REVUE BELGE DE STATISTIQUE, D'INFORMATIQUE ET DE RECHERCHE OPERATIONNELLE

Vol. 11 - N° 3
DECEMBRE 1971

BELGISCH TIJDSCHRIFT VOOR STATISTIEK, INFORMATIEK EN OPERATIONEEL ONDERZOEK

Vol. 11 - Nr 3
DECEMBER 1971

La « Revue Belge de Statistique, d'Informatique et de Recherche Opérationnelle » est publiée avec l'appui du Ministère de l'Education nationale et de la Culture, par les Sociétés suivantes :

SOGESCI. — Société Belge pour l'Application des Méthodes scientifiques de Gestion.
Secrétariat : rue du Neufchâtel, 66 - 1060 Bruxelles. Tél. 37.19.76.

S.B.S. — Société Belge de Statistique.
Siège social : rue de Louvain, 44 - 1000 Bruxelles.
Secrétariat : rue de Louvain, 44 - 1000 Bruxelles.

Het « Belgisch Tijdschrift voor Statistiek, Informatiek en Operationeel Onderzoek » wordt uitgegeven met de steun van het Ministerie van Nationale Opvoeding en Cultuur, door de volgende Verenigingen :

SOGESCI. — Belgische Vereniging voor Toepassing van Wetenschappelijke Methodes in het Bedrijfsbeheer.
Secretariaat : Neufchâtelstraat, 66 - 1060 Brussel - Tel. 37.19.76.

S.B.S. — Belgische Vereniging voor Statistiek.
Maatschappelijke zetel : Leuvensestraat, 44 - 1000 Brussel.
Secretariaat : Leuvensesetraat, 44 - 1000 Brussel.

REVUE BELGE DE STATISTIQUE
ET DE RECHERCHE OPERATIONNELLE

VOL. 11 - N° 3 - DECEMBRE 1971 VOL. 11 - Nr 3 - DECEMBER 1971

SOMMAIRE — INHOUD

BELGISCH TIJDSCHRIFT VOOR STATISTIEK
EN OPERATIONEEL ONDERZOEK

CONTRIBUTION A L'OPTIMISATION DE LA MISE EN VALEUR D'UNE MINE A CIEL OUVERT : MODELE PERMETTANT DE CALCULER LES COUTS PAR UNE SIMULATION QUI TIENT COMPTE DES POSSIBILITES DE REMBLAYAGE

J. FRANCOTTE et R. VALLET,
Centre d'Informatique Générale – Bruxelles

A. CASTIAUX, *Générale Congolaise des Mines Lubumbashi.*

1. LE PROBLEME

Lorsque la prospection géologique est terminée, le minerai est repéré dans l'espace et sa teneur peut être estimée en chaque point du gisement. Si un maillage découpe l'espace occupé par le gisement en blocs élémentaires, il est possible de constituer un fichier donnant pour chaque bloc élémentaire les volumes et les teneurs estimées.

Lorsqu'on connaît le cours commercial des produits contenus dans le minerai, les frais de mise sur marché ainsi que le rendement et le coût des traitements que le minerai devra subir avant que les produits qu'il contient soient commercialisables, il est possible d'affecter à chaque bloc une valeur en argent.

Si, par ailleurs, on connaît pour chaque bloc le coût de son extraction, on peut obtenir le profit (positif, nul ou néagitf) apporté par son extraction :

$$\text{profit} = \text{valeur} - \text{coût}$$

Or, l'exploitation d'une carrière à ciel ouvert obéit à la loi suivante :

un bloc ne peut être extrait si l'on n'a pas extrait auparavant tous les blocs pleins compris dans un cône renversé, ayant pour pointe le bloc à extraire et pour pente la pente de talutage exigée par la nature du terrain.

On dira qu'un ensemble de blocs respecte la loi d'excavation si l'on peut extraire chacun des blocs de cet ensemble sans avoir à extraire un bloc extérieur à cet ensemble.

On appellera profit d'un ensemble de blocs la somme des profits apportés par chacun des blocs qui le composent.

L'ensemble de blocs qui respecte la loi d'excavation et qui donne le profit maximum représente la forme optimale de la carrière.

Le problème de la recherche d'un tel ensemble a été résolu par MM. Helmut LERCHS et Ingo F. GROSSMANN, ref. TRANSACTION, CIM, Volume LXVIII, 1965, pages 17-24.

La forme optimale de la carrière est alors donnée par l'ensemble des branches fortes d'une arborescence normalisée.

Mais, tel qu'il est posé, ce problème demande que, pour chaque bloc, deux données soient fournies au départ.

La première de ces données, la valeur du bloc, ne dépend que de variables extérieures au problème : cours commerciaux des produits, caractéristiques du traitement que subira le minerai après son extraction. Elle peut donc être fixée à priori.

La deuxième de ces données, le coût d'extraction du bloc, se décompose en deux postes : le coût de fragmentation et le coût de transport.

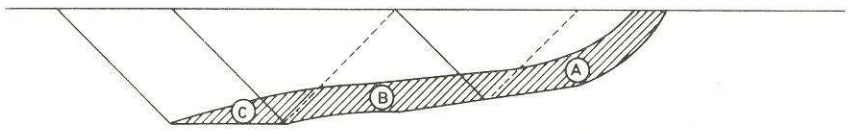
Le coût de fragmentation dépend d'une part de la dureté de la roche, donnée immuable, d'autre part du type de matériel et d'explosif employé, donnée qui peut être fixée à priori, le problème devant apporter une solution pour un type de matériel donné. Le coût de fragmentation pris isolément peut donc être déterminé à l'avance en fonction de la dureté de la roche.

Le coût du transport, quant à lui, dépend également d'un type de matériel qui peut être choisi à priori, mais aussi du trajet que le bloc devra suivre entre la position qu'il occupe in situ et son point de stockage (concentrateur ou remblai de stérile).

Le trajet d'un bloc de minerai entre sa position dans le gisement et le concentrateur est connu. Le coût du transport d'un bloc de minerai peut donc être déterminé à l'avance.

Par contre, suivant la succession chronologique des phases d'exploitation adoptée, un même bloc de stérile sera transporté soit sur un remblai situé hors de la carrière, soit dans la carrière en tel ou tel point qui peut être remblayé.

En effet, soit un gisement tel que celui qui est représenté sur le croquis.



Nous pouvons imaginer entre autres trois façons d'exploiter le gisement :

1. En une seule phase : exploitation simultanée A, B et C.
   Tous les stériles sont transportés hors de la carrière.
2. En trois phases :
   1e phase : exploitation de A. Les stériles sont transportés hors de la carrière.
   2e phase : exploitation de B. Les stériles sont transportés en A.
   3e phase : exploitation de C. Les stériles sont transportés en B.
3. En trois phases :
   1e phase : exploitation de C. Les stériles sont transportés hors de la carrière.
   2e phase : exploitation de B. Les stériles sont transportés en C.
   3e phase : exploitation de A. Les stériles sont transportés en B.

Dans tout gisement il sera possible d'effectuer une économie sensible sur les transports si, après avoir exploité une partie du gisement, il est possible de remblayer la partie déjà excavée avec les stériles extraits dans le reste du gisement.

Suivant la forme qu'a la carrière au moment où un bloc de stérile est excavé, celui-ci est transporté soit sur un back-filling en tel ou tel point de la carrière, soit sur un remblai hors de la carrière. Cette forme de la carrière à ce moment-là est elle-même fonction des opérations qui ont précédé, en particulier de la succession des phases d'exploitation.

Le problème est alors de trouver quelle est la succession de phases d'exploitation qui permet d'obtenir le profit maximum pour l'ensemble du gisement et quelles seront les limites de la carrière à la suite de chaque phase.

2. LE PROBLEME SERA RESOLU PAR UNE SIMULATION

Lorsque plusieurs projets, comportant chacun une succession de phases différente, ont été définis, si l'on dispose d'un programme capable de calculer le profit réalisé par l'exploitation de chacun de ces projets, le problème peut être résolu.

La comparaison entre les profits réalisés par chacun des projets proposés permettra de sélectionner le meilleur d'entre eux. On peut procéder à plusieurs sélections successives et obtenir ainsi la définition d'un projet qui, s'il n'est pas théoriquement le meilleur, est toutefois le meilleur de ceux que l'on a pu imaginer.

Mais, pour pouvoir optimiser la forme de la carrière dans chaque projet et pour pouvoir calculer le profit réalisé par chaque projet, il faut pouvoir calculer le coût de transport de chaque bloc. Pour pouvoir calculer le coût de transport d'un bloc de stérile, il faut connaître la forme de la carrière au moment de l'extraction de ce bloc, forme qui pour un même bloc peut varier d'un projet à l'autre.

Cette condition sera remplie si l'on dispose d'un processus de simulation permettant de connaître la forme de la carrière et toutes les conséquences qui en découlent, après l'extraction de chaque bloc.

Cette simulation devra en premier lieu se soumettre aux contraintes propres à l'exploitation d'une carrière.

De plus, les données devront lui être présentées sous une forme appropriée.

Elle simulera enfin un type défini d'exploitation.

2.1. Les contraintes propres à l'exploitation d'une carrière.

211. *Contrainte d'excavation.*

Nous l'avons déjà vu plus haut, un bloc ne peut être extrait si l'on n'a pas extrait auparavant tous les blocs pleins situés dans un cône renversé ayant pour pointe le bloc à extraire et pour pente la pente de talutage exigée par la nature du terrain.

212. *Contraintes de remblayage.*

1. Un bloc vide ne peut pas être remblayé si dans le cône pointe en haut qu'il surmonte, ayant pour pente la pente exigée par la nature du terrain, il se trouve un bloc minéralisé situé au-dessus d'un niveau limite donné.

En effet, si l'on remblayait un tel bloc vide, ce bloc devrait être vidé au moment où l'on excaverait le bloc minéralisé situé dans le cône au-dessous de lui.

2. Plusieurs blocs remplayés peuvent s'empiler les uns sur les autres, mais la pente de talutage d'une roche fragmentée est toujours inférieure à la pente de talutage d'une même roche en place.

3. Un bloc de roches quelconques, une fois fragmenté et transporté, occupe un volume plus grand que le volume qu'il occupait lorsqu'il était en place. Le coefficient de foisonnement est le rapport :

(volume fragmenté)/(volume en place).

2.2. Forme sous laquelle les données sont présentées à la simulation.

D'une part, pour faciliter la définition des phases, le gisement est morcelé en gisements partiels.

Par ailleurs, le maillage est conçu pour faciliter la génération des cônes demandés par les contraintes d'excavation.

Enfin, à chaque bloc généré par le maillage sont affectées trois données :

221. *Morcellement du gisement en gisements partiels.*

Une phase d'exploitation est définie par les limites à l'intérieur desquelles on se propose d'enlever le minerai au cours de l'exploitation de la phase.

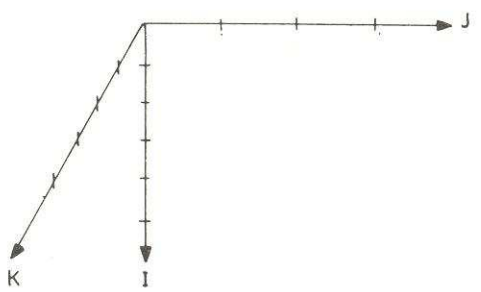
Le gisement est découpé en plusieurs gisements partiels identifiés par un numéro, même si le gisement se présente comme un ensemble continu.

La phase est alors définie par la profondeur que doit atteindre l'exploitation dans chaque gisement partiel.

222. *Le maillage dans lequel doivent s'inscrire les données qui caractérisent le gisement.*

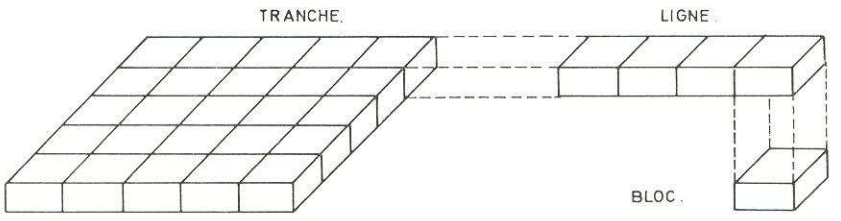
Les données géologiques que l'on possède au départ sont connues dans les trois dimensions.

Soient I, J, K les trois axes de coordonnées rectangulaires. Si l'on découpe chacun de ces trois axes en segments d'une unité de longueur



élémentaire, on obtient un réseau de maillage à trois dimensions qui génère des blocs cubiques ou parallélipipédiques suivant que l'on a adopté ou non la même unité de longueur élémentaire pour chacun des trois axes.

Pour la facilité du langage, on appellera tranche l'ensemble des blocs d'un même niveau I, et ligne l'ensemble des blocs d'une tranche ayant la même ordonnée K.

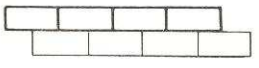


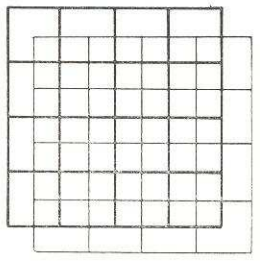
On connaît pour chaque bloc élémentaire les roches qui le composent, la quantité de minerai de chaque catégorie qu'il contient et la teneur de ces minerais. La simulation peut alors être effectuée en considérant le bloc comme unité d'extraction : tel bloc est minéralisé, il doit être extrait ainsi que tous les blocs situés dans le cône renversé qui le surmonte.

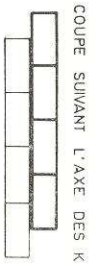
Reste à trouver un assemblage des blocs qui facilite la génération de ce cône.

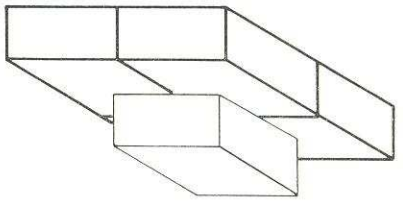
Si l'on décale horizontalement d'un demi-bloc en J et en K chaque tranche par rapport à la tranche supérieure, on obtient la disposition ci-dessous.

Coupe suivant l'axe des J









Vue en plan des deux tranches.

L'enlèvement d'un bloc nécessite l'enlèvement à la tranche supérieure des quatre blocs qui le recouvrent en partie. En remontant ainsi de proche en proche, on génère non pas un cône mais une pyramide à quatre faces. On considère que la substitution d'une pyramide au cône théorique ne fausse pas d'une manière sensible les résultats pourvu qu'on donne à ses faces les pentes demandées par la nature du terrain.

Si l'on adopte pour l'ensemble de la carrière une pente uniforme de talutage, les dimensions du bloc élémentaire seront choisies de telle sorte que la pente de la pyramide ainsi générée corresponde à la pente de talutage adoptée. (Voir en annexe 1 les pentes générées par le programme en fonction des dimensions du bloc élémentaire).

223. *Les données de chaque bloc élémentaire.*

La simulation a besoin pour chaque bloc des renseignements suivants :

223.1. Sa valeur argent.

Celle-ci est calculée avec la simulation à partir du tonnage des métaux contenus dans le bloc, du cours commercial des métaux, du coût et du rendement des traitements métallurgiques que le minerai devra subir avant la commercialisation des métaux. La valeur d'un bloc n'est pas changée en cours de simulation.

223.2. Un code fixe qui définit la dureté de la roche.

La dureté de la roche est connue au départ et le code fixe n'est pas changé en cours de simulation.

223.3. Un code variable qui définit le statut momentané du bloc.

1. Au départ tous les blocs sont connus comme ayant l'un des statuts suivants :

— Bloc vide.
— Bloc minéralisé appartenant au gisement partiel n° 1.
— Bloc minéralisé appartenant au gisement partiel n° 2.
— Bloc minéralisé appartenant au gisement partiel n° *n*.
— Bloc stérile.

Le code variable correspondant leur est attribué.

2. En cours de simulation, le statut du bloc peut changer et de nouveaux statuts qui n'existaient pas au début de la simulation peuvent apparaître.

Un bloc plein (stérile ou minéralisé) peut devenir un « Bloc repéré pour être excavé » avant de devenir un « Bloc vide ».

Un bloc vide peut devenir un « Bloc qu'il est permis de remblayer » avant de devenir, un « Bloc remblayé ».

Chaque fois que le statut du bloc change, le code variable prend la valeur correspondant au nouveau statut du bloc, si bien qu'à tout moment de la simulation l'ensemble des codes variables définit la forme momentanée de la carrière.

2.3. Définition de l'exploitation simulée par le programme.

231. *Succession des opérations.*

231.1. A l'intérieur d'un projet.

Toutes les opérations demandées par l'exploitation d'une phase sont effectuées avant de commencer la phase suivante.

C'est là l'exigence qui conditionne la variation du profit entre divers projets.

231.2. A l'intérieur d'une phase.

Les tranches sont exploitées en descendant depuis la tranche la plus haute à exploiter par la phase, jusqu'à la plus basse.

Toutes les opérations demandées par l'exploitation d'une tranche sont effectuées avant de commencer l'exploitation de la tranche suivante.

232. *Définition de l'exploitation d'une tranche.*

1. A la tranche qu'on exploite : extraction de tous les blocs minéralisés appartenant à un gisement partiel que la phase doit exploiter.

2. Aux tranches supérieures : extraction de tous les blocs pleins compris dans les pyramides renversées qui surmontent les blocs minéralisés excavés à la tranche que l'on exploite.

3. LA SIMULATION DE L'EXPLOITATION D'UNE TRANCHE

La simulation de l'exploitation d'une tranche doit se faire en trois étapes :

1. Le repérage des blocs à excaver.
2. Le calcul du coût d'extraction des blocs repérés.
3. L'inventaire des blocs vides qui peuvent être remblayés à la suite de l'exploitation de la tranche.

3.1. Le repérage des blocs à excaver pour l'exploitation de la tranche.

Ce repérage se fera en affectant aux blocs à excaver le code variable correspondant au statut « Bloc repéré pour être excavé ».

Dans la tranche à exploiter, tous les blocs minéralisés appartenant aux gisements partiels qui doivent être exploités par la phase sont repérés.

Dans chacune des pyramides renversées qui surmontent un bloc repéré de la tranche à exploiter, tous les blocs pleins sont également repérés. Ces pyramides peuvent être incomplètes : elles peuvent contenir soit des blocs vides, soit des blocs déjà repérés par l'exploration d'une pyramide voisine.

Or, tout bloc vide est lui-même surmonté par une pyramide vide. Un bloc déjà repéré est également surmonté, si le repérage a été bien mené, par une pyramide ne comprenant que des blocs vides ou des blocs déjà repérés.

Un procédé d'exploration de la pyramide pile par pile permet de n'explorer que la partie de la pyramide qui comporte des blocs pleins non encore repérés.

3.2. Le calcul du profit apporté par l'extraction d'un bloc repéré.

Une fois que sont repérés les blocs qui doivent être excavés pour l'exploitation de la tranche, le programme calcule le profit apporté par l'extraction de chacun des blocs repérés.

Lorsque le profit d'un bloc est calculé, le code variable correspondant au statut « Bloc vide » lui est attribué.

Le profit (positif, nul ou négatif) réalisé par l'extraction d'un bloc est égal à :

valeur — coût d'extraction

Le coût d'extraction se décompose lui-même en coût de fragmentation et coût de transport. La valeur du bloc est donnée. Le calcul du coût de fragmentation est simple. La difficulté réside dans le calcul du coût du transport.

321. *Le point de destination du transport du bloc.*

Si le bloc est minéralisé, il est transporté au concentrateur.

Si le bloc est stérile, et s'il existe un back-filling en cours de remplissage, le bloc est transporté au point de versement de ce back-filling.

S'il n'y a pas de back-filling en cours de remplissage, mais s'il existe des back-fillings enregistrés dans le catalogue, le programme choisit dans le catalogue le back-filling dont le plancher est le plus bas. Le back-filling choisi est alors supprimé du catalogue. Il est dès lors considéré comme back-filling en cours de remplissage et le bloc de stérile est transporté au point de versement de ce back-filling.

Si enfin il n'y a pas de back-filling en cours de remplissage, et s'il n'existe pas de back-filling enregistré dans le catalogue, le bloc de stérile est transporté sur le remblai situé hors de la carrière.

Lorsqu'un bloc est transporté hors de la carrière, le trajet se décompose en deux parties :

Le trajet 1 : trajet entre le point que le bloc occupe dans le gisement et le point de sortie de la carrière. Le trajet 1 varie pour chaque bloc.

Le trajet 2 : trajet entre le point de sortie de la carrière et le concentrateur si c'est un bloc minéralisé; trajet entre le point de sortie de la carrière et le remblai si c'est un bloc stérile. Le trajet 2 est le même pour tous les blocs. Ses caractéristiques sont données.

Lorsqu'un bloc est transporté sur un back-filling, nous n'avons que le trajet entre le point que le bloc occupe dans le gisement et le point de versement sur le back-filling, trajet qui varie pour chaque bloc. Mais par convention nous appellerons ce trajet le trajet 1, auquel vient s'ajouter un trajet 2 dont les caractéristiques sont nulles.

322. *Les paramètres de calcul du coût du transport.*

Pour un matériel de transport donné, le coût unitaire du transport, unité de volume/unité de longueur, étant connu, le coût du transport est le produit de ce coût unitaire par le cube transporté et une distance que les mineurs appellent distance standard.

La distance standard est obtenue en ajoutant à la longueur du trajet : d'une part une longueur constante qui tient compte des diverses manœuvres que l'engin de transport doit effectuer au point de chargement et au point de déchargement, d'autre part le produit par un certain coefficient de la dénivellation que l'engin chargé doit franchir en montée et qui représente le supplément d'énergie consommé dans les montées.

La longueur constante et le coefficient qui multiplie la dénivellation varient avec chaque type de matériel de transport. La simulation est effectuée pour un type donné de matériel de transport.

Pour chaque bloc la simulation devra donc déterminer :

D : longueur du trajet

H : dénivellation que le bloc doit franchir dans le sens de la montée.

Soient D1 la longueur du trajet 1, H1 la dénivellation que le bloc doit franchir dans le sens de la montée, au cours du trajet 1 ; D2 et H2 les valeurs correspondantes du trajet 2 :

$$D = D1 + D2$$

$$H = H1 + H2$$

D2 et H2 sont connus au départ.

La simulation devra calculer D1 et H1.

323. *La longueur du trajet 1.*

Lec deux extrémités du trajet 1 étant repérées dans les trois dimensions, la droite qui unit dans l'espace ces deux extrémités représente la longueur minimum du trajet 1, pour autant qu'il ne soit soumis à aucune contrainte.

Mais la pente du trajet 1 ne peut dépasser une pente que l'on s'est imposée pour les plans inclinés de la carrière. La longueur nécessaire à un plan incliné pour franchir toutes les dénivellations qui se présentent le long du trajet 1, aussi bien dans le sens de la montée que dans le sens de la descente, est la longueur minimum résultant de la contrainte de pente.

La plus grande de ces deux longueurs représente la longueur minimum qu'aura le trajet 1.

La longueur D1 qui servira au calcul du coût est obtenue en multipliant sa longueur minimum par un coefficient approprié.

324. *Les dénivellations qui se présentent le long du trajet 1.*

324.1 Si le bloc est transporté hors de la carrière, on considère qu'il n'a pas de seuil à franchir pour atteindre la sortie de la carrière.

Si la sortie de la carrière est plus haute que le bloc, la dénivellation franchie en montée est positive et égale en valeur absolue à la différence des altitudes des deux extrémités, tandis que la dénivellation franchie en descente est nulle.

Si la sortie de la carrière est plus basse que le bloc, la dénivellation franchie en montée est nulle, tandis que la dénivellation franchie en descente est égale en valeur absolue à la différence d'altitude des deux extrémités.

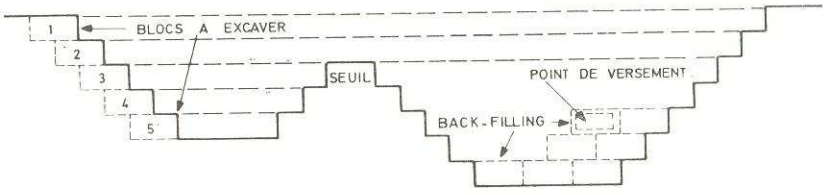
324.2. Si le bloc est transporté sur un back-filling, il arrive fréquemment qu'au cours du trajet le bloc doive franchir un seuil.

Pour connaître les dénivellations qui se présentent le long du trajet, il faut alors connaître dans quelle tranche se trouve le point haut du transport.

La dénivellation franchie en montée est alors la différence d'altitude entre le point haut et le bloc.

La dénivellation franchie en descent est la différence d'altitude entre le point haut et le point de versement sur le back-filling.

Dans le croquis ci-dessous, le point haut du transport du bloc 1 est dans la tranche qui contient le bloc 1. Le point haut du transport du bloc 2 est dans la tranche qui à la fois contient le bloc 2 et tangente le seuil. Le point haut du transport des blocs 3, 4 et 5 est dans la tranche qui tangente le seuil.



La plus basse des tranches dans laquelle peut se trouver le point haut est la plus élevée des deux tranches qui contiennent les deux extrémités du transport. C'est à partir de cette tranche qu'est recherchée en remontant la tranche qui contient le point haut.

La pyramide qui surmonte le bloc à excaver est vide.

Le point de versement a été choisi dans un bloc vide. La pyramide renversée qui le surmonte est donc vide également.

Le trajet, depuis le bloc à excaver jusqu'au point de versement sur le back-filling, devra obligatoirement passer par une tranche dans laquelle il est possible d'aller d'une pyramide à l'autre sans rencontrer de blocs pleins. Or, les tranches qui remplissent cette condition appartiennent à deux catégories :

— d'une part les tranches dans lesquelles les deux pyramides se recoupent,

— d'autre part, au-dessous des précédentes, les tranches dans lesquelles les blocs vides appartenant à la première pyramide et les blocs vides appartenant à la deuxième pyramide font partie d'un même ensemble continu de blocs vides.

Le point haut du parcours sera dans la plus basse des tranches qui appartiennent à l'une ou l'autre de ces catégories.

3.3. L'inventaire des blocs vides qui peuvent être remblayés à la suite de l'exploitation d'une tranche.

Il est possible qu'à la suite de l'exploitation de la tranche on puisse remblayer des blocs étagés sur plusieurs tranches superposées. Dans certains cas, l'empilement des blocs qui peuvent être remblayés remonte jusqu'au sommet de la carrière. Or, le remplissage d'une partie vide de la carrière s'effectue en déversant les produits stériles par le haut, mais cette hauteur de déversement a une limite. Pour l'établissement du programme, on a fixé cette limite à trois tranches.

L'ensemble des blocs vides qui pourront être remblayés à partir d'un même niveau de déversement fera l'objet d'un seul enregistrement dans le catalogue des back-fillings.

C'est un tel ensemble que nous désignerons désormais sous l'appellation de back-filling.

Un back-filling est caractérisé par :

— un numéro

— sa contenance ou nombre de blocs de terrain en place qu'il peut contenir

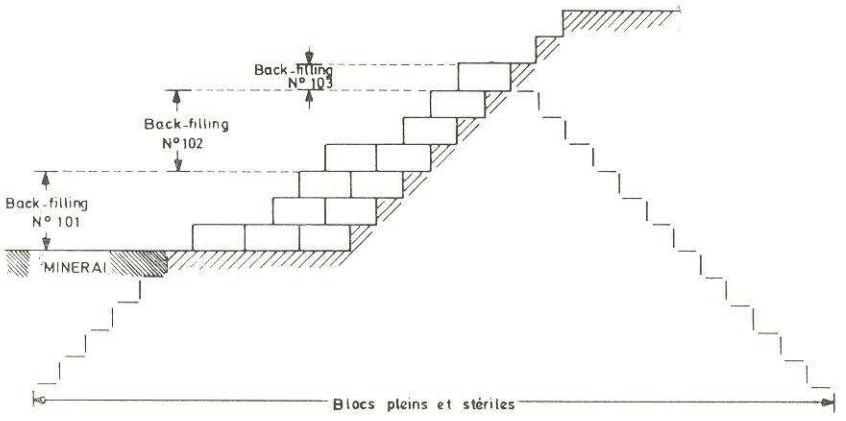
— son épaisseur ou nombre de tranches superposées qu'il comporte

— les coordonnées de son point de versement.

Comme point de versement sur le back-filling, le programme adopte le bloc qui occupe une position médiane dans la tranche supérieure du back-filling. Si la forme de la tranche est telle que ce bloc ne fait pas partie du back-filling, il choisit le bloc le plus proche faisant partie du back-filling.

L'ensembles des blocs vides qu'il est permis de remblayer, rencontrés dans les trois premières tranches comptées en remontant à partir de la tranche qui vient d'être exploitée, constituera un premier back-filling (back-filling n° 101 du croquis). L'ensemble des blocs remblayables rencontrés

dans les trois tranches suivantes constituera un deuxième back-filling (n° 102 du croquis), etc.



331. *Le respect des contraintes de remblayage.*

Ces contraintes ont été définies en 212.

331.1. L'empilement des blocs.

L'empilement adopté pour les blocs d'un back-filling est le suivant :

La tranche médiane est décalée d'un bloc et demi par rapport à la tranche inférieure. Toutes les autres tranches sont décalées d'un demi-bloc.

En procédant ainsi, quelles que soient les dimensions adoptées pour les blocs, la pente de talutage du back-filling exprimée en pourcentage (tangente de l'angle de pente) est égale aux 3/5 de la pente adoptée pour la carrière. (Voir en annexe 1 les pentes de remblai générées par le programme en fonction des dimensions du bloc élémentaire).

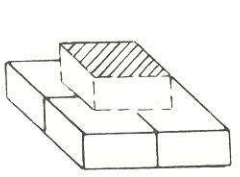
331.2. Conditions pour qu'un bloc vide puisse être remblayé.

Un bloc vide peut être remblayé s'il repose sur des blocs pleins et si l'excavation d'un bloc minéralisé ne demande pas par la suite que l'un de ces blocs pleins soit vidé. L'ensemble des blocs sur lesquels repose le bloc vide, situés dans la tranche immédiatement au-dessous du bloc vide, forme l'assise du bloc vide.

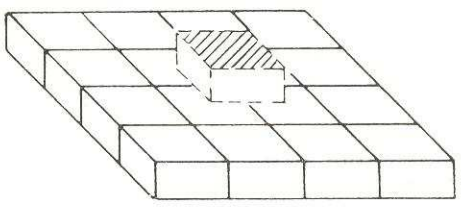
Pour respecter la pente de talutage du back-filling, l'assise du bloc vide aura des dimensions différentes suivant la tranche du back-filling à laquelle

appartient le bloc vide. Si le bloc vide appartient à la première ou à la troisième tranche d'un back-filling, son assie compte $2 \times 2 = 4$ blocs. Si le même bloc appartient à la deuxième tranche de back-filling, son assise compte $4 \times 4 = 16$ blocs.

Dimensions de l'assise d'un bloc vide



Sous un bloc vide de la 1ère ou 3ème tranche d'un back-filling.



Sous un bloc vide de la 2ème tranche d'un back-filling.

Un bloc vide pourra être remblayé seulement si son assise ne comporte que des blocs des catégories suivantes :

— bloc stérile surmontant une pyramide stérile
— bloc stérile surmontant une pyramide minéralisée au-dessous de la tranche limite
— bloc remblayé
— bloc vide qu'il est permis de remblayer.

L'application de cette règle a pour conséquence que l'exploitation d'un bloc minéralisé d'une tranche inférieure, non seulement ne peut pas vider le bloc remblayé, mais encore ne peut pas modifier la pente de talutage adoptée pour le back-filling.

331.3. Le foisonnement.

La contenance d'un back-filling est obtenue en divisant le nombre de blocs repérés pour constituer le back-filing par un coefficient de foisonnement.

Le coefficient de foisonnement adopté par le programme est 1, 3.

332. Les opérations d'inventaire.

Après l'exploration d'une tranche, le problème est d'enregistrer dans le catalogue les back-fillings constitués par les blocs vides qui peuvent être

remblayés à la suite de l'explosion de la tranche. Ces back-fillings enregistrés ne pourront être remblayés qu'au cours des exploitations ultérieures.

Dans la tranche qui vient d'être exploitée, le programme repère les blocs vides qui peuvent être remblayés.

Si ce repérage a trouvé des blocs susceptibles d'être remblayés, le même repérage est effectué à la tranche supérieure et ainsi de suite. Lorsqu'on a repéré les blocs vides de trois tranches consécutives, on a un back-filling complet et les caractéristiques de ce back-filling sont enregistrées.

Lorsqu'enfin le repérage ne trouve pas de bloc à remblayer à une tranche donnée, l'opération de repérage est arrêtée. Reste éventuellement à enregistrer le back-filling constitué par les blocs repérés des tranches inférieures.

4. LES RESULTATS DE LA SIMULATION

Les résultats de la simulation peuvent être présentés sous diverses formes :

4.1. Un tableau numérique dans lequel sont présentées les informations qui permettent de suivre la variation du profit cumulé au cours de l'exploitation du projet.

4.2. Une présentation de la situation de la carrière, à la suite de l'exploitation de chaque phase soit sous forme de plans de niveaux (voir annexe 2), soit sous forme de coupes (voir annexe 3).

4.3. Un fichier des profits qui pour chaque bloc extrait contient le profit positif nul ou négatif égal à la valeur du bloc, diminuée du coût de son extraction.

5. CONCLUSION

Cette simulation employée seule permet de déterminer la profondeur maximum à laquelle il faut exploiter la carrière pour obtenir le profit cumulé maximum, dans l'hypothèse où à chaque tranche exploitée tout le minerai est extrait.

Si, par ailleurs, on veut déterminer la forme optimale de la carrière qui procurera le profit maximum, cette simulation nous fournit pour chaque bloc un coût qui tient compte des possibilités d'économies sur les frais de transport résultant des remblayages effectués par une exploitation dont on a au préalable déterminé les phases.

Il est alors possible d'entrer dans un processus d'optimisation par arborescence, qui nous donnera la forme optimale de la carrière dans les cas où son exploitation s'effectue suivant la succession de phases adoptée pour le calcul des coûts.

ANNEXE 1

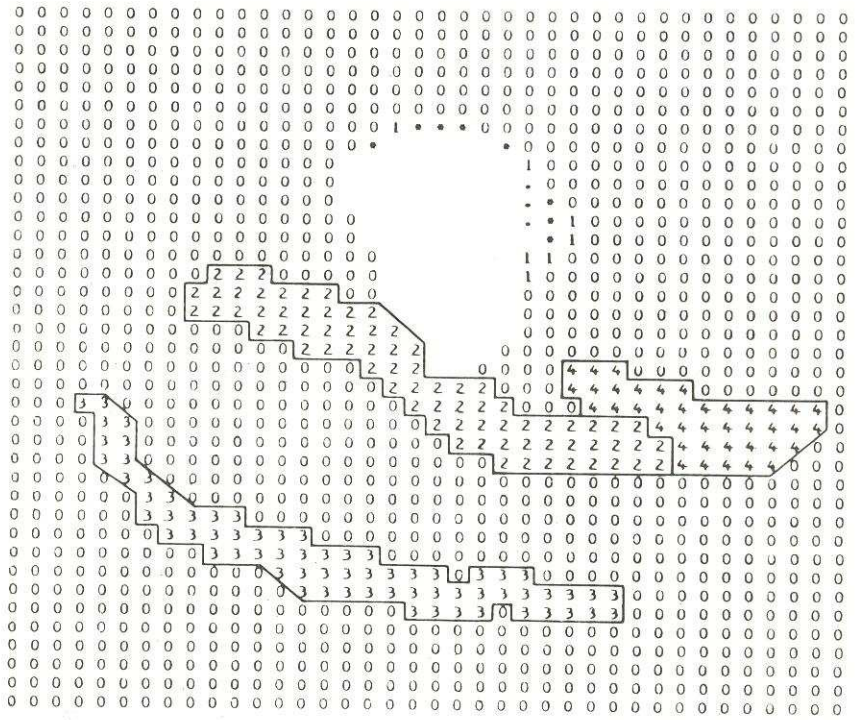
Pentes générées par le programme en fonction de diverses dimensions du bloc élémentaire.

Dimensions des blocs L × l × H	Pente de talutage de la carrière		Pente de talutage des remblais de back-filling	
	Pourcentage = 2 H/L	Angle	Pourcentage = 6H/5L	Angle
20 × 20 × 10	100 %	45°	60 %	30°58'
40 × 40 × 20	»	»	»	»
25 × 25 × 10	80 %	38°40'	48 %	25°38'
50 × 50 × 20	»	»	»	»
30 × 30 × 10	66,67 %	33°41'	40 %	21°48'
60 × 60 × 20	»	»	»	»
35 × 35 × 10	57,14 %	29°45'	34,28 %	18°56'
70 × 70 × 20	»	»	»	»
40 × 40 × 10	50 %	26°34'	30 %	16°42'
80 × 80 × 20	»	»	»	»
45 × 45 × 10	44,44 %	23°58'	26,67 %	14°56'
90 × 90 × 20	»	»	»	»
50 × 50 × 10	40 %	21°48'	24 %	13°30'
100 ×100 × 20	»	»	»	»

ANNEXE 2

Tranche No 17 — Niveau 1120

Situation à la fin de la phase 6 du projet 2.



1, 2, 3... = N° de gisement partiel d'un bloc minéralisé.

0 = Bloc stérile.

* = Bloc remblayé par un back-filling.

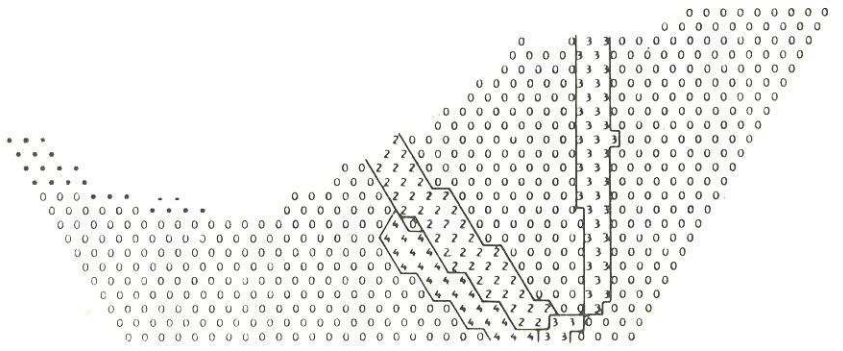
. = Bloc vide disponible pour un back-filling.

Blanc = Bloc vide.

ANNEXE 3

Coupe suivant la section X = 436590

Situation à la fin de la phase 6 du projet 2.



1, 2, 3... = N° de gisement partiel d'un bloc minéralisé.

0 = Bloc stérile.

* = Bloc remblayé par un back-filling.

. = Bloc vide disponible pour un back-filling.

Blanc = Bloc vide.

LA LOCALISATION DES ENTREPOTS

Léon KAUFMAN
Vrije Universiteit te Brussel

1. Introduction.

1.1. Localisation des entrepôts.

Le problème de l'établissement d'un réseau d'entrepôts se pose lorsqu'une entreprise doit satisfaire les demandes d'un grand nombre de clients géographiquement dispersés. L'entreprise peut expédier les produits demandés directement des usines aux clients ou bien utiliser des entrepôts. Dans ce dernier cas, des expéditions en gros peuvent être faites des usines aux entrepôts, par des moyens de transport économiques (par exemple par chemin de fer ou par gros camions). L'usage d'entrepôts permet donc de diminuer les coûts de transport, mais entraîne des dépenses d'établissement et de fonctionnement.

Le problème est de déterminer le nombre, les tailles et les emplacements des entrepôts du réseau projeté de manière à minimiser les frais d'entreposage et de transport.

1.2. Localisation des centres de service.

On rencontre dans le domaine public des problèmes de localisation de centres de service (par exemple écoles ou hôpitaux) semblables au problème de la localisation des entrepôts; les coûts de transport sont généralement à charge des usagers, l'objectif que l'on cherche à maximiser est une fonction de leur satisfaction, exprimée par exemple en termes monétaires ou en termes de kilomètres parcourus.

Le budget et par conséquent le nombre maximum de centres de service sont souvent décidés à priori par l'autorité publique et non par les planificateurs.

Le problème est de déterminer les emplacements et les tailles d'un nombre fixé de centres de service de manière à maximiser l'utilité des usagers.

1.3. Méthodes de résolution.

Depuis le XIX^{e} siècle existent des modèles pour la localisation d'un centre de service sur un plan : ces modèles utilisent des techniques de calcul différentiel.

Le problème de la localisation simultanée de plusieurs centres de service mène rapidement à des systèmes d'équations fort complexes. De plus, il est clair qu'un centre ne peut en pratique être localisé n'importe où.

Grâce aux techniques de la recherche opérationnelle, des modèles plus réalistes et permettant d'aborder des problèmes plus complexes ont été élaborés depuis une dizaine d'années. Dans ces modèles, on recherche la localisation optimale d'un nombre fixé ou même inconnu de services parmi un ensemble fini d'emplacements préalablement choisis.

Des procédures d'optimisation par séparation permettent d'obtenir la solution optimale pour ces problèmes. Depuis la parution, en 1965, de l'algorithme de EFROYMSON et RAY [7], ces modèles ont été utilisés en pratique par de nombreuses entreprises pour résoudre leurs problèmes de localisation.

Comme les modèles ne donnent qu'une approximation de la situation réelle, les résultats obtenus doivent être considérés aussi comme approchant seulement l'optimum réel.

1.4. Problèmes étudiés.

Dans la suite de cet article, on étudie principalement un modèle pour la localisation des entrepôts dû à M. L. BALINSKI [2]. On expose l'algorithme exact d'Efroymson et Ray et six algorithmes heuristiques. Ensuite, trois séries d'expériences permettent de comparer les performances de ces algorithmes.

Enfin quelques extensions du modèle de Balinski sont brièvement discutées.

2. Un modèle mathématique.

2.1. Enoncé du modèle de Balinski.

Soient m emplacements, repérés par l'indice i, où peuvent être établis des entrepôts.

L'établissement d'un entrepôt est noté à l'aide d'une variable indicatrice y_i :

$$y_i \in \{0, 1\} \qquad (i = 1, 2, ..., m) \qquad (1)$$

y_i prend la valeur 1 si un entrepôt est établi à l'emplacement i et sinon la valeur 0.

Soient n clients, repérés par l'indice j, desquels émanent des demandes d_j . S'il existe un entrepôt en i, la demande d_j peut être satisfaite à partir

de l'entrepôt i dans la proportion x_{ij} ; s'il n'y a pas d'entrepôt en i, aucune demande ne peut être satisfaite à partir de i et $x_{ij} = 0$ pour tout j :

$$0 \leqslant x_{ij} \leqslant y_i \leqslant 1 \qquad (i = 1, 2, ..., m),\ (j = 1, 2, ..., n) \qquad (2)$$

Toutes les demandes doivent être satisfaites.

$$\sum_i x_{ij} = 1 \qquad (i = 1, 2, ..., m),\ (j = 1, 2, ..., n) \qquad (3)$$

Les données numériques sont les coûts fixes f_i pour l'établissement d'un entrepôt à l'emplacement i et les coûts c_{ij} pour satisfaire la demande d_j à partir de l'entrepôt établi en i.

Le but est de minimiser, sous les contraintes (1), (2) et (3), la fonction économique :

$$z = \sum_i \sum_j c_{ij}\, x_{ij} + \sum_i f_i\, y_i \qquad (i = 1, 2, ..., m),\ (j = 1, 2, ..., n) \qquad (4)$$

Dans ce modèle, appelé M1 dans la suite, on fait l'hypothèse que les capacités des entrepôts sont illimitées (simple warehouse location problem). Dans la solution optimale, la demande de chaque client j sera entièrement satisfaite à partir de l'entrepôt k tel que :

$$c_{kj} = \min_i \{c_{ij} \mid y_i = 1\} \qquad (5)$$

Si les coûts c_{ij} sont des fonctions linéaires des distances entre i et j, l'équation (5) signifie que la demande de chaque client est satisfaite à partir de l'entrepôt le plus proche. Il existe toujours une solution optimale telle qu'une variable x_{ij} par colonne prenne la valeur 1 et les autres variables la valeur 0 (en cas d'égalité de certains coefficients c_{ij}, d'autres types de solution existent également).

Le modèle mathématique M1 décrit par les équations (1), (2), (3) et (4) ci-dessus a été utilisé par Efroymson et Ray, par K. Spielberg [19], [20] et par divers autres auteurs [1], [10], [21], [22]. On peut substituer les termes généraux de « source » et « puits » à « entrepôts » et « client » dans l'énoncé. Cette terminologie permet d'appliquer le même modèle au problème de la détermination des emplacements et des tailles optimales d'un réseau d'usines pour minimiser les frais de production et de distribution.

2.2. Un algorithme exact.

Le problème décrit par le modèle M1 est un programme linéaire mixte et ne peut être résolu par l'algorithme du simplexe car les variables y_i pourraient prendre des valeurs fractionnaires dans la solution optimale. Les

algorithmes exacts qui se sont montrés efficaces pour résoudre ce problème sont tous des procédures d'optimisation par séparation.

L'algorithme LOCA 1, utilisé dans les expériences décrites ci-dessous, est une version modifiée de celui de Efroymson et Ray; LOCA 1 est une procédure d'optimisation par séparation et évaluation séquentielle (P.S.E.S.) et non par séparation et évaluation progressive (P.S.E.P.). Ceci permet de diminuer fortement la capacité de mémoire requise et d'éviter des sous-routines de classement lors de la résolution sur ordinateur. La formulation de Efroymson et Ray est légèrement différente de celle employée par Balinski; les contraintes :

$$x_{ij} \leqslant y_i \tag{6}$$

sont remplacées par les contraintes :

$$\sum_j x_{ij} \leqslant n_i \, y_i \tag{7}$$

où n_i est le nombre de clients plus proches de i que d'un autre entrepôt ouvert. A l'étape initiale $n_i = n$ pour tout i ,mais lorsque des variables y_i sont fixées à 1, les n_i diminuent rapidement.

Le principe de séparation utilisé dans l'algorithme consiste à choisir une variable y_i à mettre soit à 0 soit à 1. A une itération K, on désigne par :

K_0 : l'ensemble des indices des variables Y_i mises à 0

K_1 : l'ensemble des indices des variables Y_i mises à 1 et

K_2 : l'ensemble des indices des variables Y_i libres, c'est-à-dire non encore fixées à 0 ou 1.

Le problème à une itération K se réduit à :

$$\min \left(\sum_{i \in K_2} f_i \, y_i + \sum_{i \in K_1} f_i + \sum_{i \in K_1 \cup K_2} \sum_j c_{ij} \, x_{ij} \right) \tag{8}$$

$$\sum_{i \in K_1 \cup K_2} x_{ij} = 1 \qquad j = 1, \ldots, n \tag{9}$$

$$\sum_{j=1}^{n} x_{ij} \leqslant n_i \, y_i \qquad i \in K_2 \tag{10}$$

$$0 \leqslant x_{ij} \qquad j = 1, \ldots, n, \; i \in K_2 \tag{11}$$

$$y_i \in \{0, 1\} \qquad i \in K_2 \tag{12}$$

LOCA 1 utilise un test d'optimalité et un test d'optimalité conditionnelle.

La valeur de la fonction d'évaluation employée pour le test d'optimalité à chaque itération est la solution du problème obtenu en remplaçant les contraintes (12) d'intégralité des variables y_i par :

$$0 \leqslant y_i \leqslant 1 \qquad i \in K_2 \tag{12 bis}$$

Cette valeur est facilement obtenue grâce au théorème suivant :

Théorème : Dans la solution optimale de (8), (9), (10), (11) et (12 bis) :

$$y_i = \frac{1}{n_i} \sum_j x_{ij} \qquad i \in K_2 \tag{13}$$

Démonstration : La démonstration se fait par l'absurde. Supposons que dans la solution optimale existe un indice i appartenant à l'ensemble K_2 tel que :

$$y_i > \frac{1}{n_i} \sum_j x_{ij} \tag{14}$$

la valeur de y_i peut être diminuée jusqu'à ce qu'on obtienne l'égalité tout en laissant les autres x_{ij} et y_i inchangés. Comme on suppose que les f_i sont strictement positifs, ceci diminue la valeur de la fonction économique; ce qui contredit l'optimalité. CQFD.

Pour obtenir la fonction d'évaluation, on remplace y_i par $\sum_j \frac{x_{ij}}{n_i}$ dans la fonction économique, ce qui donne le problème suivant :

$$\min \Big(\sum_{i \in K_1} f_i + \sum_{i \in K_2} \sum_j \frac{f_i \, x_{ij}}{n_i} + \sum_{i \in K_1 \cup K_2} \sum_j c_{ij} \, x_{ij} \Big) \tag{15}$$

$$\sum_{i \in K_1 \cup K_2} x_{ij} = 1 \qquad j = 1, \ldots, n \tag{16}$$

$$0 \leqslant x_{ij} \qquad i \in K_1 \cup K_2, \; j = 1, \ldots, n \tag{17}$$

En posant :

$$g_k = f_k \text{ pour } k \in K_2 \tag{18}$$

$$g_k = 0 \text{ pour } k \in K_1$$

le problème devient :

$$\sum_{i \in K_1} f_i + \min \Big(\sum_j \sum_{i \in K_1 \cup K_2} \Big[\Big(c_{ij} + \frac{g_i}{n_i} \Big) x_{ij} \Big] \Big) \tag{19}$$

$$\sum_{i \in K_1 \cup K_2} x_{ij} = 1 \qquad j = 1, \ldots, n \tag{20}$$

$$0 \leqslant x_{ij} \qquad i \in K_1 \cup K_2, \; j = 1, \ldots, n \tag{21}$$

La solution optimale est donnée par :

$$x_{ij} = 1 \quad \text{si } c_{ij} + \frac{g_i}{n_i} = \min_{k \in K_1 \cup K_2} (c_{kj} + \frac{g_k}{n_k}) \tag{22}$$

$$x_{ij} = 0 \tag{23}$$

sinon on calcule, pour le test d'optimalité, la valeur de la fonction d'évaluation et on la compare avec la valeur de la meilleure solution déjà obtenue; deux cas sont possibles :

a) la fonction d'évaluation est plus grande :
alors l'ensemble ne contient pas de solution de coût inférieur à celui de la meilleure solution déjà obtenue et il est sondé;

b) la fonction d'évaluation est plus petite :
- K_2 est vide : l'ensemble comprend un seul élément et on a obtenu une solution dont la valeur est moindre que celle de la meilleure solution dejà obtenue par l'algorithme;
- K_2 n'est pas vide : l'ensemble n'est pas sondé et on continue avec les tests suivants.

Un test d'optimalité conditionnelle permet de voir si certaines variables doivent être mises à 0 pour que le sous-ensemble de solutions candidates considéré puisse contenir la solution optimale.

C'est le cas lorsque le gain maximum qui pourrait être réalisé sur les coûts de transport en ouvrant un entrepôt est plus petit que le coût fixe d'établissement. Pour exprimer ceci mathématiquement, on définit :

$$cm_j = \min_{k \in K_1} c_{kj} \tag{24}$$

La condition s'exprime alors :

$$\sum_j \max (0, cm_j - c_{ij}) < f_i \tag{25}$$

La sommation est faite sur les valeurs positives de $cm_j - c_{ij}$ parce que seuls les gains doivent être pris en compte.

Il n'y a pas de tests d'admissibilité car les contraintes peuvent toujours être vérifiées dès qu'un y_i est égal à 1.

La règle de choix de la variable utilisée pour la séparation consiste à mettre la variable y_i à 1 si :

$$\sum_j \max (0, cm_j - c_{ij}) - f_i = \max_{k \in K_2} [\sum_j \max (0, cm_j - c_{kj}) - f_k] \tag{26}$$

On met donc à 1 la variable qui entraîne la plus grande réduction des coûts.

2.3. Six algorithmes heuristiques.

Divers auteurs [8], [10], [11] ont proposé des algorithmes heuristiques pour le modèle M1 ou d'autres modèles proches de M1. Ces algorithmes permettent d'obtenir rapidement une bonne solution, qui n'est pas nécessairement optimale.

Six algorithmes heuristiques ont été utilisés dans les expériences. Les trois algorithmes ATTILA 1, BABEL 1 et BABILA 1 travaillent par modifications successives de composantes du vecteur y, une composante à la fois. Les valeurs des variables x_{ij} se déduisent immédiatement de celles des variables y_i.

1) ATTILA 1. Initialement on suppose qu'il existe un entrepôt en chaque emplacement ($y_i = 1, \forall i$). A la k^{me} itération, on calcule pour chaque emplacement où un entrepôt reste ouvert la variation de coût qui serait obtenue en fermant cet entrepôt. On calcule le minimum de ces variations de coût. Si ce minimum est négatif on ferme l'entrepôt correspondant et on passe à l'itération suivante. Si ce minimum est positif on a atteint un optimum local et on s'arrête.

2) BABEL 1. Initialement on suppose qu'il n'existe aucun entrepôt et que le coût de transport est un nombre fini très grand. A la k^{me} itération, on calcule pour chaque emplacement où il n'y a pas d'entrepôt la variation de coût qui serait obtenue en ouvrant un entrepôt en cet emplacement. On calcule le minimum de ces variations de coût. Si ce minimum est négatif on ouvre l'entrepôt correspondant et on passe à l'itération suivante. Si ce minimum est positif ou nul on a atteint un optimum local et on s'arrête.

3) BABILA 1. Initialement on part d'une solution quelconque, avec des entrepôts en certaines emplacements ou éventuellement en tous. A la k^{me} itération, on calcule pour chaque emplacement où un entrepôt est ouvert la variation de coût qui serait obtenue en le fermant et pour chaque emplacement où il n'y a pas d'entrepôt la variation de coût qui serait obtenue en y ouvrant un entrepôt. On calcule le minimum de ces variations de coût. Si ce minimum est négatif, ou ouvre (ou on ferme) l'entrepôt correspondant et on passe à l'itération suivante. Si ce minimum est positif ou nul on a atteint un optimum local et on s'arrête.

4) ATTILA 2, BABEL 2 et BABILA 2. Ces trois algorithmes sont des versions améliorées des algorithmes ATTILA 1, BABEL 1 et BABILA 1. Après l'obtention d'une bonne solution par une de ces méthodes, on examine si le coût peut être réduit par déplacement d'un entrepôt. A cet effet, à la k^{me} itération, on calcule pour tous les couples (i_1, i_2) d'emplacements, tel qu'il y ait un entrepôt en i_1 et pas d'entrepôt en i_2, la variation de coût qui serait obtenue en déplaçant l'entrepôt de l'emplacement i_1 à l'emplacement i_2. Si une variation de coût est négative on déplace l'entrepôt correspondant et on passe à l'itération suivante. Si toutes les variations de coût sont positives ou nulles, la solution obtenue ne peut plus être améliorée par déplacement d'un entrepôt à la fois et on s'arrête.

3. Expériences.

3.1. Un modèle numérique.

Pour tester ces algorithmes, nous considérons une entreprise fictive d'ameublement. Cette entreprise possède une seule fabrique et envisage l'établissement d'un réseau d'entrepôts. Le modèle tient compte des coûts de transport de la fabrique aux entrepôts, des entrepôts aux clients et des coûts d'établissement et de fonctionnement des entrepôts.

La génération des problèmes ainsi que les résultats des deux premières séries d'expériences sont décrits plus en détail dans [9].

3.2. Première série.

Le but de cette série d'expériences est d'étudier les distributions des erreurs par rapport à la solution optimale obtenues en utilisant les algorithmes heuristiques. A cet effet 50 problèmes de même taille sont résolus par tous les algorithmes. Chaque problème est obtenu par tirage au sort de 40 des 57 villes; ces 40 villes sont considérées comme lieux de demande et comme endroits possibles pour construire des entrepôts.

Pour chaque problème et chaque algorithme, on note le temps de résolution, le coût optimal, le nombre et les emplacements des entrepôts.

3.3. Deuxième série.

Le but de cette série d'expériences est d'étudier la variation des temps de résolution sur ordinateur en fonction de la taille des problèmes. A cet effet 50 problèmes de taille variable sont résolus par l'algorithme exact

et par l'algorithme heuristique ayant donné la plus petite erreur moyenne au cours de la première série d'expériences.

Les 57 villes sont considérées comme lieux de demande. Pour chaque problème, un nombre m compris entre 5 et 50 est d'abord tiré au hasard. m villes sont ensuite tirées au hasard parmi les 57 villes, comme emplacements possibles pour établir des entrepôts.

Pour chaque problème et pour les deux algorithmes, on note le temps de résolution, le coût optimal, le nombre et les emplacements des entrepôts.

3.4. Troisième série.

Le but de cette série d'expériences est d'étudier la variation de la solution optimale lorsqu'on modifie les coûts fixes d'établissement des entrepôts; on considère des problèmes avec 57 lieux de demande et un nombre d'emplacements où l'on peut construire des entrepôts variant entre 30 et 50.

Ces problèmes sont résolus par LOCA 1 et la série est recommencée pour des coûts fixes diminuant de 10 % de leur valeur, en partant de 100 % et allant jusqu'à 60 % de celle-ci. Il y a donc pour chaque problème 5 niveaux de coûts fixes.

3.5. Conclusions.

La première série d'expériences montre que tous les algorithmes heuristiques donnent de bons résultats du point de vue des erreurs : une erreur non nulle est obtenue pour moins de 40 % des problèmes avec n'importe lequel de ces algorithmes. La moyenne des erreurs positives est toujours inférieure à 0,5 % de la valeur de la solution optimale. Les meilleurs résultats sont obtenus avec l'algorithme ATTILA 2 : une solution non optimale a été obtenue pour 4 problèmes sur 50 seulement. Les algorithmes ATTILA 1 et 2 donnent de meilleurs résultats que BABEL 1 et 2 respectivement.

Le temps de calcul par ATTILA 2 est 4 fois moindre que par LOCA 1. La dispersion des temps de calcul est faible pour tous les algorithmes heuristiques et forte pour LOCA 1.

Pour analyser les résultats de la deuxième série, on a ajusté aux résultats obtenus par les 2 algorithmes une courbe d'équation :

$$T = a\, m^b \tag{27}$$

où T est le nombre de secondes de temps de calcul sur IBM 7040 et m le nombre d'emplacements.

On trouve pour LOCA 1 :

$$T = 0{,}01\ m^{2{,}64} \tag{28}$$

et pour ATTILA 2 :

$$T = 0{,}03\ m^{1{,}97} \tag{29}$$

La troisième série d'expériences montre que les solutions obtenues pour ce type de problèmes sont peu sensibles aux variations des coefficients f_i. Pour 4 problèmes sur 11, la même solution est obtenue pour les 5 valeurs des coûts fixes et pour 6 autres problèmes sur 11 la même solution est obtenue pour 4 valeurs sur 5 des coûts fixes.

Pour 9 problèmes sur 11, la diminution des coûts fixes se traduit uniquement par l'ouverture de nouveaux entrepôts. Dans 2 cas seulement sur 11 il y a également une fermeture.

En conclusion, quand les problèmes sont de taille moyenne, jusqu'à 80 villes environ, l'algorithme LOCA 1 est efficace. Pour des problèmes plus larges, il faut employer une méthode heuristique. Dans ce cas l'algorithme ATTILA 2 donne de bons résultats.

4. Extensions.

4.1. Un modèle avec contraintes de capacité.

Le modèle M1 suppose que les entrepôts ont des capacités illimitées; cette hypothèse est valable au cas où un réseau entièrement nouveau doit être établi; au cas où certains entrepôts existent déjà (les variables y_i correspondantes sont alors fixées à 1), il est nécessaire de tenir compte des contraintes de capacité; de même, lorsqu'on étudie un problème de localisation d'usines, il faut tenir compte des limites des capacités de production des usines existantes. Pour ce problème, la généralisation suivante du modèle de Balinski a été proposée par G. SA [18] :

$$\min \left(\sum_{i \in M} \sum_{j} c_{ij}\ x_{ij} + \sum f_i\ y_i \right) \tag{30}$$

$$\sum_{i} x_{ij} = d_j \qquad j = 1, ..., n \tag{31}$$

$$\sum_{j} x_{ij} \leqslant y_i\ a_i \qquad i = 1, ..., m \tag{32}$$

$$y_i \in \{0, 1\} \qquad i = 1, ..., m \tag{33}$$

$$0 \leqslant x_{ij} \qquad i = 1, ..., m,\ j = 1, ..., n \tag{34}$$

où x_{ij} est la quantité de bien allant de i à j,
c_{ij} le coût unitaire de transport de i à j,
a_i la capacité de l'entrepôt projeté en i,
d_j la demande en j; les f_i et y_i s'interprètent comme dans le modèle M1.

Sâ a proposé un algorithme semblable à l'algorithme d'Efroymson et Ray mais moins efficace.

P.S. DAVIS et T.L. RAY [5] ont proposé un autre algorithme utilisant des évaluations très précises de la valeur de la fonction économique; ces évaluations sont calculées en résolvant un problème de transport généralisé, par une méthode de décomposition utilisant récursivement l'algorithme « out of kilter » de Fulkerson. La solution optimale est souvent obtenue sans séparation pour des problèmes pratiques; le calcul de l'évaluation étant long, l'algorithme ne peut être utilisé que pour des problèmes de taille petite ou moyenne.

D.H. MARKS [13], [14] ainsi que B. ROY et R. BENAYOUN [17] ont proposé des procédures d'optimisation par séparation sur un graphe qui peuvent également être utilisées pour ce problème.

4.2. Un modèle pour la localisation de centres de service.

On cherche à maximiser une fonction de l'utilité des usagers avec un budget limité.

Le modèle suivant, dû à C. REVELLE et R.W. SWAIN [15], permet de localiser p centres de services de manière à minimiser la somme des trajets des usagers :

$$\min \sum_i \sum_j a_{ij}\, x_{ij} \tag{35}$$

$$\sum_i x_{ij} = 1 \qquad j = 1, \ldots, n \tag{36}$$

$$\sum_j x_{ij} \leqslant n_i\, y_i \qquad i = 1, \ldots, m \tag{37}$$

$$\sum_i y_i = \mathrm{P} \tag{38}$$

$$y_i \in \{0, 1\} \qquad i = 1, \ldots, m \tag{39}$$

$$0 \leqslant x_{ij} \qquad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{40}$$

où a_{ij} est la distance entre i et j,

P le nombre d'entrepôts à ouvrir; les x_{ij}, les n_i et les y_i s'interprètent comme dans le modèle M1.

La contrainte (38) est due au budget limité.

La solution obtenue par programmation linéaire est souvent en variables entières; si ce n'est pas le cas, on utilise une procédure d'optimisation par séparation en choisissant une variable fractionnaire et en la fixant d'une part à 0 et d'autre part à 1.

Le nombre de variables ($n \times m$) limite en pratique la taille des proplèmes qu'on peut résoudre par cette méthode.

REFERENCES

[1] ATKINS, R.J. and SHRIVER, R.H. (1968) : « New Approach to Facilities Location », *Harvard Business Review* 46, 70-79.

[2] BALINSKI, M.L. (1961) : « Fixed-cost Transportation Problems », *Naval Research Logistics Quarterly,* 8, 41-54.

[3] BALINSKI, M.L. (1964) : « On Finding Integer Solutions to Linear Programs », *Mathematica-Princeton,* N.J.

[4] BAUMOL, W.J. and KUHN, M.W. (1962) : « An approximate Algorithm for the Fixed-Charges Transportation Problem », *Naval Research Logistics Quarterly,* 9, 1-15.

[5] DAVIS, P.S. and RAY, T.I. (1969) : « A Brand-Bound Algorithm for the capacitated Facilities Location Problem ». *Naval Research Logistics Quarterly* 16, 331-344.

[6] DESCAMPS, R. (1966) : « Un algorithme d'optimisation pour une classe de problèmes d'implantation » dans HERTZ, D.B. et MELESE, J., Editeurs, *Actes de la Quatrième Conférence internationale de Recherche Opérationnelle,* Wiley-Interscience, New York, 834-842.

[7] EFROYMSON, M.A. and RAY, T.L. (1966) : « A Branch and Bound Algorithm for Plant Location », *Operations Research,* 14, 361-368.

[8] FELDMAN, E., LEHRER, F.A. and RAY, T.L. (1966) : « Warehouse Location under continuous Economies of Scale ». *Management Science,* 12, 670-684.

[9] HANSEN, P. et KAUFMAN, L. (1970) : « Comparaison d'algorithmes pour le problème de la localisation des entrepôts », à paraître dans J. BRENNAN, Editor, « *Operations Research in Industrial Systems* », English University Press (1971).

[10] HENRY, G. (1968) : « Recherche d'un réseau de dépôts optimum », *Revue Française d'Informatique et de Recherche Opérationnelle,* 2, 61-70.

[11] KUEHN, A.A. and HAMBURGER, M.J. (1963) : « A Heuristic Program for Locating Warehouses ». *Management Science,* 10, 643-666.

[12] MANNE, A.S. (1964) : « Plant Location under Economies of Scale Decentralisation and Computation », *Management Science* 11, 213-235.

[13] MARKS, D.H. (1969) : « Facility Location and Routing models in Solid Waste Collection Systems », Ph. D. Thesis, The Johns Hopkins University.

[14] REVELLE, C., MARKS, D. and LIEBMAN, J.C. (1970) : « An Analysis of Private and Public Sector Location Models », *Management Science* 16, 692-707.

[15] REVELLE, C. and SWAIN, R.W. (1970) : « Central Facilities Location », *Geographical Analysis,* 2, 30-42.

[16] ROY, B. (1969) : « Procédures d'exploration par séparation et évaluation (P.S.E.P. et P.S.E.S.), *Revue Française d'Informatique et de Recherche Opérationnelle,* 3, 61-90.

[17] ROY, B. et BENAYOUN, R. (1967) : « Programmes linéaires en variables bivalentes et continues sur un graphe (le programme POLIGAMI) », *METRA,* 6, 675-710.

[18] SA, G. (1969) : « Branch and Bound and Approximate Solutions to the Capacitated Plant Location Problem », *Operations Research,* 17, 1005-1016.

[19] SPIELBERG, K. (1969) : « Algorithms for the Simple Plant-Location Problem with some Side Conditions », *Operations Research,* 17, 85-111.

[20] SPIELBERG, K. (1969) : « Plant Location with Generalized Search Origin », *Management Science,* 16, 165-178.

[21] STURM, L.B.J.M. (1969) : « On the Warehouse Location Problem », Communication at the « *1969 European Econometric Society Meeting* », Brussels.

[22] ZIMMERMAN, R. (1967) : « A Branch and Bound Algorithm for Depot Location », *METRA* 4, 661-674.

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

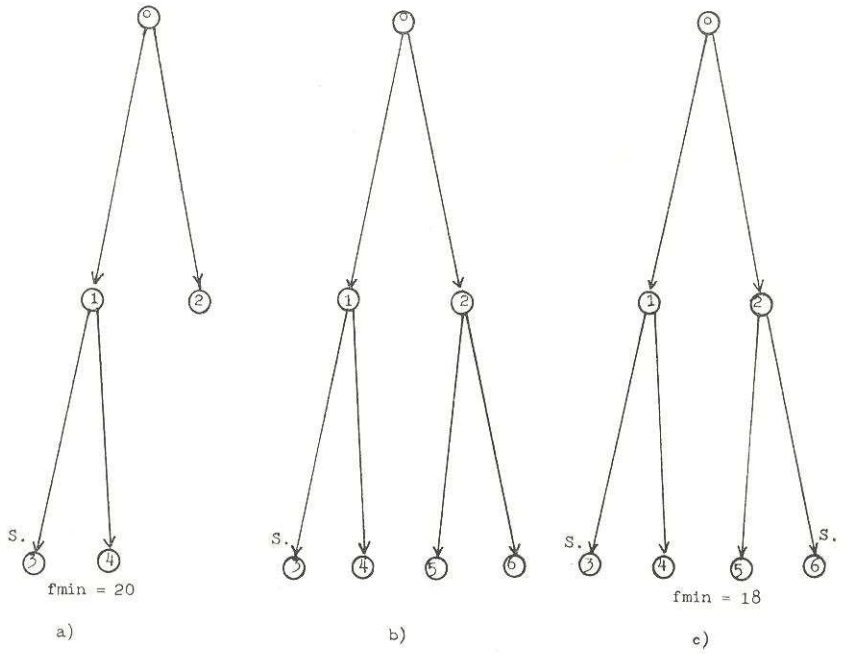


Fig. 1.

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.

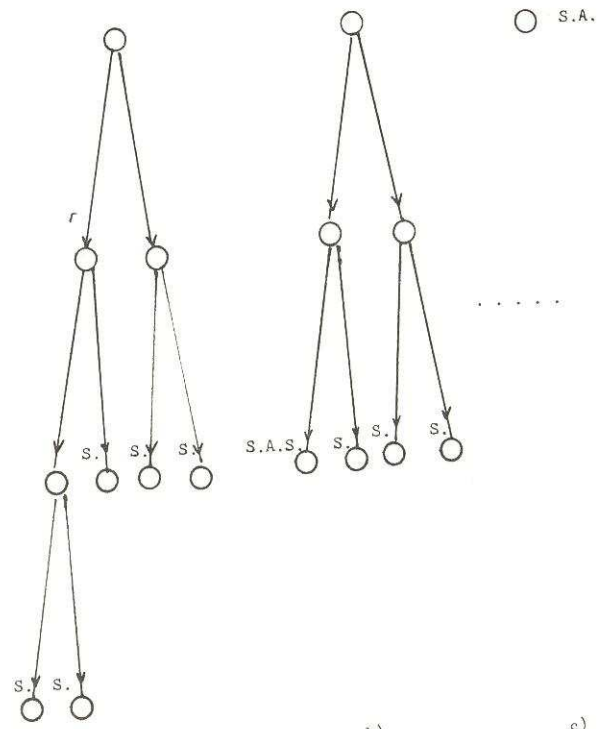


Fig. 2.

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 :

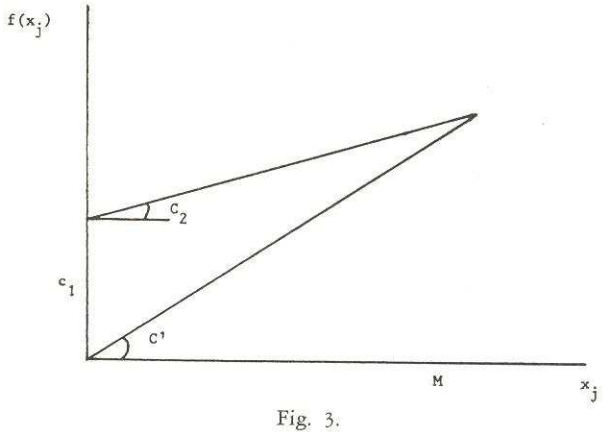


Fig. 3.

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

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

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

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

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).

Remarque concernant l'article

SUR UN MODELE DE GESTION DE STOCK SUR SEUIL AVEC REVISION CYCLIQUE OU CONTINUE ET DELAI DE LIVRAISON

paru dans le vol. 11 n° 2 de la Revue.

M. Roubens

En page 19, nous pouvons lire le résultat suivant :

$$SR = SS + \frac{q}{2} - \left[\frac{\hat{Y}(T+L) - \hat{Y}(L)}{2}\right] \qquad (1)$$

SR y représente le stock net moyen en révision *cyclique* et SS le stock de sécurité en révison *continue*. Il ne faut cependant pas en déduire que les stocks de sécurité en révision cyclique et continue sont identiques.

Afin de lever cette dangereuse ambiguïté, nous proposons le tableau suivant (cas des clients *patients*) :

	Révision continue	Révision cyclique
Stock de sécurité	$z - \hat{Y}(L)$	$z + \frac{q}{2} - \hat{Y}(T+L)$
Stock net moyen	$z + \frac{q}{2} - \hat{Y}(L)$	$z + \frac{q}{2} - \left\{\frac{\hat{Y}(T+L) + \hat{Y}(L)}{2}\right\}$

Dès lors, la formule (1) peut s'écrire :

$$SR = SS + \left[\frac{\hat{Y}(T+L) - \hat{Y}(L)}{2}\right]$$

où SR et SS représentent, cette fois, le stock net moyen et le stock de sécurité en révision *cyclique*.

COMMUNICATIONS MEDEDELINGEN

INTERNATIONAL ASSOCIATION OF SURVEY STATISTICIANS

The International Statistical Institute at its 38-th Session (August 1971) approved the formation of a new Association, the International Association of Survey Statisticians. The Association will be affiliated with the Institute and function as one of its sections; membership in the Association, however, is *not* restricted to members of the ISI.

The objectives of the Association are to promote the study and development of the theory and practice of statistical censuses and surveys and associated subjects, and to foster interest in these subjects. These objectives may be furthered through the organization of meetings, seminars, conferences, research or training programmes, publications, etc.

An organizing committee has prepared the draft Statutes of the Association. This committee was under the Chairmanship of I.P. Fellegi (Canada) and included as its members Messrs. J.P.M.R. Desabie (France), L. Kish (U.S.A.), M.N. Murthy (India), M.R. Sampford (U.K.), and Z.Z. Zarkovich (Yugoslavia). The draft statutes will be voted upon by the charter members of the new Association some time during 1972.

In order to become charter members, all persons with an active interest in furthering the objectives of the Association are requested to send their name and address to Mr. E. Lunenberg, Director of the Permanent Office, International Statistical Institute, 2 Oostduinlaan, The Hague by 31 March 1972. After this date the draft statutes and a slate of officers nominated by the organizing committee will be sent for voting to all persons on Mr. Lunenberg's list. All these persons will become charter members by returning their ballot sheets plus their annual membership fee ($ 5,– for the first year).

The first formal activity of the new Association will be to organize some sessions as part of the regular programme of the next meeting of the International Statistical Institute (Vienna, August 1973) and some additional sessions directly preceding, succeeding or concurrent with the regular ISI programme. Persons who have suggestions for topics to be discussed by the new Association during its first meeting, or who would like to present a contributed paper should contact the Chairman of the organizing committee (I.P. Fellegi, Statistics Canada, Ottawa, Canada).

THE MATHEMATICAL PROGRAMMING SOCIETY

The Organizing Committee for the Mathematical Programming Society was formed during the 7th Mathematical Programming Symposium, held at The Hague on September 14-18, 1970. The members of this Committee are: A. Orden (Chairman), J. Abadie, M.L. Balinski (as Editor-in-Chief of the Journal), E.M.L. Beale, A. Charnes, G.B. Dantzig, R.E. Gomory, A.S. Manne, W. Orchard-Hays, M.J.D. Powell, A. Prekopa, A.W. Tucker, S. Vajda, P. Wolfe, and G. Zoutendijk.

The purpose of the Society is the advancement of mathematics, algorithms, applications, and methods of computation in the field which has come to be known as mathematical programming. The scope of this field has been identified in the course of the seven major symposia which have occurred at intervals of about three years since 1951.

In addition to the development of mathematics and algorithms for linear, nonlinear, and integer programming, it is concerned with computer programming and experimentation on algorithms, with applicational models which involve new interpretations of the mathematical structures and processes, and with allied mathematical topics such as unconstrained optimization and graph theory.

The Society will continue the international symposia — heretofore held under ad hoc sponsorship of diverse organizations —, and in cooperation with the North-Holland Publishing Company, will publish the new journal, MATHEMATICAL PROGRAMMING. Individual subscriptions to the Journal are to be handled by the Society, and institutional subscriptions by North-Holland.

Further information can be obtained from the provisional Secretariat of the Society: c/o the International Statistical Institute, 2 Oostduinlaan, The Hague - Netherlands.

PUBLICATIONS REÇUES — ONTVANGEN PUBLICATIES

1) Q.F. (Qualité et Fiabilité), Bulletin de l'AFCIQ, Paris, Vol. VII, nos 2 et 3, 1971.
2) Etudes et Documents du CNBOS, nos364 à 368 inclus.
3) Revue française d'Informatique et de Recherche opérationnelle, no R-I, 1971.
4) Journal of Operations Research Society of Japan, Vol. 13, nos 3 et 4, Vol. 14, no 1.
5) Key to Turkish Science,Applied Economics, December 1970, Vol. 2, no 1-2 (Turkish Scientific and Technical Documentation Centre).
6) Annales de Sciences économiques appliquées (U.C.Lv.), nos 2 et 3, 1971.
7) Trabajos de Estadistica y de Investigacion Operativa, Volumen XXII, Cuadernos 1 y 2, Madrid.
8) Mathematical Centre Tracts - Preservation of Infinite Divisibility under Mixing and Related Topics (F.W. Steutel), Mathematical Centre, Amsterdam).
9) Notas de Logica Matematica - Aspectos de la Logica Model, Instituto de Matematica, Universidad Nacional del Sur, Bahia Blanca, Argentina.
10) Systèmes d'Informatique Magazine, Honeywell-Bull, Printemps 1971, Eté 1971.
11) Giornale degli Economisti e Annali di Economia, Universita Commerciale Luigi Bocconi, nos 3-4 et 5-6, 1971, Milan.
12) BIN Revue IBN (Institut belge de Normalisation — Belgisch Instituut voor Normalisatie), nos/nrs 4 à 9, 1971.
13) Organisation scientifique (CNBOS), nos 5 à 9, 1971.
14) Paninformatic, Bruxelles, nos 7 à 9, 1971.
15) Industrielle Organisation - Institut d'Organisation industrielle de l'Ecole polytechnique fédérale de Zurich, nos 6 à 10, 1971.
16) Bulletin économique pour l'Europe (Nations Unies), Vol. 22, no 1.
17) Etude sur la situation économique pour l'Europe en 1969. Première partie - Tendances et perspectives structurelles de l'économie européenne - Nations Unies, Genève.
18) AGIFORS Proceedings - Tenth AGIFORS Symposium November 1970, Terrigal, Australia.
19) Operational Research Quaterly, Vol. 22, Special Conference Issue - 1970 Conference et Vol. 22, no 3, septembre 1971.
20) Revue de Statistique appliquée, Vol. 19, no 3, 1971.

Mathematical Programming 1 (1971) 123–125. North-Holland Publishing Company

THE MATHEMATICAL PROGRAMMING SOCIETY

The Organizing Committee for the Mathematical Programming Society was formed during the 7th Mathematical Programming Symposium, held at the Hague on September 14-18, 1970. The Committee* now invites all who are active in the field of mathematical programming to become Charter Members of the Society.

The purpose of the Society is the advancement of mathematics, algorithms, applications, and methods of computation in the field which has come to be known as *mathematical programming*. The scope of this field has been identified in the course of the seven major symposia which have occurred at intervals of about three years since 1951. In addition to the development of mathematics and algorithms for linear, nonlinear, and integer programming, it is concerned with computer programming and experimentation on algorithms with applicational models which involve new interpretations of the mathematical structures and processes, and with allied mathematical topics such as unconstrained optimization and graph theory.

The Society will continue the international symposia – heretofore held under ad hoc sponsorship of diverse organizations – and in cooperation with the North-Holland Publishing Company, will publish the new journal, MATHEMATICAL PROGRAMMING. Individual subscriptions to the Journal are to be handled by the Society, and institutional subscriptions by North-Holland.

Volume 1 of MATHEMATICAL PROGRAMMING will appear in three issues during the remainder of 1971. It will consist primarily of papers which were presented at the 7th Symposium. It is planned that, beginning in 1972, the Journal will appear six times per year (two volumes of three issues each).

* The members of the Organizing Committee are: A. Orden, Chairman, J. Abadie, M.L. Balinski (Editor-in-Chief of the Journal), E.M.L. Beale, A. Charnes, G.B. Dantzig, R.E. Gomory, A.S. Manne, W. Orchard-Hays, M.J.D. Powell, A. Prekopa, A.W. Tucker, S. Vajda, P. Wolfe and G. Zoutendijk.

Membership in the Society

Individuals who enroll as members of the Society by December 31, 1971 will be classed as Charter Members. They will receive the Journal beginning with the first issue of Volume 1, and will participate in the election of the first Council. They will be asked to express their views on the structure and scope of the Society, as expressed in a draft Constitution, and will vote on ratification of the Constitution.

A membership form is attached. The fee for individual membership is $22 (or equivalent in other currencies) per calendar year.

The 7th Symposium registration fee included $10 for printed Proceedings. Regardless of whether they join the Society, the participants in the Symposium will – in lieu of separate Proceedings – receive Volume 1 of the Journal. The membership registration form calls for an indication of whether they wish to enroll in the Society. Those who do not are kindly requested to complete the form for administrative purposes.

Non-participants in the 7th Symposium should include $10 (or equivalent), for which they will receive Volume 1 of the Journal.

Everyone who enrolls as a Member is requested to pay the dues for 1972 ($22 or equivalent) not later than January 1972, to enable the Society to meet its obligations towards the Publisher of the Journal.

The Secretariat

The membership form and membership fee should be sent to the

The Mathematical Programming Society
c/o International Statistical Institute
2, Oostduinlaan
The Hague – Netherlands

The I.S.I. will conduct the enrollment of Charter members of the Society, and provisionally will establish a continuing Secretariat for the Society.

THE MATHEMATICAL PROGRAMMING SOCIETY

Membership Registration Form
(For use until Dec. 31, 1971)

Mail to: Mathematical Programming Society
c/o International Statistical Institute
2, Oostduinlaan
The Hague – Netherlands

A. For those who did not attend the 7th Mathematical Programming Symposium

☐ I wish to enroll as a Charter Member of the Society.
The membership fee for 1971 ($10 or equivalent) is enclosed. This covers membership until Dec. 31, 1971, and entitles me to volume 1 of the journal.
I will pay the membership fee for 1972 ($22 or equivalent) – which entitles me to volumes 2 and 3 of the journal – not later than January 1972.

B. For those who attended the 7th Mathematical Programming Symposium.

☐ I wish to enroll as a Charter Member of the Society.
Since payment of the Symposium fee has entitled me to volume 1 of the journal, I do not need to pay an additional membership fee for 1971.
I will pay the membership fee for 1972 ($22 or equivalent) – which entitles me to volumes 2 and 3 of the journal – not later than January 1972.

☐ I do not wish to enroll as a member of the Society.
Since I have paid the registration fee for the 7th Symposium, I understand that I will receive volume 1 of the journal as Proceedings of the Symposium.

By registration as a member of the Society I subscribe to the journal for my personal use and not for the benefit of any library or other institution.

Signature: Name:

Mailing address: Address:

Note:

Please send a cheque or money-order, made payable to the Mathematical Programming Society, in one of the following currencies:

$10 or £ 4-25 or Sw.Fr. 42.50 or FF 60 or DM 35 or 35 Dutch Guilders
$22 or £ 9-35 or Sw.Fr. 93.50 or FF 132 or DM 77 or 77 Dutch Guilders

Prix de vente

Au numéro :	Belgique	75 FB
	Etranger	90 FB
Abonnement :	Belgique	250 FB
(4 numéros)	Etranger	300 FB

Tarif de publicité

(4 numéros)

La page	: 5.000 F
La 1/2 page	: 3.000 F
Le 1/4 page	: 2.000 F

Les frais de clichés sont à charge de l'annonceur.

Publications d'articles

1) La Revue est ouverte aux articles traitant de statistique pure et appliquée, de recherche opérationnelle et de « quality control ».
2) Les manuscrits seront dactylographiés et peuvent être envoyés au secrétariat de la Revue : 66, rue de Neufchâtel, Bruxelles 6.
3) Les auteurs d'articles techniques recevront 25 tirés à part de leurs textes.
4) La responsabilité des articles n'incombe qu'à leurs auteurs.

Verkoopprijs

Per nummer :	België	75 BF
	Buitenland	90 BF
Abonnement :	België	250 BF
(4 nummers)	Buitenland	300 BF

Advertentietarief

(4 nummers)

Per bladzijde	: 5.000 F
Per 1/2 bladzijde	: 3.000 F
Per 1/4 bladzijde	: 2.000 F

De cliché-onkosten vallen ten laste van de adverteerders.

Publicaties van artikels

1) Het Tijdschrift neemt artikels aan over wiskundige statistiek en toepassingen, over operationeel onderzoek en kwaliteitszorg.
2) De teksten dienen getipt gestuurd te worden naar het secretariaat van het Tijdschrift : 66, Neufchâtelstraat, Brussel 6.
3) De auteurs ontvangen 25 overdrukken van de technische artikels.
4) De auteurs zijn alleen verantwoordelijk voor de inhoud van hun teksten.