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

KHOAN VO-KHAC

Utilisation des goûts pondérés et des variables bivalentes dans le problème des tournées : sectorisation sous contraintes nombreuses

Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle, tome 6, n° V2 (1972), p. 3-19

<http://www.numdam.org/item?id=RO_1972__6_2_3_0>

R.A.I.R.O.
(6e année, octobre 1972, V-2, p. 3-20)

UTILISATION DES COUTS PONDERES ET DES VARIABLES BIVALENTES DANS LE PROBLEME DES TOURNEES : SECTORISATION SOUS CONTRAINTES NOMBREUSES

par Khoan Vo-Khac [1]

Résumé. — *Une nouvelle méthode additive de résolution de programmes à variables bivalentes et à contraintes booléennes est proposée. Cette méthode est ensuite utilisée à l'étude du problème des tournées, après avoir créé des contraintes artificielles à l'aide des coûts pondérés.*

La présente note s'inspire d'un article de Balinski et Quandt [2]. Dans cet article, d'une part on considère l'ensemble de *toutes* les tournées admissibles; d'autre part, l'algorithme utilisé est celui du « cutting-plane » de Gomory. Les auteurs ont fait des essais numériques sur des problèmes comportant moins de 15 clients. La présente note essaie d'améliorer cet article en le rendant utilisable pour un grand nombre de clients. Deux idées nouvelles sont introduites : la *réduction* des tournées admissibles à l'aide des *coûts pondérés* et un nouvel *algorithme additif* pour la résolution des programmes à variables bivalentes. Plusieurs exemples numériques sont donnés. Quelques réalisations pratiques ont été effectuées à la CEGOS.

PARTIE A

UN PROGRAMME A VARIABLES BIVALENTES

Il existe de nombreuses méthodes de résolution de programmes à variables bivalentes : l'algorithme additif de Balas [1], la méthode du « cutting-plane », basée sur la programmation linéaire, de Gilmore et Gomory [3], la méthode

(1) Université d'Orléans et École Polytechnique, Paris.

de l'ordre lexicographique de Lawler et Bell [4], la méthode de Shapiro [6], basée sur la théorie des classes résiduelles modulo D, une méthode de « séparation et évaluation progressive » de la S.E.M.A. [5]...

Pour notre problème très particulier (programme à *variables et à contraintes booléennes*, avec des coûts positifs), nous proposons une méthode de résolution très simple et aussi *additive* comme celle de Balas.

1. Le problème posé

1.1. Données du problème

On désigne par $\mathbf{R}_+$ la demi-droite réelle positive ($\geqslant 0$), par $\mathbf{B}$ l'ensemble $\{0,1\}$ et par I et J deux ensembles finis.

On se donne un élément $\{ c(j),\ j \in J \}$ de $\mathbf{R}_+^J$ (appelé vecteur des coûts) et un élément $\{ a(i,j);\ i \in I,\ j \in J \}$ de $\mathbf{B}^{I \times J}$ (appelé matrice des contraintes).

Pour tout $x \in \mathbf{B}^J$, on pose :

$$\psi(x) = \sum_{j \in J} c(j)x(j)$$

$$\gamma_i(x) = \sum_{j \in J} a(i,j)x(j), \qquad i \in I$$

Puis, on pose :

$$\Gamma(i) = \{ x \in \mathbf{B}^J ;\ \gamma_i(x) \geqslant 1 \} \text{ (resp. } \gamma_i(x) = 1 \})$$

$$\Gamma = \bigcap_{i \in I} \Gamma(i)$$

On supposera que l'ensemble Γ ne soit pas vide.

1.2. Position du problème

On cherche un élément $\hat{x} \in \Gamma$ tel que $\psi(\hat{x}) \leqslant \psi(x)$ pour tout $x \in \Gamma$.

Ce problème s'interprète, au point de vue théorique, aisément comme un problème d'affectation concernant les graphes bipartis.

Au point de vue pratique, ce problème trouve son application dans l'étude des tournées (cas où les contraintes sont des égalités) et dans l'étude de l'ordonnancement des pilotages (cas où les contraintes sont des inégalités).

2. Quelques propriétés

L'algorithme de résolution du problème posé utilise la proposition suivante et son corollaire. Pour tout $i \in I$, on pose $J(i) = \{ j \in J ;\ a(i,j) \neq 0 \}$.

2.1. Proposition

Soient I' et I'' deux parties de I telles que $J' = \bigcup_{i \in I'} J(i)$ et $J'' = \bigcup_{i \in I''} J(i)$ soient *disjoints*. Alors, on a l'inégalité suivante :

$$\min_{x \in \Gamma} \psi(x) \geqslant \min_{x \in \Gamma'} \psi(x) + \min_{x \in \Gamma''} \psi(x)$$

où

$$\Gamma' = \bigcap_{i \in I'} \Gamma(i) \text{ et } \Gamma'' = \bigcap_{i \in I''} \Gamma(i)$$

Notons que cette proposition utilise la *positivité* des nombres $c(j)$ et $x(j)$, mais n'utilise pas la bivalence des nombres $x(j)$ et $a(i, j)$.

PREUVE

Soit x un élément de Γ. Déterminons x' et x'', éléments de $\mathbf{B}^J$, par :

$$x'(j) = x(j) \text{ si } j \in J', = 0 \text{ si } j \notin J'$$

$$x''(j) = x(j) \text{ si } j \in J'', = 0 \text{ si } j \notin J''.$$

Comme les $c(j)$ sont des nombres *positifs* ($\geqslant 0$) et comme J' et J'' sont disjoints, on a :

$$\psi(x) = \sum_{j \in J} c(j)x(j) \geqslant \sum_{j \in J'} c(j)x(j) + \sum_{j \in J''} c(j)x(j) = \psi(x') + \psi(x'')$$

D'autre part, pour tout $i \in I'$, on a :

$$\gamma_i(x') = \sum_{j \in J'} a(i, j)x(j) = \sum_{j \in J(i)} a(i, j)x(j) = \sum_{j \in J} a(i, j)x(j)$$

$$= \gamma_i(x) \geqslant 1 \text{ (resp. } \gamma_i(x) = 1)$$

ce qui montre que x' est élément de Γ'. De même on démontre que x'' est élément de Γ''. Par suite

$$\psi(x) \geqslant \min_{x \in \Gamma'} \psi(x) + \min_{x \in \Gamma''} \psi(x).$$

2.2. Corollaire

Pour tout $i_0 \in I$, on pose :

$$D(i_0) = \{ i \in I ; J(i) \cap J(i_0) = \varnothing \}, \bar{\Gamma}(i_0) = \bigcap_{i \in D(i_0)} \Gamma(i)$$

Alors, on a :

$$\min_{x \in \Gamma} \psi(x) \geqslant \min_{J \in J(i_0)} c(j) + \min_{x \in \bar{\Gamma}(i_0)} \psi(x)$$

L'ensemble $D(i_0)$ sera appelé l'ensemble des contraintes *indépendantes* de la contrainte i_0. L'élément $j(i_0)$ tel que $c(j(i_0)) = \min_{j \in J(i_0)} c(j)$ sera appelé l'*indice minimal* de l'indice i_0.

PREUVE

Comme $J(i_0)$ et $\bigcap_{i \in D(i_0)} J(i)$ sont disjoints, la proposition précédente donne :

$$\min_{x \in \Gamma} \psi(x) \geqslant \min_{x \in \Gamma(i_0)} \psi(x) + \min_{x \in \overline{\Gamma}(i_0)} \psi(x)$$

Or, pour tout $x \in \Gamma(i_0)$, on a :

$$\sum_{j \in J(i_0)} a(i,j)x(j) \geqslant 1$$

La *positivité* des $a(i,j)$ montre que $a(i_0, j)$ est strictement positif si $j \in J(i_0)$. La nature *booléenne* des variables $x(j)$ montre alors que l'une des variables $x(j)$, $j \in J(i_0)$, doit être égale à 1. Par conséquent, il existe $\hat{j} \in J(i_0)$ tel que $\psi(x) \geqslant c(\hat{j})$ pour tout $x \in \Gamma(i_0)$, ce qui implique que :

$$\min_{x \in \Gamma(i_0)} \psi(x) \geqslant \min_{j \in J(i_0)} c(j)$$

Le corollaire en résulte.

REMARQUE

Le corollaire précédent utilise la positivité des nombres $c(j)$ et $a(i,j)$ ainsi que la bivalence des nombres $x(j)$, mais n'utilise pas la bivalence des nombres $a(i,j)$.

3. Résolution

La méthode de résolution emploie le principe de « ramification et contrôle » (branch and bound). En éliminant les variables $x(j)$ connues (égales à 0, ou égales à 1) et les contraintes superflues, on peut toujours se ramener à la première étape.

On suppose *dans tout ce qui suit que l'ensemble J est ordonné suivant l'ordre croissant des coûts ; autrement dit* : $j \geqslant j' \Leftrightarrow c(j) \geqslant c(j')$.

3.1. Algorithme théorique

Il s'agit de savoir comment on fait la ramification et comment on calcule l'évaluation par défaut.

3.1.1. Ramification

On utilise ici une biramification qui consiste à rendre une variable $x(j_0)$ égale à 0 ou à 1. Le choix de l'indice j_0 est *théoriquement* arbitraire.

3.1.2. Contrôle

L'évaluation par défaut utilise le corollaire 2.2. En éliminant les variables connues et les contraintes satisfaites, on peut toujours se ramener à l'évaluation par défaut à la racine de l'arborescence.

a) Soit i_0 un indice quelconque de I; *théoriquement* le choix de i_0 est arbitraire; soit $j(i_0)$ *l'indice minimal* de l'indice i_0, c'est-à-dire $j(i_0) = \min J(i_0)$; retenir le coût $c(j(i_0))$;

b) Supprimer toutes les contraintes dépendantes de i_0; autrement dit, ne retenir que l'ensemble $D(i_0)$ des contraintes indépendantes de i_0.

c) Recommencer pour l'ensemble des contraintes restantes (remplacer l'ensemble I par l'ensemble $D(i_0)$) et ainsi de suite.

d) S'arrêter lorsqu'il n'y a plus de contraintes.

e) La somme des coûts retenus est une évaluation par défaut de $\min \{\psi(x);\ x \in \Gamma\}$.

Les calculs se font *additivement* et très rapidement.

REMARQUE

L'algorithme précédent est valable pourvu que les nombre $a(i, j)$ soient positifs ($\geqslant 0$). Cependant *l'évaluation par défaut n'est bonne que si les nombres* $a(i, j)$ *sont booléens*. D'autre part, la nature booléenne des $a(i, j)$ montre que si $x(j_0) = 1$, alors pour tout j appartenant à $\bigcap_{i \in I(j_0)} J(i)$ et différent de j_0, on a $x(j) \geqslant 0$ (et même $x(j) = 0$ lorsque les contraintes sont des égalités).

Par conséquent, si $x(j_0) = 1$, alors *on peut éliminer tous les indices* $i \in I(j_0)$ (c'est-à-dire les contraintes « liées » à j_0) et, *dans le cas où les contraintes sont des égalités, tous les indices* $j \in \bigcap_{i \in I(j_0)} J(i)$ (indices «liés» à j_0 par des contraintes).

En résumé, l'algorithme précédent est mauvais si les nombres $a(i, j)$ ne sont pas booléens, est assez bon si les nombres $a(i, j)$ sont booléens et est bon si de plus les contraintes sont des égalités.

3.2. Pratique de l'algorithme

3.2.1. Ramification

Un choix judicieux de l'indice j_0 pour faire la dichotomie peut accélérer l'algorithme. Nous proposons deux méthodes :

a) *Première méthode*

(i) Pour tout $j \in J$, calculer le nombre

$$s(j) = \text{card } I(j) = \sum_{i \in I} a(i,j)$$

où $I(j) = \{i \in I;\ a(i,j) = \neq 0\}$.

(ii) Choisir le *plus petit* j qui rend *maximale* la fonction s (ne pas oublier que l'on a ordonné J suivant l'ordre croissant des coûts).

b) *Deuxième méthode*

(i) Pour tout $j \in J$, calculer le nombre

$$\sigma(j) = c(j) + \Sigma\, c(j')\,, j' \notin \bigcup_{i \in I(j)} J(i)$$

(ii) Choisir le *plus petit* j pour lequel la fonction σ est *minimale.*

La deuxième méthode est meilleure mais plus compliquée que la première; dans les exemples numériques, nous utiliserons la première méthode.

3.2.2. Contrôle

Un choix judicieux de i_0 permet de rendre le contrôle plus efficace (donc d'accélérer l'algorithme). Nous proposons deux méthodes.

a) *Première méthode*

(i) Pour tout $i \in I$, calculer le nombre

$$d(i) = \text{card } D(i), \quad \text{où } D(i) = \{i' \in I\,;\, J(i') \cap J(i) = \varnothing\}$$

Ce nombre sera appelé nombre de contraintes indépendantes de la contrainte i.

(ii) Choisir l'indice i_0 qui rend maximale la fonction d. S'il y a plusieurs, choisir l'indice i_0 qui possède le plus grand indice minimal $j(i_0)$. S'il y en a encore plusieurs, choisir arbitrairement.

b) *Deuxième méthode*

Comme dans la première méthode, mais au lieu de la fonction d, utiliser la fonction δ définie par :

$$\delta(i) = c(j(i)) + \sum_{i' \in D(i)} c(j(i'))$$

La deuxième méthode est meilleure mais plus compliquée que la première; dans les exemples numériques nous utilisons la première méthode.

4. Exemples numériques

Dans les trois premiers exemples, les contraintes sont des égalités; dans l'exemple 4.4, les contraintes sont des inégalités.

4.1. Premier exemple

4.1.1. Le problème

Minimiser

$$212x_{10} + 236x_{12} + 252x_{14} + 272x_{16} + 712x_{17} + 736x_{18} + 760x_{19} + 780x_{20} + 780x_{21},$$

où les x_j sont des variables booléennes satisfaisant aux contraintes suivantes :

(2, 10, 12, 14, 16)
(2, 10, 12, 16, 21)
(2, 10, 14, 16, 17)
(2, 17, 18, 19, 21)
(10, 12, 14, 19, 20)
(12, 14, 16, 18, 20)
(17, 18, 19, 20, 21)

La dernière ligne, par exemple, signifie :

$$x_{17} + x_{18} + x_{19} + x_{20} + x_{21} = 1$$

4.1.2. Solution

On utilise la méthode indiquée. L'arborescence est représentée ci-dessous (fig. 1). Le chiffre 2 dans le rond veut dire $x_2 = 1$; le chiffre $\bar{2}$ dans le rond veut dire $x_2 = 0$. On a choisi l'indice 2, car 2 est le plus petit indice j qui rend maximale la fonction s. Les chiffres se trouvant à côté des ronds désignent des évaluations par défaut. L'évaluation à la racine de l'arborescence est inutile, elle est donnée à titre indicatif. Aussi à titre indicatif, on explique ici comment on a calculé ces évaluations;

à la base : $c_{17} + c_2 = 712$
au sommet 2 : $c_2 + c_{20} = 780$
au sommet $\bar{2}$: $c_{17} + c_{10} = 924$

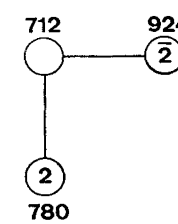

Figure 1

La solution optimale est donc : $x_2 = x_{20} = 1$.

4.2. Deuxième exemple

4.2.1. Le problème

Minimiser

$$30x_1 + 49x_2 + 51x_3 + 52x_4 + 52x_5 + 55x_6 + 67x_7 + 67x_8 + 70x_9 + 81x_{10} + 81x_{11} + 83x_{12},$$

sous les contraintes (1, 2, 3, 4, 8, 9), (1, 2, 3, 6, 7, 10), (1, 4, 5, 6, 8, 11), (2, 5, 7, 8, 9, 12), (3, 4, 9, 10, 11, 12), (5, 6, 7, 10, 11, 12).

4.2.2. Solution

L'arborescence est représentée à la figure 2. Evaluations à la base : $c_5 + c_1 = 82$

au sommet 1 : $c_1 + c_{12} = 113$, au sommet $\bar{1}$: $c_5 + c_2 = 101$

au sommet 2 : $c_2 + c_{11} = 130$, au sommet $\bar{2}$: $c_5 + c_3 = 103$

au sommet 3 : $c_3 + c_5 = 103$, au sommet $\bar{3}$: $c_5 + c_4 = 104$

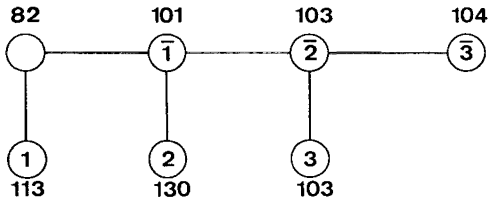

Figure 2

La solution optimale est donc : $x_3 = x_5 = 1$.

4.3. Troisième exemple

4.3.1. Le problème

Minimiser $x_1 + 2x_2 + 3x_3 + 4x_4 + \ldots + 18x_{18} + 19x_{19} + 20x_{20}$ sous les contraintes (1, 2, 5, 7, 13), (2, 7, 14, 18), (3, 9, 10, 11), (4, 6, 8, 12), (4, 9, 11, 13, 19), (5, 10, 15), (5, 16, 20), (8, 14, 15, 17).

4.3.2. Solution

L'arborescence est représentée à la figure 3. Voici comment on a calculé les évaluations par défaut.

A la base : $c_5 + c_3 + c_4 + c_2 = 14$
Au sommet 5 : $c_5 + c_{14} + c_4 + c_3 = 26$
Au sommet 4 : $c_5 + c_4 + c_3 + c_{14} = 26$
Au sommet 14 : $c_5 + c_4 + c_3 + c_{14} = 26$
Au sommet $\bar{5}$: $c_{16} + c_{10} + c_4 + c_2 = 32$
Au sommet $\bar{4}$: $c_5 + c_{14} + c_6 + c_9 = 34$
Au sommet $\overline{14}$: $c_5 + c_{14} + c_3 + c_8 + c_{18} = 48$

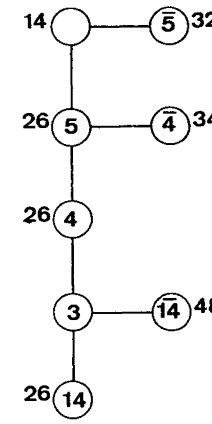

Figure 3

La solution optimale est donc :

$$x_5 = x_4 = x_3 = x_{14} = 1.$$

4.4. Quatrième exemple

4.4.1. Le problème

Comme l'exemple 4.3, mais les contraintes sont des inégalités.

4.4.2. Solution

L'arborescence est représentée à la figure 4. Voici comment on a calculé les évaluations par défaut.

Au sommet 5 : $5 + 3 + 4 + 2 = 14$
Au sommet $\bar{5}$: $16 + 10 + 4 + 2 = 32$
Au sommet 4 : $5 + 4 + 3 + 8 = 20$
Au sommet $\bar{4}$: $5 + 9 + 6 + 2 = 22$
Au sommet 14 : $5 + 4 + 14 + 3 = 26$
Au sommet $\overline{14}$: $5 + 4 + 8 + 3 + 2 = 22$
Au sommet 2 : $5 + 4 + 2 + 8 + 3 = 22$
Aux sommets 3, 8 : $5 + 4 + 2 + 3 + 8 = 22$
Aux sommets $\bar{2}$, $\bar{3}$, $\bar{8}$: inutiles.

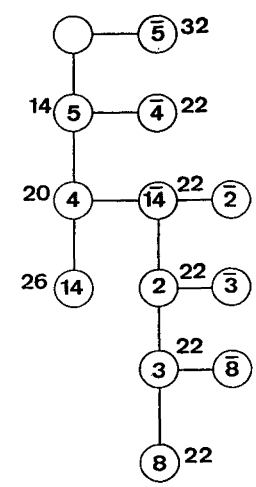

Figure 4

Une solution optimale est donc :

$$x_5 = x_4 = x_2 = x_3 = x_8 = 1.$$

PARTIE B

UTILISATION DES VARIABLES BIVALENTES ET DES COUTS PONDERES DANS LE PROBLEME DES TOURNEES

1. Le problème des tournées à centres multiples

1.1. Les dépôts distributeurs seront désignés par Ω_α, Ω_β,..., Ω_λ. Les clients seront désignés par ω_i, $i \in I$. Un camion peut partir d'un dépôt distributeur pour revenir au même dépôt, mais il peut aussi terminer sa livraison dans un autre dépôt. On appelle *tournée* tout chemin (passant par les clients) reliant deux dépôts; lorsque ces deux dépôts sont identiques, on dira aussi *circuit* au lieu de tournée. Une tournée est dite *admissible* lorsqu'elle satisfait à toutes les contraintes imposées : contraintes sur les clients, celles sur les dépôts distributeurs, contraintes sur les camions, sur les horaires de transport...

Il s'agit de minimiser le coût de transport (somme des coûts des tournées).

1.2. Nous avons vu, dans Vo-Khac [8], comment on peut utiliser la pondération des coûts pour avoir une bonne solution réalisable du problème des tournées à un dépôt distributeur. Dans le cas de plusieurs dépôts distributeurs, la méthode est la même, mais la *pondération* se fait de la manière suivante :

Si l'on sait à l'avance le nombre ν_α de fois qu'une tournée arrive à (ou part de) un dépôt distributeur Ω_α, on remplace Ω_α par ν_α centres fictifs. Si l'on ne sait pas à l'avance les nombres ν_α, alors on prendra $\nu_\alpha = \ldots \nu_\lambda \simeq \frac{\nu}{l}$, où ν est le nombre des tournées nécessaires et l le nombre des dépôts distributeurs.

1.3. Nous allons voir comment l'utilisation conjointe de la pondération et des variables booléennes permet de résoudre le problème des tournées.

2. Transformation du problème des tournées en un problème de programmation linéaire à variables et à contraintes booléennes

Le principe est extrêmement simple.

2.1. Calculs préliminaires

Trois phases à opérer :

a) on dénombre *toutes* les tournées *admissibles* T_j, $j \in J$,

b) on calcule le coût c_j de chaque tournée T_j, $j \in J$,

c) on détermine la matrice d'appartenance des clients aux tournées :

$a_{i,j} = 1$ si le client ω_i appartient à la tournée T_j, $(i \in I)$,

$= 0$ sinon $(i \in I)$.

2.2. Programme à variables bivalentes

Pour tout $j \in J$, soit x_j, la variable booléenne caractéristique de la tournée T_j : $x_j = 1$ si le transport utilise effectivement la tournée T_j, $x_j = 0$ sinon.

(Comme chaque tournée admissible est utilisée une fois au plus, x_j doit être une variable booléenne.) Exprimons que chaque client se trouve sur une tournée et une seule :

$$\sum_{j \in J} a_{ij} x_j = 1, \quad i \in I.$$

Nous sommes alors amenés à la détermination d'un $x = (x_j, j \in J)$, élément de $\mathbf{B}^J$ et satisfaisant aux contraintes précédemment écrites, qui minimise la fonction économique ψ donnée par :

$$\psi(x) = \sum_{j \in J} c_j x_j.$$

C'est précisément le problème que nous avons considéré à la partie *A*.

3. Réduction du nombre de tournées admissibles

Le principe de transformation du problème des tournées en un problème de programme en variables bivalentes exposé au paragraphe 2 est simple; l'algorithme de résolution de ce programme, exposé dans la partie *A*, n'est guère plus compliqué.

La difficulté est due à l'*énormité* du nombre de variables bivalentes (c'est-à-dire des tournées admissibles). Lorsque l'on a *un très grand nombre de contraintes naturelles*, la méthode s'applique facilement (cf. exemple numérique 4.1 de cette partie). Sinon, *il faut créer des contraintes artificielles* pour réduire le nombre des tournées admissibles (cf. exemple numérique 4.2 de cette partie).

3.1. Première réduction

La première réduction est basée sur la contrainte relative au *nombre* ν *des tournées*. On doit fixer la valeur de ν. Cette valeur peut être obtenue rigoureusement par un raisonnement mathématique ou en admettant le principe approximatif suivant : *une solution optimale à* ν *tournées est souvent meilleure qu'une solution optimale à* $\nu + 1$ *tournées*. On a donc intérêt à chercher (d'abord) une solution optimale comportant le nombre minimum de tournées (ce nombre est facilement calculable en utilisant les diverses contraintes).

Lorsque le nombre ν des tournées est fixé, on fixe du même coup d'autres contraintes (de tonnages ou d'effectifs).

3.2. Deuxième réduction

Deux tournées sont dites équivalentes si elles passent par les mêmes clients et dépôts distributeurs. L'ensemble des classes d'équivalence obtenu sera appelé l'*ensemble des secteurs.* Un secteur est donc représenté par plusieurs tournées équivalentes. Parmi les tournées équivalentes, on peut définir une relation d'ordre : soient T_1 et T_2 deux tournées équivalentes, on dit que T_1 est dominé par T_2 si le coût de T_1 est supérieur (ou égal) au coût de T_2. Chaque élément maximal sera appelé *tournée dominante* du secteur; ce n'est autre que la tournée la moins coûteuse du secteur.

Dans le programme à variables bivalentes, on ne retient que les tournées dominantes, et le problème des tournées devient un problème de sectorisation.

Comment trouve-t-on rapidement la tournée dominante (ou « quasi-dominante ») ? Dans le cas d'un dépôt distributeur unique, le problème est celui du voyageur de commerce; dans le cas où les dépôts distributeurs sont différents, c'est le problème du plus court chemin entre deux sommets fixes d'un graphe. On utilise alors les coûts régularisés, exposés dans Vo-Khac [7].

3.3. Troisième réduction

Il s'agit de l'interdiction des tournées mathématiquement inadmissibles, car l'utilisation de ces tournées ne rend certainement pas minimale la fonction économique.

Voici un exemple simple. Supposons que l'on ait trouvé une *solution réalisable* de coût total C. Classons les tournées admissibles par ordre des coûts croissants. Posons $c' = C - \sum_{j=1}^{\nu-1} c_j$ (où ν est le nombre des tournées nécessaires) ; alors *toute tournée de coût supérieur à c' est économiquement inadmissible.*

3.4. Quatrième réduction

Il s'agit de l'interdiction des tournées « *intuitivement* » *inadmissibles.* Cette réduction est extrêmement importante.

L'intuition montre par exemple que dans une solution optimale deux clients « trop éloignés » ne sont pas situés sur une même tournée. Autrement dit, une tournée passant par deux clients éloignés est intuitivement inadmissible. Comment peut-on systématiser cette considération intuitive? Que signifie l'adjectif éloigné?

Nous proposons la réponse suivante : éloigné veut dire coûteux au point de vue des *coûts pondérés* (cf. Vo-Khac [8] et le paragraphe 1 de cette partie *B*).

On cherche la meilleure solution réalisable obtenue par la méthode de sectorisation hiérarchique (cf. Vo-Khac [8]); on retient la dernière arête utilisée dans cette solution ainsi que toutes les arêtes moins coûteuses (toujours au point de vue de coûts pondérés) que cette arête. Toutes les arêtes retenues sont dites *intuitivement admissibles* (footnote). On admet alors le principe approximatif suivant : *deux clients ne peuvent appartenir à un même secteur que s'il existe une arête intuitivement admissible les reliant.*

La meilleure sectorisation obtenue en utilisant ce principe sera appelée la *sectorisation intuitivement optimale.*

Cette quatrième réduction n'est pas rigoureuse; mais elle permet d'énormes simplifications (cf. l'exemple numérique 4.2 de cette partie).

Footnote. — Si l'on est prudent, on retient aussi les arêtes légèrement plus coûteuses que la dernière arête utilisée dans la sectorisation hiérarchique par coûts pondérés.

4. Exemples numériques

Voici deux exemples numériques, déjà étudiés dans Vo-Khac [8]. On les reprendra ici en utilisant la méthode exposée précédemment.

4.1. Exemple de Marconi

On fera les deux premières réductions (nombre de tournées nécessaires et tournées dominantes). Comme on ne fera la quatrième réduction (inadmissibilité intuitive), on n'a pas besoin des coûts pondérés et on utilisera les coûts bruts.

4.1.1. Réductions

a) *Première réduction*

On va former une bisectorisation ($\nu = 2$). Comme le total des quantités à livrer est de 40 000 et comme la capacité des camions est de 20 000, cela entraîne que la capacité de chaque secteur est 20 000. Comme trois clients quelconques forment un secteur de capacité au plus 19 000 (et comme 5 clients quelconques forment un secteur de capacité au moins 21 000), l'effectif de chaque secteur est nécessairement 4. Les secteurs admissibles sont alors (en tenant compte des contraintes de capacité) : { 1, 2, 3, 4 }, { 1, 2, 4, 6 }, { 1, 2, 4, 8 }, { 1, 3, 4, 6 }, { 1, 3, 4, 8 }, {1, 4, 6, 8 }, { 5, 6, 7, 8 }, { 3, 5, 7, 8 }, { 3, 5, 6, 7 }, { 2, 5, 7, 8 }, { 2, 5, 6, 7 } et { 2, 3, 5, 7 }.

b) *Deuxième réduction*

On cherche le circuit dominant de chaque secteur par la méthode exposée dans Vo-Khac [7]. Voici les 12 circuits dominants, classés par ordre des coûts croissants.

j	circuit dominant	c_j	j	circuit dominant	c_j	j	circuit dominant	c_j
1	(0, 3, 5, 7, 2, 0)	30	5	(0, 3, 1, 4, 6, 0)	52	9	(0, 6, 8, 5, 7, 0)	70
2	(0, 2, 5, 7, 6, 0)	49	6	(0, 2, 3, 1, 4, 0)	55	10	(0, 1, 4, 2, 8, 0)	81
3	(0, 2, 5, 7, 8, 0)	51	7	(0, 2, 1, 4, 6, 0)	67	11	(0, 4, 1, 3, 8, 0)	81
4	(0, 3, 5, 7, 8, 0)	52	8	(0, 3, 5, 7, 6, 0)	67	12	(0, 1, 4, 6, 8, 0)	83

4.1.2. Programme linéaire à variables bivalentes

Il s'agit dès lors de minimiser

$$30x_1 + 49x_2 + 51x_3 + 52x_4 + 52x_5 + 55x_6 + 67x_7 + 67x_8 + 70x_9 + 81x_{10} + 81x_{11} + 83x_{12},$$

sous les contraintes booléennes :

$$\begin{aligned}
x_1 + x_2 + x_3 + x_4 + x_8 + x_9 &= 1 \\
x_1 + x_2 + x_3 + x_6 + x_7 + x_{10} &= 1 \\
x_1 + x_4 + x_5 + x_5 + x_8 + x_{11} &= 1 \\
x_2 + x_5 + x_7 + x_8 + x_9 + x_{12} &= 1 \\
x_3 + x_4 + x_9 + x_{10} + x_{11} + x_{12} &= 1 \\
x_5 + x_6 + x_7 + x_{10} + x_{11} + x_{12} &= 1
\end{aligned}$$

Ce problème a été résolu à la partie *A* (exemple 4.2). On a trouvé $x_3 = x_5 = 1$. Par conséquent la bisectorisation (rigoureusement) optimale est $\{2, 5, 7, 8\}$, $\{3, 1, 4, 6\}$, avec coût égal à $51 + 52 = 103$. On retrouve le résultat établi dans Vo-Khac [8].

4.2. Exemple de Dantzig

On fera les 4 réductions; les affinités pondérées sont calculées dans [8].

4.2.1. Réduction

a) *Première réduction*

On essaie de minimiser le nombre des tournées; par conséquent, on va faire une quadri-sectorisation ($\nu = 4$).

Le fait de fixer $\nu = 4$ implique des contraintes sur la capacité et l'effectif de chaque secteur. En effet, le nombre total des quantités à livrer étant 18 200 et la capacité maximale des camions étant 6 000, la capacité de chaque secteur est comprise entre 200 et 6 000. D'autre part, 5 clients exigeant au moins une capacité égale à 6 300, le nombre maximum de clients par secteurs est 4.

b) *Quatrième réduction*

La méthode hiérarchique utilisant les affinités pondérées [8] permet d'obtenir la quadri-sectorisation suivante : { 1, 2, 3, 4 }, { 5 }, { 6, 8, 9 }, { 7, 10, 11, 12 }, la dernière arête utilisée étant l'arête 6-8 d'affinité 520. Toutes les arêtes d'affinités supérieures ou égales à 520 sont donc intuitivement admissibles et les autres arêtes sont inadmissibles intuitivement.

On voit alors qu'aucune arête intuitivement admissible relie le client 5 à un autre client; le client 5 constitue donc à lui-seul un secteur. D'autre part, aucune arête intuitivement admissible reliant l'ensemble des clients { 1, 2, 3, 4 } aux autres clients; cet ensemble { 1, 2, 3, 4 } constitue donc un secteur. Finalement, tout revient à partager l'ensemble des 7 clients restants {6, 7, 8, 9, 10, 11, 12} en deux secteurs.

c) *Première et deuxième réduction*

Il en résulte de nouvelles contraintes sur l'effectif et la capacité de chaque secteur : la capacité de chaque secteur doit être comprise entre 4 700 et 6 000 et chaque secteur doit contenir 3 ou 4 clients. Voici les secteurs admissibles, leurs circuits dominants et les affinités pondérées de ces circuits.

d) *Troisième réduction*

On connaît la solution réalisable suivante 0-7-10-11-0-6-8-9-0, d'affinité 2 876 (cf. Vo-Khac [8]). Or le circuit d'affinité maximale est 0-6-9-11-12-0, d'affinité 1 844. Tout circuit d'affinité inférieure (ou égale) à 2 876-1 844 = 1 032 doit être éliminé. Il ne reste alors que 21 circuits admissibles. On voit facilement que cette réduction implique automatiquement l'inadmissibilité des circuits numéros 1, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15 (car le circuit n° 1, par exemple, est le circuit complémentaire du circuit n° 28, économiquement inadmissible).

j	circuit dominant	p_j	j	circuit dominant	p_j	j	circuit dominant	p_j
1	6-9-11-12	1 844	12	6-9-10-7	1 592	23	8-10-9	1 028
2	7-10-11-12	1 828	13	7-8-12-9	1 584	24	8-12-9	1 012
3	6-10-11-12	1 800	14	6-9-12-7	1 576	25	6-9-11	1 000
4	7-11-12-8	1 772	15	6-8-12-7	1 560	26	7-8-9	992
5	7-9-11-12	1 732	16	7-9-10-12	1 556	27	9-11-10	980
6	7-12-10-8	1 708	17	8-11-12	1 116	28	7-10-8	976
7	6-10-12-9	1 696	18	8-11-9	1 092	29	6-10-11	956
8	6-8-10-12	1 680	19	6-8-11	1 068	30	6-8-10	948
9	6-11-12-7	1 680	20	6-8-9	1 048	31	6-9-10	936
10	6-10-12-7	1 616	21	8-10-11	1 048	32	7-8-11	928
11	6-11-10-7	1 612	22	7-11-9	1 028			

4.2.2. Programme à variables bivalentes

Finalement, on est conduit à maximiser

$$\varphi(x) = 1\,828x_2 + 1\,616x_{10} + 1\,592x_{12} + 1\,576x_{14} + 1\,556x_{16} + 1\,116x_{17} + 1\,092x_{18} + 1\,068x_{19} + 1\,048x_{20} + 1\,048x_{21}.$$

Sous les contraintes suivantes :

$$\begin{aligned}
x_2 + x_{10} + x_{12} + x_{14} + x_{16} &= 1\\
x_2 + x_{10} + x_{12} + x_{16} + x_{21} &= 1\\
x_2 + x_{10} + x_{14} + x_{16} + x_{17} &= 1\\
x_2 + x_{17} + x_{18} + x_{19} + x_{21} &= 1\\
x_{10} + x_{12} + x_{14} + x_{19} + x_{20} &= 1\\
x_{12} + x_{14} + x_{16} + x_{18} + x_{20} &= 1\\
x_{17} + x_{18} + x_{19} + x_{20} + x_{21} &= 1
\end{aligned}$$

Comme on a :

$$x_2 + x_{10} + x_{12} + x_{14} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} = 2,$$

maximiser $\varphi(x)$ revient à minimiser $\psi(x)$ avec :

$$\psi(x) = 212x_{10} + 236x_{12} + 252x_{14} + 272x_{16} + 712x_{17} + 736x_{18} + 760x_{19} + 780x_{20} + 780x_{21}.$$

Ce dernier problème a été résolu à la partie *A* (exemple numérique 4.1) : $x_2 = x_{20} = 1$. Par conséquent la quadrisectorisation « intuitivement » optimale du problème de Dantzig est { 1, 2, 3, 4 }, { 5 }, { 6, 8, 9 }, { 7, 10, 11, 12 }.

BIBLIOGRAPHIE

[1] BALAS (E.), « Un algorithme additif pour la résolution des programmes linéaires en variables bivalentes » *C. R. Acad. Sc. Paris*, 258 (1964), 3817-3820. — An additive algorithm for solving linear programs with 0-1 variables. *Journal of Operations Research Society of America* 13 (1965), 517-547. — Discrete programming for the Filter Method. *Journal of Operations Research Society of America*,15 (1967), 915-957.

[2] BALINSKI (M. L.) et QUANDT (R. E.), On an integer program for a delivery problem *Journal of Operations Research Society of America* (12 (1964), 300-304.

[3] GILMORE (P. C.) et GOMORY (R. E.), « A linear programming approach to the cutting stock problem », *J.O.R.S.A.*, 11 (1963), 863-888. — « The theory and computation of knapsack functions », *J.O.R.S.A.*, 14 (1966), 1045-1074.

[4] LAWLER (E. L.) et BELL (M. D.), « A method for solving discrete optimization problems », *J.O.R.S.A.*, 14 (1966) ,1098-1112.

[5] ROY (B.), NGHIEM (P. T.) et BERTIER (P.), Programmes linéaires en nombres entiers et procédures SEP. *METRA*, 4 (1965), 441-460.

[6] SHAPIRO (J. F.), « Dynamic programming algorithms for the integer programming problems », *J.O.R.S.A.*, 16 (1968), 103-121.

[7] VO-KHAC (K.), « La régularisation dans les problèmes combinatoires et son application au problème des tournées de livraison », *Revue française d'Informatique et de Recherche Opérationnelle* 3e année (1969), 91-104, V. 1, 91-104.

[8] VO-KHAC (K.), « La régularisation dans les problèmes combinatoires et son application au problème de sectorisation », *R.I.R.O.*, 5e année (1971), V. 2, 59-77.