 + 2\,x_2 - x_3 + x_4 \geqslant 0$$

$$x_1\ ,\ x_2\ ,\ x_3\ ,\ x_4 \geqslant 0 \qquad (13)$$

La fonction économique est la fonction linéaire (11); les contraintes globales sont les inégalités linéaires (12); les contraintes individuelles sont les contraintes de non-négativité des variables; le domaine de chacune des variables est l'ensemble R^+ des réels positifs.

b) Programme non linéaire :

Minimiser

$$2x_1 + 2x_2x_3 + x_1x_2x_3 \tag{14}$$

sous les contraintes

$$x_1 + 2x_2 - 1 \geqslant 0 \tag{15}$$

$$2x_2 + x_3 - 7 \geqslant 0$$

$$x_1, x_2, x_3 \geqslant 0 \tag{16}$$

La fonction économique est la fonction non linéaire (14).

c) Programme linéaire mixte :

Ajoutons au programme linéaire (11), (12), (13) la contrainte :

$$x_3 \equiv 0 \bmod (1) \tag{17}$$

Les contraintes individuelles sont les contraintes de non-négativité des variables et la contrainte d'intégralité de la variable x_3 (x_3 congruent à 0 modulo 1); le domaine de la variable x_3 est l'ensemble Z^+ des entiers positifs.

d) Programme linéaire en variables 0-1 :

Ajoutons au programme linéaire (11), (12), (13) la contrainte :

$$x_1, x_2, x_3, x_4 \in \{0, 1\} \tag{18}$$

Les contraintes individuelles sont les contraintes de non-négativité des variables et les contraintes exigeant que les variables prennent la valeur 0 ou 1; les ensembles D_1 à D_4 sont égaux à l'ensemble $B_2 = \{0, 1\}$ des nombres booléens; l'ensemble D^N est égal à l'ensemble B_2^4 des vecteurs booléens à quatre composantes; les contraintes de non-négativité des variables sont redondantes et peuvent être omises.

Les programmes des exemples a) et b) sont structurés; les programmes des exemples c) et d) ne sont pas structurés et peuvent être résolus à l'aide d'une P.S.

3. Sous-ensemble sondable de solutions.

3.1. Caractéristiques des P.S.

La résolution d'un programme mathématique à l'aide d'une P.S. consiste en des séparations successives de l'ensemble des solutions candidates et en l'application de tests aux sous-problèmes ainsi obtenus. Pour caractériser les P.S., il faut donc préciser :

- quand un sous-ensemble doit être séparé;
- comment les séparations doivent être faites;
- quelles sont les formes des tests utilisés.

Le concept de *sous-ensemble sondable* permet de répondre au premier point et celui de *principe de séparation* au deuxième.

3.2. Sous-ensemble sondable.

Soit un programme mathématique à résoudre par une P.S.; considérons un sous-ensemble de solutions candidates — brièvement un sous-ensemble — obtenu par des séparations successives de l'ensemble des solutions candidates; l'application des tests de la P.S. fournit une certaine information à propos de ce sous-ensemble; par définition un sous-ensemble est dit *sondable* * si les tests de la P.S. fournissent toute l'information requise concernant ce sous-ensemble pour la résolution du programme. Un sous-ensemble est sondable dans les trois cas suivants :

a) les tests permettent d'obtenir la solution optimale du sous-problème considéré;

b) les tests permettent de montrer que le sous-ensemble ne contient pas de solution admissible;

c) les tests permettent de montrer que le sous-ensemble ne contient pas de solution admissible qui donne à la fonction économique une valeur inférieure à celle donnée par la meilleure solution déjà connue.

La résolution d'un programme mathématique structuré à l'aide d'un algorithme fournit la solution optimale ou la preuve que le problème n'admet pas de solution admissible; ces cas correspondent aux cas a) et b) ci-dessus; le cas c) provient de ce que la P.S. comporte l'examen successif de plusieurs sous-problèmes et la solution optimale d'un sous-problème n'est pas nécessairement celle du programme originel.

---

* Ce terme a été introduit par A. Geoffrion [9] à propos des P.S. pour les programmes linéaires en variables 0-1.

Le terme sondable rappelle que les sous-problèmes considérés ne sont pas résolus par un algorithme, mais simplement soumis à des tests. Tous les sous-ensembles de solutions candidates ne sont évidemment pas sondables; un sous-ensemble non sondable devra être séparé jusqu'à l'obtention de sous-ensembles sondables. Le fait qu'un sous-ensemble soit sondable dépend d'une part des solutions candidates qui le constituent, mais également d'autre part des tests qui sont utilisés; c'est ainsi, par exemple, qu'un sous-ensemble peut ne pas contenir de solution admissible sans que les tests utilisés permettent de le montrer; il est possible, dans ce cas, que d'autres tests puissent le faire.

Pour voir si des sous-ensembles sont sondables dans le cas c), il faut obtenir, dès que possible, une solution admissible dont la valeur soit une bonne estimation de celle de la solution optimale; une telle solution peut être obtenue en prenant la solution utilisée en pratique si l'on étudie un problème réel; sinon, un algorithme heuristique peut être utilisé dans ce but, préalablement à la résolution par une P.S.

4. Principe de séparation.

4.1. Axiomes.

La manière dont se font les séparations lors de l'application d'une P.S. est précisée en un principe de séparation. Une P.S. pour une classe de problèmes est dite *convergente* si, pour tout problème de cette classe, elle fournit en un temps de calcul fini la solution optimale ou la preuve que le problème n'admet pas de solution admissible. Certaines conditions doivent être vérifiées par un principe de séparation pour que la ou les P.S. qui l'utilisent soient convergentes : diverses formulations de ces conditions sont possibles : les quatre axiomes suivants constituent un ensemble de conditions suffisant pour la convergence et vérifié par un grand nombre de P.S.

Axiome A 1 : Le premier ensemble à séparer est l'ensemble des solutions candidates.

Si les tests de la P.S. montrent que l'ensemble des solutions candidates est sondable, le problème est résolu sans séparation; sinon cet ensemble doit être séparé.

Axiome A 2 : La réunion des sous-ensembles obtenus lors d'une séparation doit être égale à l'ensemble séparé.

Cet axiome traduit le fait qu'on ne peut, lors d'une séparation, perdre de solution; toute solution devra en définitive être comprise dans un ensemble sondable.

Dans la plupart des P.S., les sous-ensembles obtenus lors d'une séparation sont disjoints et constituent donc une partition de l'ensemble séparé, condition plus forte que la condition de recouvrement requise par l'axiome A 2.

Axiome A 3 : Lorsqu'un sous-ensemble de solutions ne peut être séparé, il doit être sondable.

Cet axiome est souvent vérifié de manière triviale car le sous-ensemble ne pouvant être séparé suivant le principe de séparation ne contient qu'une seule solution candidate; ce n'est pas toujours le cas : dans les P.S. pour les programmes mixtes, par exemple, lorsque le principe de séparation ne peut plus être appliqué, on a un programme linéaire.

Axiome A 4 : Le nombre de séparations doit être fini.

Cet axiome peut être abandonné, comme l'a montré Mitten; la démonstration de la convergence est alors plus difficile.

4.2. **Formalisation des axiomes.**

Les axiomes peuvent être formalisés, en utilisant quelques concepts usuels de théorie des graphes.*

Soit $H = (Z, \Gamma)$ une arborescence de racine z_0, associée à une P.S. de la manière suivante : tout sommet $z \in Z$ de H est associé à un sous-ensemble de solutions qui peut être obtenu par des séparations successives, suivant le principe de séparation, de l'ensemble des solutions candidates; aux suivants d'un sommet sont associés les sous-ensembles obtenus par séparation de l'ensemble associé à ce sommet. Le sous-ensemble de solution associé au sommet z_n sera désigné par P_n. Par abus de langage, un sommet sera dit sondable si le sous-ensemble auquel il est associé est sondable.

Les axiomes s'écrivent alors :

$$\text{A 1 :} \qquad P_0 = D^n \tag{19}$$

L'ensemble des solutions candidates est associé à la racine de l'arborescence.

$$\text{A 2 :} \qquad \bigcup_{z_y \in \Gamma z_x} P_y = P_x \tag{20}$$

* La terminologie utilisée est celle de C. Berge [7] auquel on renvoie : rappelons qu'une arborescence est un graphe connexe avec un seul sommet sans précédent (la racine) et dont tous les autres sommets ont un seul précédent ; un sommet pendant d'une arborescence est un sommet sans suivant ; la notion intuitive d'arborescence se comprend aisément d'après la figure 1.

La réunion des sous-ensembles associés aux suivants d'un sommet est égale au sous-ensemble associé à ce sommet.

A 3 : $\Gamma\, z_x = \phi \Rightarrow P_x$ est sondable. (21)

Un sommet dont le sous-ensemble associé ne peut être séparé est dit terminal; un tel sommet n'a pas de suivant.

A 4 : $|Z| < \infty$ (22)

L'arborescence H est finie.

4.3. **Formalisation du concept de principe de séparation.**

Les résultats précédents permettent de formaliser le concept de principe de séparation : un principe de séparation S sur D^n, ensemble des solutions candidates d'un programme mathématique, est un triplet :

$$S = (Z, \Gamma, P) \tag{23}$$

tel que

- Z est un ensemble fini
- Γ est une application de Z dans P (Z), ensemble des parties de Z

$$\Gamma : Z \to P(Z) : z \to \Gamma(z) \tag{24}$$

dont le graphe est une arborescence

- P est une application de Z dans $P(D^n)$

$$P : Z \to P(D^n) : z_x \to P_x \tag{25}$$

vérifiant les axiomes A 1, A 2, A 3 et A 4.

4.4. **Exemple.**

Le programme linéaire en variables 0-1 (11), (12), (13), (18) du paragraphe 2.4 admet pour ensemble des solutions candidates l'ensemble des vecteurs booléens à quatre composantes :

				(26)
0000	0100	1000	1100	
0001	0101	1001	1101	
0010	0110	1010	1110	
0011	0111	1011	1111 .	

Un principe de séparation valable pour des problèmes de cette classe consiste à choisir une variable — par exemple la première variable non encore choisie, ou libre, dans l'ordre des indices croissants — et à séparer le sous-ensemble considéré d'après les deux valeurs que peut prendre cette variable.

En appliquant ce principe, on choisit la variable x_1 et on sépare l'ensemble $P_0 = B_2^4$ en deux ensembles P_1 et P_2 comprenant respectivement les huit vecteurs de gauche et les huit vecteurs de droite de (26).

Il est aisé de voir que les axiomes A 1 et A 2 sont vérifiés; les sous-ensembles obtenus constituent d'ailleurs une partition de l'ensemble séparé. Comme un sous-ensemble ne peut plus être séparé seulement s'il n'a plus de variables libres et qu'il ne contient alors qu'un vecteur unique et est sondable, l'axiome A 3 est vérifié; comme le cardinal de B_2^4 — et, plus généralement de B_2^n pour n fini — est fini, le cardinal de $P(B_2^4)$ — et de $P(B_2^n)$ — l'est également et l'axiome A 4 est vérifié.

4.5. **Familles de principes de séparation.**

Il est fréquent qu'un principe de séparation consiste en le choix d'une variable du programme considéré et en l'imposition de conditions sur les valeurs que peut prendre cette variable; les conditions consisteront par exemple, comme au paragraphe précédent, à fixer la variable choisie à certaines valeurs, ou bien à imposer des bornes inférieures et supérieures sur les valeurs qu'elle peut prendre.

Deux principes de séparation seront dits de la même *famille* s'ils diffèrent seulement par la règle de choix de la variable utilisée pour les séparations; la convergence d'une P.S. ne dépend généralement pas de la règle de choix et peut être démontrée simultanément pour tous les principes de séparation d'une famille; par contre, les performances d'une P.S. sur ordinateur dépendent fortement de cette règle.

5. Théorème de convergence.

5.1. **Etapes des P.S.**

Les concepts de sous-ensemble sondable et de principe de séparation permettent de préciser les étapes des P.S. et leur enchaînement; à mesure

que progresse la résolution, une arborescence H1 est construite, qui s'accroît de nouveaux sommets à chaque séparation; cette arborescence est un sous-graphe de l'arborescence H associée au principe de séparation et se confond avec H seulement dans le cas où aucun sous-ensemble de solutions associé à un sommet non terminal n'est sondable.

Trois étapes se succèdent dans une P.S. :

1. Choisir parmi les sommets pendants de H1 un sommet non sondable, s'il en existe; sinon la procédure est terminée.

2. Séparer suivant le principe de séparation le sous-ensemble de solutions associé au sommet choisi.

3. Appliquer aux sous-ensembles obtenus par la séparation les tests de la P.S.; noter les sous-ensembles sondables; si un sous-ensemble est sondable dans le cas a) et la solution optimale du sous-problème considéré

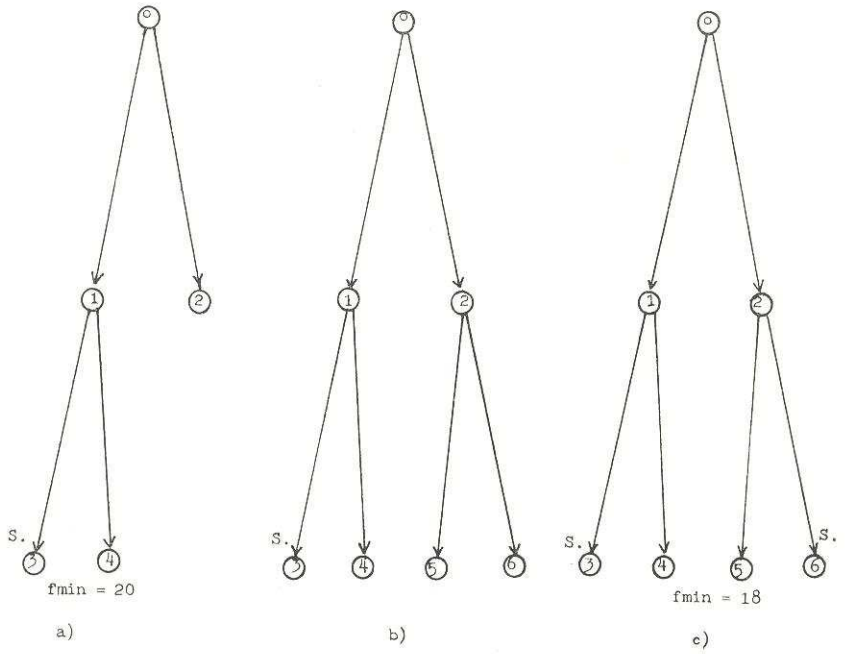

Fig. 1.

est la meilleure solution obtenue jusqu'à présent, la conserver.
Aller en 1.

La figure 1 illustre schématiquement ces étapes : les sommets 2, 3 et 4 sont les sommets pendants de l'arborescence en 1, a) : on suppose que le sommet 3 est sondable, ce qui est noté par S et qu'une solution admissible de valeur 20 est connue : fmin désigne cette valeur, la meilleure déjà obtenue. A l'étape 1, le choix se fait entre les sommets 2 et 4. On suppose que le sommet 2 est choisi. A l'étape 2, la séparation fournit deux sous-ensembles et l'arborescence s'accroît des sommets 5 et 6 (fig. 1, b). A l'étape 3, les tests sont appliqués aux sous-ensembles associés aux sommets 5 et 6; on suppose que le sous-ensemble associé au sommet 5 n'est pas sondable et celui associé au sommet 6 est sondable dans le cas a) et fournit une solution de valeur 18. Cette solution est conservée; on retourne alors à l'étape 1, les sommets pendants non sondables étant les sommets 4 et 5 (fig. 1, c).

5.2. **Démonstration.**

En raisonnant sur l'arborescence H1 obtenue lors de la résolution, on démontre le

Théorème 1 : Toute procédure d'optimisation par séparation dont le principe de séparation vérifie les axiomes A1 à A4 est convergente.

Lorsque la résolution par une PS est terminée, tous les sommets pendants de H1 sont sondables. Il faut montrer qu'alors la solution optimale du problème est connue ou que le problème n'admet pas de solution admissible. Un ensemble de solutions associé à un sommet quelconque de l'arborescence sera dit *sondable après séparation* si toutes ses solutions sont comprises dans des ensembles sondables obtenus lors de la résolution; démontrer la convergence de la P.S. et équivalent à montrer que l'ensemble $P_0 = D^n$ des solutions candidates est sondable après séparation; l'arborescence H1 a au moins un sommet z_x dont les suivants z_v, z_y, ... sont les seuls descendants : en vertu de l'axiome A2, P_x est sondable après séparation. En supprimant les suivants de z_x, on obtient une nouvelle arborescence dont tous les sommets pendants sont sondables ou sondables après séparation. Par récurrence, l'arborescence est réduite à sa racine et l'ensemble P_0 est donc sondable après séparation. La figure 2 illustre cette démonstration.

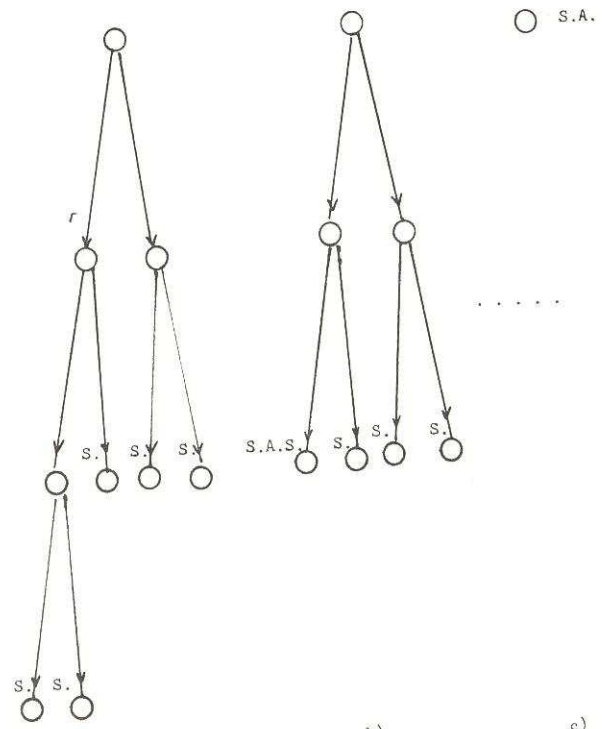

Fig. 2.

5.3. **Choix du sommet pour la séparation.**

La manière de choisir, parmi les sommets pendants non sondables de l'arborescence, le sommet pour la séparation n'a pas encore été précisée. Plusieurs règles de choix sont possibles : comme dans le cas de la règle de choix de la variable pour la séparation, ces règles n'influencent pas la convergence des P.S. mais fortement les performances sur ordinateur. Comme l'efficacité de certains des tests utilisés dans les P.S. dépend aussi de la règle choisie, la discussion des options possibles est reportée au paragraphe 8, après l'examen des différentes formes de tests.

6. Fonctions d'évaluation.

6.1. **Définition.**

Les tests des P.S. utilisent des minorants et des majorants des valeurs prises par la fonction économique ou par le membre de gauche des contraintes

globales sur les solutions admissibles d'un sous-ensemble de solutions donné; ces minorants et majorants sont calculés à l'aide de *fonctions d'évaluation*; celles-ci sont donc des applications dans les réels de l'ensemble des sous-ensembles de solutions qui peuvent être obtenus à l'aide du principe de séparation, ou, ce qui est équivalent, des applications de l'ensemble Z des sommets de l'arborescence H dans les réels.

Une fonction d'évaluation $\underline{f}$ associée à la fonction économique

$$\underline{f} : Z \to R : z_x \to \underline{f}(z_x) \tag{27}$$

vérifie la condition

$$\forall\ z_x \in Z \quad \text{et} \quad \forall\ X \in P'_x : \underline{f}(z_x) \leqslant f(X) \tag{28}$$

où P_x' est l'ensemble des solutions admissibles de P_x,

et une fonction d'évaluation $\overline{g}_i$ associée à la contrainte i

$$\overline{g}_i : Z \to R : z_v \to \overline{g}_i(z_x) \tag{29}$$

vérifie la condition

$$\forall\ z_x \in Z \quad \text{et} \quad \forall\ X \in P'_x : \overline{g}_i(z_x) \leqslant \overline{g}_i(X)\,. \tag{30}$$

6.2. **Obtention des fonctions d'évaluation.**

Les fonctions d'évaluation se présentent comme des problèmes associés au problème à résoudre et plus simples que celui-ci; deux procédés permettent d'obtenir ces fonctions :

a) abandon ou affaiblissement d'une partie des contraintes du problème;

b) construction d'une nouvelle fonction bornant la fonction économique ou une des contraintes globales.

Le procédé a) est le plus usité; les axiomatiques de Balas et Mitten n'en prévoient pas d'autres. Les manières les plus courantes d'utiliser ce procédé consistent à :

- abandonner les contraintes d'intégralité des variables libres et, éventuellement, une partie des contraintes globales;
- conserver une seule contrainte globale et les contraintes individuelles.

Le procédé b) est utilisé surtout en présence de fonctions non linéaires (comme, par exemple, dans les programmes séparables non linéaires et non convexes) et permet d'associer à ces fonctions de nouvelles fonctions linéaires par morceaux; ce procédé permet aussi d'obtenir une fonction d'évaluation

sous forme de programme linéaire pour les programmes mixtes du type « programme à coûts fixes » (voir exemple ci-dessous).

6.3. **Exemples.**

a) Programme linéaire en variables 0-1.

Considérons à nouveau le programme linéaire en variables 0-1 (11), (12), (13), (18) et supposons être en un somment z_n de l'arborescence H1 tel que $x_1 = x_2 = 1$, x_3 et x_4 étant libres; en remplaçant x_1 et x_2 par leur valeur, le programme s'écrit :

Minimiser
$$5 + 4x_3 + 5x_4 \tag{31}$$

sous les contraintes
$$x_3 + x_4 - 1 \geqslant 0 \tag{32}$$
$$-x_3 + x_4 \geqslant 0$$
$$x_3, x_4 \geqslant 0 \tag{33}$$
$$x_3, x_4 \in \{0, 1\}. \tag{34}$$

En abandonnant la contrainte (34) d'intégralité des variables libres x_3 et x_4, on obtient un programme linéaire dont la solution optimale est $x_3 = x_4 = 1/2$, de valeur 9,5; on a ainsi calculé un minorant $f(z) = 9{,}5$ de la valeur de f sur le sous-ensemble P_n considéré.

Si, dans le même exemple, on conserve seulement la première contrainte globale, on obtient un problème plus simple (du type « problème du sac de campeur »); la solution optimale de ce problème est $x_3 = 1$, $x_4 = 0$, de valeur 9; la résolution du problème donnant le minorant $\underline{f}(z_n)$ est plus aisée, mais la valeur de ce minorant est plus petite.

Un majorant de la valeur prise par le membre de gauche de la première contrainte est obtenu en fixant à 1 les variables libres correspondant aux coefficients positifs de cette contrainte et à 0 les autres variables libres; on trouve $\overline{g}_1(z_n) = 1$.

b) Programme à coûts fixes.

Dans ce type de programme, qui correspond à de nombreuses situations économiques, un coût fixe est encouru dès qu'une variable x_j prend une valeur non nulle; la figure 3 représente les coûts correspondant à la variable x_j; ces coûts s'expriment par les relations :

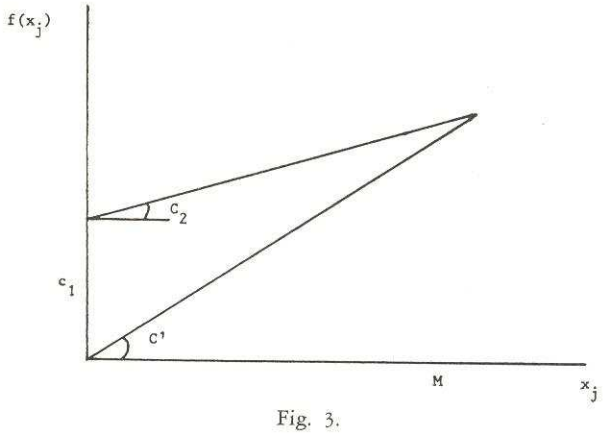

Fig. 3.

$$f(x_j) = C_1\, x'_j + C_2\, x_j \tag{35}$$

$$-x_j + M\, x'_j \geqslant 0 \tag{36}$$

$$x_j \geqslant 0 \tag{37}$$

$$x'_j \in \{0, 1\} \tag{38}$$

où C_1 est le coût fixe, C_2 le coût proportionnel, M une borne supérieure de la valeur de x_j et x'_j une variable booléenne auxiliaire. Un minorant de $f(x_j)$ peut être obtenu en utilisant la fonction :

$$f'(x_j) = C'\, x_j = \left(\frac{C_1}{M} + C_2\right) x_j\,. \tag{39}$$

L'évaluation ainsi obtenue sera exacte si x_j prend la valeur 0 ou la valeur M.

6.4. **Choix des fonctions d'évaluation.**

Les exemples ci-dessus montrent que diverses fonctions d'évaluation peuvent être utilisées dans une P.S. pour une même classe de problèmes; l'efficacité des tests dépendra fortement des fonctions choisies; il faut prendre en compte, pour le choix des fonctions d'évaluation :

- la facilité de calcul sur ordinateur, en temps de calcul et en nombre de mémoires utilisées;

. la précision, c'est-à-dire la différence entre le minorant (respectivement le majorant) calculé et le minimum (respectivement le maximum) de la fonction évaluée sur l'ensemble des solutions admisibles du sous-ensemble de solutions considéré.

7. Tests.

7.1. Classification des tests.

Les tests permettent de voir si un sous-ensemble est sondable dans le cas a) b) ou c) du paragraphe 3.2 et peuvent être classés suivant ces trois cas :

. un *test de résolution* permet de voir si une solution particulière est la solution optimale du sous-problème considéré;
. un *test d'admissibilité* permet de voir si le sous-ensemble de solutions considéré ne peut pas contenir de solution admissible;
. un test *d'optimalité* permet de voir si le sous-ensemble de solutions considéré ne peut pas contenir de solution admissible donnant à la fonction économique une valeur inférieure à la meilleure valeur déjà obtenue.

D'autre part, les tests peuvent être classés suivant le fait qu'ils s'appliquent à la totalité du sous-ensemble de solutions associé au sommet considéré, ou à une partie seulement de ce sous-ensemble :

. un test *direct* permet de voir si le sous-ensemble de solutions considéré est sondable ou non, dans le cas a), b) ou c);
. un *test conditionnel* permet de voir si une partie du sous-ensemble de solutions associé au sommet considéré, caractérisée par une *condition*, est sondable dans le cas a), b) ou c).

Enfin, on peut utiliser également dans les P.S. des tests auxiliaires qui n'entrent pas dans ces deux classifications : un test auxiliaire est utilisé en conjonction avec un test direct ou un test conditionnel et permet de voir s'il est certain ou probable que ce test ne fournisse aucune information utile à la résolution; le but d'un test auxiliaire est de permettre un gain de temps de calcul en évitant d'appliquer inutilement un test direct ou conditionnel; pour qu'un test auxiliaire soit utile, son temps de calcul doit être nettement inférieur à celui du test auquel il est associé.

7.2. Tests de résolution.

Un test direct de résolution consiste à voir si la solution optimale d'un problème associé au sous-problème considéré et obtenu en négligeant cer-

taines contraintes vérifie toutes les contraintes du sous-problème; il est fréquent que le sous-problème associé soit un sous-problème utilisé pour calculer un minorant ou un majorant pour un autre test.

Considérons le programme linéaire en variables 0-1 (31), (32), (33), (34); au paragraphe 6.3, on a montré comment le programme linéaire associé (31), (32), (33) permet de calculer un minorant de la valeur prise par la fonction économique : si toutes les variables prenaient la valeur 0 ou 1 dans la solution optimale de ce programme linéaire, le programme (31), (32), (33), (34) serait résolu; on voit que ce n'est pas le cas.

Considérons le problème (11), (12), (13), (18) et le sous-ensemble de solutions caractérisé par $x_1 = x_2 = x_4 = 1$; il reste une variable libre : x_3; la solution optimale du problème associé obtenu en négligeant les contraintes globales est $x_3 = 0$, les coefficients de la fonction économique étant positifs; la solution $x_1 = x_2 = x_4 = 1$ et $x_3 = 0$ vérifie les contraintes globales : le test de résolution montre donc que c'est la solution optimale du sous-problème considéré.

Un test conditionnel de résolution consisterait, par exemple, à fixer une variable à une valeur donnée, puis à appliquer un test de résolution.

7.3. **Test d'admissibilité.**

Un test d'admissibilité consiste à calculer, pour chaque contrainte globale i, un majorant de la valeur prise par le membre de gauche de cette contrainte sur l'ensemble des solutions admissibles du sous-ensemble de solutions considéré; si un de ces majorants est négatif, le sous-ensemble est sondable.

Au paragraphe 6.3, on a montré comment un majorant pouvait être calculé pour une contrainte d'un programme linéaire en variables 0-1 en additionnant les coefficients des variables fixées à 1, les coefficients positifs des variables libres et la constante de cette contrainte.

Un test conditionnel d'admissibilité consiste, par exemple, à fixer une variable à une valeur donnée puis à appliquer un test d'admissibilité.

Considérons encore le programme linéaire en variables 0-1 (11), (12), (13), (18) et supposons être en un sommet z_n de l'arborescence H1 tel que $x_3 = 1$, $x_4 = 0$, les variables x_1 et x_2 étant libres; le programme s'écrit alors :

Minimiser

$$3\,x_1 + 2\,x_2 + 4 \qquad (40)$$

sous les contraintes

$$x_1 + x_2 - 2 \geqslant 0 \tag{41}$$

$$-2x_1 + 2x_2 - 1 \geqslant 0$$

$$x_1, x_2 > 0 \tag{42}$$

$$x_1, x_2 \in \{0, 1\}. \tag{43}$$

Le test direct d'admissibilité donne pour la seconde contrainte un majorant de 1; si la condition $x_1 = 1$ est imposée, le test conditionnel d'admissibilité donne pour cette contrainte un majorant de -1; le sous-ensemble caractérisé par $x_3 = 1$, $x_4 = 0$ et $x_1 = 1$ est donc sondable; le sous-ensemble considéré ne peut contenir que des solutions admissibles pour lesquelles $x_1 = 0$; de fait, ce sous-ensemble ne contient pas de solution admissible sous cette condition non plus; ceci illustre le fait que si un sous-ensemble n'est pas sondable dans le cas b), cela signifie que les tests utilisés ne peuvent montrer qu'il n'y a pas de solution admissible et non qu'il en existe nécessairement une.

7.4. **Tests d'optimalité.**

Un test direct d'optimalité consiste à calculer un minorant de la valeur prise par la fonction économique sur l'ensemble des solutions admissibles du sous-ensemble considéré; si ce minorant est supérieur à la valeur de la meilleure solution déjà obtenue, le sous-ensemble est sondable.

Deux procédés de calcul d'un minorant ont été illustrés, dans le cas d'un programme linéaire en variable 0-1, au paragraphe 6.3; les minorants obtenus étant de 9,5 et 9, le sous-ensemble considéré serait sondable seulement si une solution de valeur inférieure à 9,5 était connue.

Les tests conditionnels d'optimalité sont de deux formes :

a) Fixer une variable à une valeur donnée, puis appliquer un test d'optimalité.

b) Voir s'il existe des variables telles que toute solution où une variable prend une certaine valeur est meilleure que toute solution où cette variable prend une autre valeur, les autres variables restant égales.

Pour tenir compte rigoureusement de la seconde forme, la définition d'ensemble sondable devrait être légèrement modifiée : en effet, on tient compte d'une meilleure solution qui n'est pas encore connue, mais dont l'existence doit être certaine, pour éliminer des solutions; la seconde forme

des tests d'optimalité conditionnelle s'applique plus rarement que la première; elle est très utile pour des classes particulières de problèmes, comme le « problème de la localisation des entrepôts ».

7.5. **Tests auxiliaires.**

Les tests auxiliaires peuvent être utilisés de diverses manières; on peut distinguer trois formes de ces tests :

. *tests auxiliaires déterministes* : ces tests permettent de voir si les tests directs ou conditionnels auxquels ils sont associés ne peuvent fournir d'information utile; par exemple, un test auxiliaire permet de voir si toutes les solutions du sous-ensemble considéré sont admissibles; si c'est le cas, il est inutile d'employer un test direct ou conditionnel d'admissibilité.

. *test auxiliaires probabilistes* : ces tests permettent de voir si les test auxquels ils sont associés n'ont qu'une faible chance de fournir une information utile. Certains tests directs ou conditionnels sont efficaces seulement si les valeurs de la plupart des variables sont fixées; un test auxiliaire probabiliste très simple consiste à compter le nombre de variables libres et à utiliser les tests directs ou conditionnels seulement si ce nombre est inférieur à une valeur donnée.

. *test auxiliaires adaptatifs* : ces tests tiennent compte de l'information qui a déjà été fournie par les tests auxquels ils sont associés pour voir s'il y a lieu de les utiliser ou non; un test auxiliaire adaptatif très simple consiste à noter le nombre de variables libres la première fois que le test direct ou conditionnel fournit une information utile et à utiliser ensuite ce test seulement lorsque le nombre de variables libres est voisin de ce nombre.

8. Procédures S.E.P. et S.E.S.

8.1. **Procédures d'optimisation par séparation et évaluation progressive.**

On distingue plusieurs classes de P.S. d'après la manière dont on choisit le sommet pendant non sondable pour une séparation; la classe la plus étudiée de ces procédures, les procédures par séparation et évaluation progressive (P.S.E.P.), utilise la règle suivante, un minorant de la valeur de la fonction économique ayant été calculé en chaque sommet pendant :

- on sélectionne le sommet pendant non sondable dont le minorant est minimum

Il est fréquent que la valeur de la fonction d'évaluation associée à la fonction économique coïncide avec la valeur de la solution optimale en tout sommet terminal; cette condition, qui n'est pas toujours vérifiée, est requise par les axiomatiques, à l'exception de celle de Hervé. Si la condition est vérifiée, la première solution admissible à être obtenue est la solution optimale du problème; cette solution peut être obtenue en un sommet non terminal sondable par résolution (cas a) ou, plus fréquemment, en un sommet terminal dont le sous-ensemble associé contient une solution admissible au moins.

La règle de choix des P.S.E.P. permet de ne pas explorer de sommets de l'arborescence dont le minorant de la fonction économique a une valeur supérieure à la valeur de la solution optimale; c'est pourquoi cette règle a été souvent recommandée; elle présente cependant des défauts des points de vue de l'organisation des calculs sur ordinateur et de l'emploi des tests d'optimalité.

L'emploi d'une P.S.E.P. sur ordinateur implique la conservation en mémoire de la liste des sommets pendants de l'arborescence H1, de l'état des variables fixées à une valeur entière ou bornées et des minorants calculés en chacun des sommets; si le programme à résoudre est de taille moyenne, la capacité de la mémoire interne peut être dépassée et l'usage de mémoires externes ralentit fortement les calculs; de plus, pour choisir le plus petit minorant, il faut utiliser une sous-routine de lecture de liste ou une sous-routine de classement et toutes deux prennent du temps dès que la liste s'allonge.

8.2. **Procédures d'optimisation par séparation et évaluation séquentielle.**

Une autre classe de P.S., les procédures d'optimisation par séparation et évaluation séquentielle, ne présente pas les mêmes défauts que les P.S.E.P.; la règle de choix du sommet pour la séparation est la suivante :

- on sélectionne le sommet pendant non sondable le plus à droite dans l'arborescence H1.

Suivant cette règle, on emprunte l'arc le plus à droite en tout sommet de l'arborescence M, en partant de la racine, jusqu'à atteindre un sommet sondable, puis on revient sur ses pas jusqu'à un sommet ayant au moins un suivant non sondable et non séparé, on explore ce sommet suivant et

ainsi de suite; lorsque la racine de l'arborescence n'a plus de suivant non sondable et non séparé, la procédure est terminée.

L'emploi d'une P.S.E.P. sur ordinateur nécessite seulement la conservation d'un vecteur caractérisant l'état des variables en le sommet étudié et d'un vecteur spécifiant le chemin pour y arriver. La règle rigide de progression et de régression dans l'arborescence H permet de choisir le sommet à séparer avec un minimum de calculs.

Il arrive qu'une P.S.E.S. explore inutilement une partie de l'arborescence où ne se trouvent que de mauvaises solutions, la règle de choix permettant l'examen de sommets dont le minorant est supérieur à la valeur de la solution optimale; cependant, pour de nombreuses classes de problèmes, on peut construire une P.S.E.S. donnant rapidement une solution optimale ou proche de l'optimum.

8.3. **Procédures mixtes.**

Des procédures ont été proposées [6] pour pallier certains défauts des P.S.E.P. et des P.S.E.S. : on peut, par exemple, débuter par l'emploi d'une P.S.E.P. jusqu'à ce que l'arborescence H1 atteigne un nombre donné de sommets, puis poursuivre par une P.S.E.S. l'exploration des branches de l'arborescence issues des sommets choisis. De cette façon, on n'utilise pas trop de place en mémoire et on espère ne pas explorer inutilement des zones de l'arborescence éloignées de la solution optimale.

8.4. **Tests d'optimalité dans les P.S.E.P. et les P.S.E.S.**

Les tests d'optimalité sont d'ordinaire les tests les plus efficaces des P.S.; ils utilisent la valeur de la meilleure solution admissible déjà obtenue au cours de la résolution. Les P.S.E.P. utilisent peu ces tests : les tests d'optimalité servent seulement à prouver, de manière triviale, que la première solution admissible obtenue est optimale; les tests d'optimalité conditionnelle ne sont pas utilisés, sauf si une bonne solution est déjà connue par ailleurs. Dans les P.S.E.S., les tests d'optimalité peuvent être utilisés aussitôt qu'une solution admissible est obtenue; si cette solution est proche de la solution optimale, les tests d'optimalité conditionnelle se montrent très efficaces.

*
* *

BIBLIOGRAPHIE

[1] AGIN, N., « Optimum Seeking with Branch and Bound », *Management Science 13,* n° 4, pp. 176-185 (1966).

[2] BALAS, E., « A note on the Branch and Bound Principle », *Operations Research, 16,* pp. 442-445 (Errata p. 886), (1968).

[3] BALAS, E., « The Intersection cut — A New Cutting Plane for Integer Programming », *Management Sciences Research Report,* n° 187, Carnegie Mellon University.

[4] BALINSKI, M.L., « Integer Programming : Methods, Uses, Computation », *Management Science,* 12, 3, pp. 253-313 (1965).

[5] BALINSKI, M.L., « Some General Methods in Integer Programming » in J. Abadie (ed.), « *Nonlinear Programming* », North Holland Publishing Company, Amsterdam (1967).

[6] BALINSKI, M.L. et SPIELBERG, K., « Methods for Integer Programming », pp. 195-292 in J.S. ARONOFSKY (ed.), « *Progress in Operations Research,* Volume III », John Wiley, New York, London, Sydney, Toronto (1969).

[7] BERGE, C., « *Théorie des Graphes et ses applications* », Dunod, Paris, 1958, 2ème édition (1963).

[8] BERTIER, P. et ROY, B., « Procédure de résolution pour une classe de problèmes pouvant avoir un caractère combinatoire », *Cahiers du Centre d'Etudes de Recherche Opérationnelle* », 6, pp. 202-208 (1964).

[9] GEOFFRION, A., « Integer Programming by implicit Enumeration and Balas' Method », *SIAM Review,* 9, pp. 178-190 (1967).

[10] GOMORY, R., « An Algorithm for Integer Solutions to Linear Programs », Princeton IBM Math. Res. Report Nov. 1958, also in R.L. GRAVES and P. WOLFE (ed.), « *Recent Advances in Mathematical Programming* », McGraw-Hill, New York, pp. 269-302 (1963).

[11] GOMORY, R., « All-Integer Programming Algorithm », IBM Research Center Report RC - 189 Jan. 1960 ; also in J.F. MUTH and G.L. THOMSON (ed.), « *Industrial Scheduling* », Prentice-Hall, Englewood Cliffs, N.J., pp. 193-206 (1963).

12[] GOMORY, R., « An Algorithm for the Mixed Integer Problem », RAND Report P-1885, Feb. 1960.

[13] GOMORY, R., « Integer Faces of a Polyhedron », *Proc. Nat. Acad. Sci. U.S.A. 57,* (1), 16-18 (Jan. 1967).

[14] GOMORY, R., « Some Polyhedra Related to combinatorial Problems », J. Linear *Algebra and its Applications, 2* (4), (1969).

[15] HERVE, P., « Les procédures arborescentes d'optimisation », *Revue Française d'Informatique et de Recherche Opérationnelle,* 2, n° 14, pp. 69-80 (1968).

[16] HU, T.C., « *Integer Programming and Network Flows* », Addison-Wesley, Reading, Massachussetts, Menlo Park, California, London, Don Mills, Ontario (1969).

[17] LAND, A.H. et DOIG, A.G., « An Automatic Method for Solving Discrete Programming Problems », *Econometrica, 28,* pp. 497-520 (1960).

[18] LAWLER, E.L. and WOOD, D.E., « Branch and Bound Methods : A Survey », *Operations Research, 14,* pp. 669-719 (1966).

[19] LITTLE, J.D.C., MURTY, K.G., SWEENEY, D.W. and KAREL, C., « An Algorithm for the Travelling Salesman Problem », *Operations Research, 11,* pp. 972-989 (1963).

[20] MITTEN, L.G., « Branch and Bound Methods : General Formulations and Properties », *Operations Research, 18,* pp. 24-34 (1970).

[21] ROY, B., « Procédures d'exploration par séparation et évaluation (P.S.E.P. et P.S.E.S.) », *Revue Française d'Informatique et de Recherche Opérationnelle,* V. 1, pp. 61-90 (1969).

[22] ROY, B., « Algèbre Moderne et Théorie des Graphes », Dunod, Paris, Tome 1 (1969) et Tome 2 (1970).