RAIRO. RECHERCHE OPÉRATIONNELLE

C. PASCHE

Flot à coût convexe linéaire par morceaux

RAIRO. Recherche opérationnelle, tome 21, n° 3 (1987), p. 205-217

<http://www.numdam.org/item?id=RO_1987__21_3_205_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 21, n° 3, août 1987, p. 205 à 217)

FLOT À COÛT CONVEXE LINÉAIRE PAR MORCEAUX (*)

par C. PASCHE (1)

Résumé. — *Seul un algorithme spécialisé permet de résoudre rapidement les problèmes de flot à coût convexe linéaire par morceaux. Une variante de l'algorithme du simplexe est présentée; la méthode de pivotage utilisée permet d'effectuer plusieurs modifications de flot pour un seul pivot. Des tests sur deux types de problèmes montrent l'efficacité de ce nouveau pivotage.*

Mots clés : Problème de flot; algorithme du simplexe; fonction linéaire par morceaux.

Abstract. — *To solve network flow problems with piecewise linear convex cost function with efficiency, a specific algorithm must be used. A modified network simplex method is developed; the pivoting strategy used allows several flow modifications in one pivot. Computational experiments, on two types of problems, show the efficiency of this new pivot operation.*

Keywords : Network optimization; network simplex; piecewise linear programming.

0. INTRODUCTION

Cet article traite du problème de flot à coût minimum, où le coût de transit du flot sur chaque arc est une fonction convexe linéaire par morceaux. Ce problème sera appelé problème de flot à coût convexe linéaire (PFCCL).

De nombreux problèmes pratiques peuvent se formuler comme un PFCCL; c'est le cas du problème d'augmentation à moindre coût de la capacité d'un réseau [1], de certains problèmes liés aux réseaux téléphoniques [5] et de certains problèmes de transport urbain [9]. Le dual de PFCCL peut être interprété comme un problème d'ordonnancement de tâches de durées variables [1], [7]. Le PFCCL peut être utilisé pour résoudre, par approximation, un problème de flot dans lequel il faut minimiser une fonction convexe des transits; de tels réseaux peuvent être associés à des problèmes de répartition

(*) Reçu décembre 1986.
(1) École Polytechnique Fédérale de Lausanne, Département de Mathématiques, MA (Ecublens), CH-1015 Lausanne, Suisse.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/87/03 205 13/$ 3.30

de courant dans des réseaux électriques [10] et de transport de liquide dans un système de conduites [6].

En théorie, on résout le PFCCL en le transformant, par décomposition des arcs en plusieurs arcs parallèles, en un problème de flot à coût linéaire. Pour être efficace, un algorithme doit traiter ces variables supplémentaires de manière implicite. Monma et Segal [15] proposent une structure de données adaptées au PFCCL. Ahaja, Batra et Gupta [1] ont développé un algorithme dual spécifique.

Pour utiliser au mieux la structure particulière du PFCCL, une nouvelle stratégie de pivotage a été introduite dans l'algorithme primal du simplexe. Plusieurs pivots successifs peuvent être combinés; c'est-à-dire que la nouvelle opération de pivotage effectue plusieurs modifications de flot pour une seule mise-à-jour de la base et des variables duales.

1. PRÉSENTATION DU PROBLÈME

Considérons un réseau $\mathbf{R}=(\mathbf{V}, \mathbf{E})$ formé d'un ensemble $\mathbf{V}$ de n sommets et d'un ensemble $\mathbf{E}$ de m arcs. Le problème se formule de la façon suivante :

$$\underset{X_{(i,j)}}{\text{Minimiser}} \sum_{(i,j)\in \mathbf{E}} C_{(i,j)}(X_{(i,j)}) \tag{1}$$

s. c.

$$\sum_{j\in \mathbf{S}_i} X_{(i,j)} - \sum_{j\in \mathbf{P}_i} X_{(i,j)} = B_i, \qquad \forall i \in V \tag{2}$$

$$0 \leqslant X_{(i,j)} \leqslant U_{(i,j)}, \qquad \forall (i,j)\in \mathbf{E} \tag{3}$$

où

$-\mathbf{S}_i=\{j \in \mathbf{V} \mid (i, j) \in \mathbf{E}\}$ et $\mathbf{P}_i=\{j \in \mathbf{V} \mid (j, i) \in \mathbf{E}\}$;

$-X_{(i,j)}$ représente le flot transitant sur l'arc (i, j) de i vers j;

$-U_{(i,j)}$ est la capacité de l'arc (i,j);

$-B_i$ est le bilan au sommet i; pour que le problème possède une solution on impose :

$$\sum_{i\in \mathbf{V}} B_i = 0$$

$-C_{(i,j)} : [0, U_{(i,j)}] \to \mathbb{R}$, fonction convexe linéaire par morceaux, représente le coût de transfert du flot sur l'arc (i,j). Elle est composée de $K(i,j)$ fonctions

linéaires données par :

- les points de cassures $0 = U_{(i,j),0} < U_{(i,j),1} < \ldots < U_{(i,j),K(i,j)} = U_{(i,j)}$;
- les coûts unitaires $C_{(i,j),1} < \ldots < C_{(i,j),K(i,j)}$.

Sur le segment numéro k, correspondant à l'intervalle $[U_{(i,j),k-1}, U_{(i,j),k}]$, le coût est linéaire de pente $C_{(i,j),k}$. La figure 1 fournit un exemple de coût.

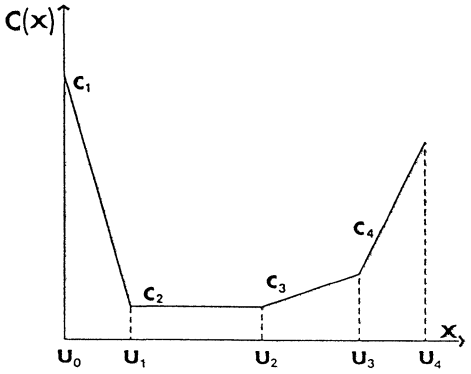

Figure 1

Ce problème peut être transformé en un problème linéaire en introduisant pour chaque arc $(i,j) \in \mathbf{E}$, $K(i,j)$ variables $Y_{(i,j),1}, \ldots, Y_{(i,j),K(i,j)}$. La variable $Y_{(i,j),k}$ représentant la part du flot $X_{(i,j),k}$ transportée au coût unitaire $C_{(i,j),k}$. On a

$$X_{(i,j)} = \sum_{k=1}^{K(i,j)} Y_{(i,j),k}. \tag{4}$$

Le problème se reformule alors de la manière suivante :

$$\text{Minimiser} \sum_{(i,j) \in \mathbf{E}} \sum_{k=1}^{K(i,j)} C_{(i,j),k}\, Y_{(i,j),k} \tag{5}$$

$$\sum_{j \in \mathbf{S}_i} \sum_{k=1}^{K(i,j)} Y_{(i,j),k} - \sum_{j \in \mathbf{P}_i} \sum_{k=1}^{K(j,i)} Y_{(j,i),k} = B_i, \qquad \forall i \in V \tag{6}$$

$$0 \leqslant Y_{(i,j),k} \leqslant U_{(i,j),k} - U_{(i,j),k-1}, \qquad \forall (i,j) \in \mathbf{E}, \quad \forall k = 1, \ldots, K(i,j) \tag{7}$$

$$\left.\begin{array}{r} Y_{(i,j),k} > 0 \Rightarrow Y_{(i,j),k-1} = U_{(i,j),k-1} - U_{(i,j),k-2} \\ \forall (i,j) \in \mathbf{E} \qquad \forall k = 2, \ldots, K(i,j) \end{array}\right\} \tag{8}$$

Les contraintes (8) sont superflues; en effet les coûts unitaires $C_{(i,j),k}$ étant croissants pour $k = 1, \ldots, K(i,j)$, elles sont forcément vérifiées dans une

solution optimale. Les contraintes restantes définissent un problème de flot à coût linéaire minimum, où chaque arc (i,j) a été remplacé par $K(i,j)$ arcs parallèles. La figure 2 linéarise le coût donné par la figure 1.

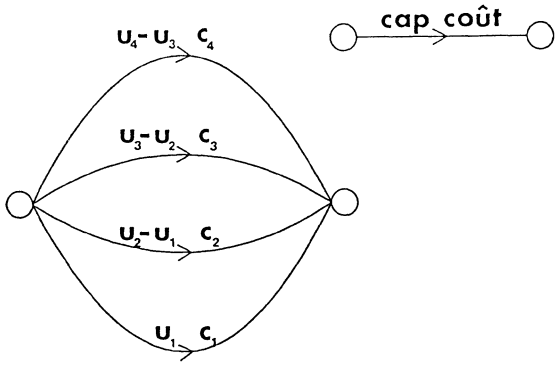

Figure 2

Après cette transformation, le problème peut être résolu par tout algorithme classique de flot à coût minimum. L'efficacité d'un tel algorithme peut cependant être augmentée s'il prend en compte implicitement l'existence de $K(i,j)$-uples d'arcs parallèles. Au lieu de modifier le réseau par éclatement des arcs, on introduit la notion d'état $e(i,j)$ pour les arcs $(i,j)\in\mathbf{E}$. On détermine $e(i,j)$ de sorte que :

$$e(i,j)=k \quad \Rightarrow \quad X_{(i,j)}\in[U_{(i,j),k-1},\ U_{(i,j),k}]. \tag{9}$$

L'implication (9) ne définit pas l'état de manière univoque. En effet pour $X_{(i,j)}=U_{(i,j),k}$ $(k\neq K(i,j))$ l'état de l'arc (i,j) peut valoir k ou $k+1$; ceci dépend de la modification du flot $X_{(i,j)}$ envisagée :

- si c'est une augmentation alors $e(i,j)=k+1$;
- si c'est une diminution alors $e(i,j)=k$.

Dans la suite, nous allons montrer comment intégrer la notion d'état à l'algorithme du simplexe pour les réseaux (*cf.* [4, 12, 13]).

2. L'ALGORITHME DU SIMPLEXE

Avant de décrire l'algorithme, il est nécessaire de définir les notions de base, de flot de base et de coût réduit pour ce problème particulier.

Une base est formée d'un ensemble $\mathbf{T}\subset\mathbf{E}$ de $n-1$ arcs d'un arbre, d'un sommet racine $r\in\mathbf{V}$ et des états $e(i,j)$ pour les arcs $(i,j)\in\mathbf{E}$.

Un flot de base $\{X_{(i,j)};\ (i,j)\in E\}$ est un flot (contraintes 12 et 13) tel que :

– les variables de base $X_{(i,j)}$, $(i,j)\in\mathbf{T}$, sont dans le segment numéro $e(i,j)$; c'est-à-dire :

$$X_{(i,j)}\in[U_{(i,j),e(i,j)-1},\ U_{(i,j),e(i,j)}],\qquad \forall (i,j)\in\mathbf{T} \tag{10}$$

– les variables hors base $X_{(i,j)}$, $(i,j)\in\mathbf{E}-\mathbf{T}$, sont égales à l'une des valeurs qu'elles peuvent prendre en un point de rupture; c'est-à-dire :

$$X_{(i,j)}=U_{(i,j),e(i,j)-1}\quad\text{ou}\quad U_{(i,j),e(i,j)},\qquad \forall (i,j)\in\mathbf{E}-\mathbf{T} \tag{11}$$

A chaque sommet $i\in\mathbf{V}$, on attribue une variable duale P_i donnant la longueur de l'unique chaîne de $\mathbf{T}$ allant de i vers la racine r. Cette longueur est calculée en utilisant les coûts des arcs correspondant à leur état; les arcs parcourus dans le bon sens ayant une contribution positive, ceux parcourus en sens inverse ayant une contribution négative.

Ces variables duales permettent de définir pour tout arc hors base (i,j) deux coûts réduits :

– $ra(i,j)$ un coût réduit d'augmentation,

– $rd(i,j)$ un coût réduit de diminution.

Pour $(i,j)\in\mathbf{E}-\mathbf{T}$ avec $X_{(i,j)}=U_{(i,j),k}$ on pose :

$$ra(i,j)=\left\{\begin{array}{cc}\infty & \text{si}\quad k=K(i,j)\\ P_j-P_i+C_{(i,j),k+1} & \text{sinon}\end{array}\right\} \tag{12}$$

$$rd(i,j)=\left\{\begin{array}{cc}\infty & \text{si}\quad k=0\\ P_i-P_j-C_{(i,j),k} & \text{sinon}\end{array}\right\} \tag{13}$$

L'algorithme du simplexe va procéder par élimination successive de tous les cycles de longueur négative le long desquels le flot peut être modifié. L'énoncé de l'algorithme classique du simplexe pour le PFCCL est présenté dans le tableau I.

Remarques :

– La base et le flot de base initiaux peuvent être obtenus en utilisant la méthode « big-M » ([4, 16]).

– La règle de Cunningham [5] pour le choix de l'arc sortant produit un algorithme fini.

– Les opérations sur les arbres peuvent être implantées de manière très efficace ([2, 12, 13]).

– Il est facile de modifier cet algorithme de façon à traiter des problèmes où les contraintes (3) sont remplacées par :

$$L_{(i,j)} \leqslant X_{(i,j)} \leqslant U_{(i,j)}, \qquad \forall (i,j) \in E \tag{14}$$

si $L_{(i,j)}$ est négatif, l'arc est dit biorienté et $X_{(i,j)}$ négatif signifie que $-X_{(i,j)}$ unités transitent de j vers i.

TABLEAU I

Algorithme du simplexe.

(0) *Initialisation.*
– Trouver une base **T** et un flot de base $\{X_{(i,j)};\ (i,j) \in E\}$.
(1) *Choix de l'arc entrant :* (p, q).
– Choisir un arc $(p, q) \in \mathbf{E} - \mathbf{T}$ de coût réduit négatif.
– S'il n'en existe pas STOP « le flot est optimal ».
– Changer l'état de l'arc entrant (p, q) (supposons que $X_{(p,q)} = U_{(p,q),k}$)
si $ra(p, q) < 0$ alors $e(p, q) := k + 1$ et on va augmenter le flot sur l'arc (p, q)
si $rd(p, q) < 0$ alors $e(p, q) := k$ et on va diminuer le flot sur l'arc (p, q).
(2) *Calcul de l'amplitude de la modification.*
– Chercher le cycle **C** formé par l'arc (p, q) et l'arbre **T** de la base.
– Orienter **C** selon (p, q) pour augmenter le flot et à l'inverse de (p, q) pour le diminuer.
– Calculer la quantité maximale $\triangle$ de flot qui peut être canalisé par **C** sans qu'aucun arc ne doive changer d'état.
– Choisir comme arc sortant (u, v) l'un des arcs qui détermine $\triangle$.
(3) *Modification du flot.*
– Modifier $X_{(i,j)}$ pour $(i,j) \in \mathbf{C}$ en faisant avancer les $\triangle$ unités de flot.
(4) *Mise à jour de la base.*
– $\mathbf{T} := \mathbf{T} + (p, q) - (u, v)$.
– Mettre à jour les variables duales $P_i, i \in \mathbf{V}$.
– Retourner à (1).

3. COMBINAISON DE PIVOTS

Le choix de l'arc entrant dans la base est crucial pour l'efficacité de l'algorithme du simplexe; plusieurs auteurs [4, 8, 16] proposent des stratégies de choix pour une fonction coût linéaire. Pour exploiter la structure particulière du problème PFCCL (qui possède conceptuellement un grand nombre d'arcs parallèles), il faut utiliser au maximum les cycles de coût négatif. Une méthode simple consiste à essayer de faire entrer la variable sortante. Après chaque pivotage, on calcule le nouveau coût réduit de l'arc sortant; s'il est négatif, on le choisit comme arc entrant pour le pivotage suivant, sinon on utilise une des stratégies classiques.

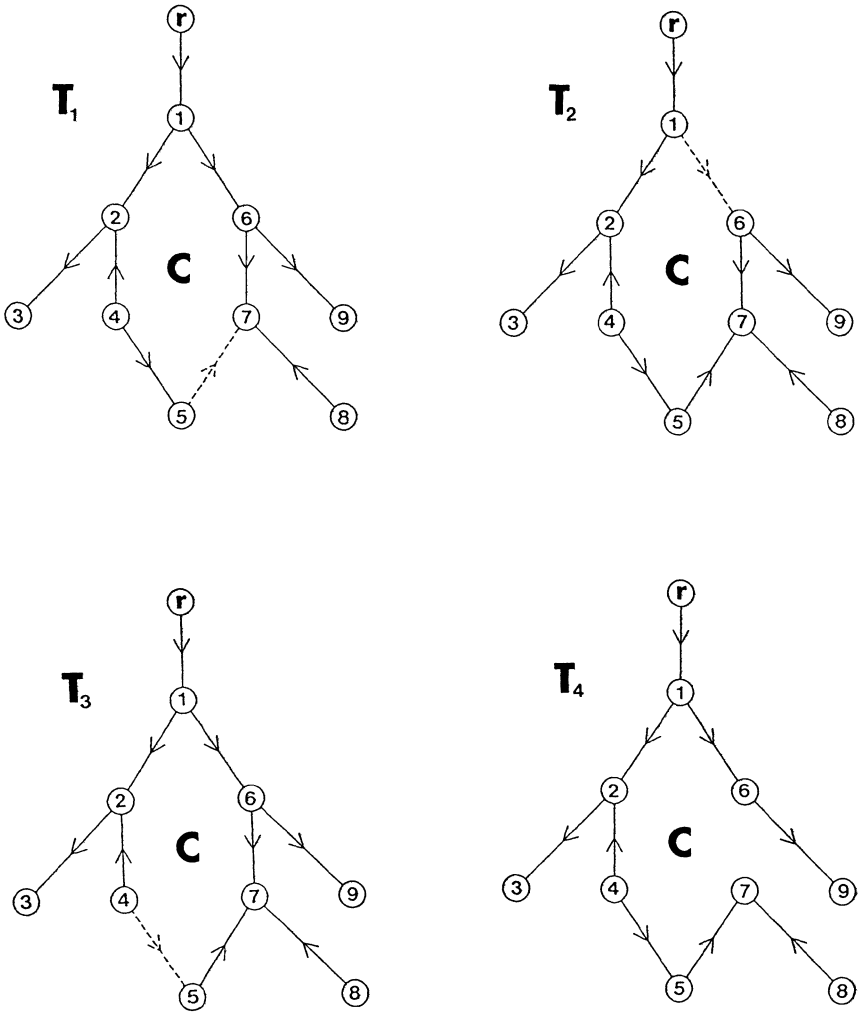

Figure 3

La figure 3 illustre le traitement d'un cycle **C** par cette méthode :

– le pivot 1 fait entrer (5,7) et sortir (1,6); le coût réduit de (1,6) est négatif;

– le pivot 2 fait entrer (1,6) et sortir (4,5); le coût réduit de (4,5) est négatif;

– le pivot 3 fait entrer (4,5) et sortir (6,7); le coût réduit de (6,7) est positif.

Pour modifier le flot le long de **C**, les arbres $\mathbf{T}_2$ et $\mathbf{T}_3$ sont inutiles; on peut passer directement de $\mathbf{T}_1$ à $\mathbf{T}_4$ en posant $\mathbf{T}_4 = \mathbf{T}_1 - (5,7) + (6,7)$. En règle générale quel que soit le nombre de modifications successives effectuées le long d'un cycle, un seul changement de base est nécessaire; c'est le principe des pivots combinés.

Pour incorporer cette nouvelle manière de pivoter à l'algorithme du tableau I, il faut remplacer les points (2) et (3) par les points (2′) et (3′) suivants :

(2′) *Recherche du cycle* **C.**

– Chercher le cycle **C** formé par l'arc (p, q) et l'arbre **T** de la base

– Orienter **C** selon (p, q) pour augmenter le flot et à l'inverse de (p, q) pour le diminuer.

(3′) *Modification du flot le long de* **C.**

– Calculer la quantité maximale $\triangle$ de flot qui peut circuler sur **C** sans qu'aucun arc ne doive changer d'état

– Choisir un arc (u, v) déterminant $\triangle$

– Modifier $X_{(i\ j)}$ pour $(i, j) \in \mathbf{C}$ en faisant avancer les $\triangle$ unités de flot et changer l'état de (u, v).

– Si **C** possède encore un coût réduit négatif, alors aller en (3′).

Cunningham [5] a proposé une règle de choix de l'arc sortant qui évite le cyclage de l'algorithme du simplexe pour les réseaux. Cette règle peut être adaptée au nouveau pivotage. Dans l'étape (3′) on choisit comme arc (u, v) changeant d'état, le premier arc déterminant $\triangle$ rencontré le long de C en partant de la jointure (la jointure étant le sommet de C le moins profond dans l'arbre). Cette règle assure que l'algorithme modifié trouve une solution optimale en un nombre fini de pivots.

La méthode des pivots combinés peut être comparée à l'algorithme du convexe simplexe utilisé pour les problèmes d'optimisation convexe (le lecteur trouvera la description de cet algorithme pour les réseaux aux pages 192-193 de [8]). En général ces deux algorithmes suivent des trajectoires différentes. De plus la continuité des dérivées des fonctions coût est nécessaire pour pouvoir assurer la validité du convexe simplexe. Il est facile de construire de petits PFCCL pour lesquels cet algorithme obtient une solution non optimale.

4. TEST

Pour effectuer les tests, nous avons utilisés deux générateurs de PFCCL programmés selon les principes de Netgen [14] :

– le premier engendre des problèmes où les caractéristiques des arcs (nombre de segments, capacités des segments et coûts des segments) sont

uniformément distribuées. Il fournit les problèmes $P1, \ldots, P20$ présentés dans le tableau II.

TABLEAU II

nom	# sommets	# arcs	# max. de segments	
			négatifs	positifs
P1	100	500	0	4
P2	100	500	0	4
P3	100	500	5	5
P4	300	1500	0	4
P5	300	1500	0	8
P6	500	2000	4	5
P7	500	2000	0	5
P8	500	2500	0	7
P9	700	3000	5	5
P10	700	4000	0	8
P11	750	4000	2	4
P12	750	4000	2	4
P13	1000	5000	3	3
P14	1000	6000	0	4
P15	1000	6000	0	5
P16	1500	5000	0	4
P17	1500	6000	2	2
P18	1500	6000	0	4
P19	2000	6000	0	5
P20	2000	6000	0	3

Note : Les arcs sont biorientés; les colonnes 3 et 4 du tableau donnent le nombre maximal de segments pour le flot positif et le flot négatif.

— le second produit des problèmes où les coûts des arcs sont quadratiques. Ces problèmes sont transformés en approchant le coût quadratique par une fonction linéaire par morceaux possédant un nombre fixé de segments égaux et en faisant une simple interpolation entre les points de cassure. Ce sont

les problèmes $Q1, \ldots, Q10$ dont les caractéristiques sont données dans le tableau III.

TABLEAU III

nom	# sommets	# arcs	# segments
Q1	100	500	30
Q2	100	500	20
Q3	100	1000	18
Q4	200	750	16
Q5	200	1000	14
Q6	200	2000	8
Q7	300	1000	14
Q8	300	1500	12
Q9	300	1500	10
Q10	500	2000	8

Note : Les arcs sont biorientés; les segments sont égaux et disposés symétriquement par rapport à 0.

Les trois programmes testés sont les suivants :

– FLOW qui implante l'algorithme du simplexe pour les flots à coût linéaire. Les structures de donnée utilisées pour représenter la base sont les prédécesseurs, les profondeurs et un ordre transversal [2, 4]. Le temps nécessaire à la transformation d'un PFCCL en un problème à coût linéaire n'est pas pris en compte.

– SGFLOW qui implante un algorithme particularisé aux PFCCL. La notion d'état a été introduite dans le code précédent; les procédures de pivotage restant identiques.

– SGFLOW 1 qui est une variante de SGFLOW dans laquelle les pivots sont combinés. La différence principale d'implantation réside dans la procédure de mise-à-jour des variables duales. Dans l'algorithme classique, cette mise-à-jour s'effectue en additionnant une constante aux variables duales associées aux sommets déconnectés de la racine r par la suppression de l'arc sortant. Pour la nouvelle méthode de pivotage, cette opération est plus

complexe :

– les variables à modifier correspondent aux sommets déconnectés de la racine par le retrait des arcs ayant changé d'état au cours du pivotage. Pour l'exemple de la figure 3 les sommets 5, 6, 7, 8, 9.

– la modification des variables n'est plus uniforme; il faut par conséquent utiliser un ordre transversal pour recalculer les variables duales de ces som-

TABLEAU IV

nom	FLOW	SGFLOW	SGFLOW1
P1	1.4	1.1	0.5
P2	1.2	1.0	0.5
P3	2.8	1.6	1.2
P4	7.1	5.9	3.2
P5	17.8	10.0	3.8
P6	24.8	15.5	10.1
P7	15.8	12.6	6.7
P8	27.2	17.0	7.5
P9	46.1	28.3	18.6
P10	42.4	23.6	12.4
P11	35.6	26.9	27.7
P12	31.2	22.6	20.0
P13	50.2	36.9	26.5
P14	34.8	27.9	15.6
P15	41.3	31.2	22.0
P16	24.5	21.0	15.7
P17	54.0	48.9	62.0
P18	35.5	29.8	29.0
P19	45.3	36.4	21.7
P20	27.5	25.9	17.9
TOTAL:	566.5	424.1	322.6

Note : Les temps de calcul sont donnés en s. sur un Cyber 170/855.

mets. Dans l'exemple, on part du sommet 4 pour mettre à jour P_5, P_7 et P_8, puis à partir de 1 on recalcule P_6 et P_9.

TABLEAU V

nom	FLOW	SGFLOW	SGFLOW1
Q1	60.0	9.7	2.9
Q2	23.6	4.8	2.1
Q3	118.3	18.5	5.0
Q4	41.5	9.9	4.8
Q5	35.5	9.6	5.8
Q6	77.3	24.3	11.6
Q7	41.1	14.8	5.6
Q8	62.9	20.3	10.9
Q9	38.9	14.9	9.9
Q10	55.9	26.6	17.5
TOTAL:	555.0	195.4	76.1

Note : Les temps de calcul sont donnés en s. sur un Cyber 170/855.

L'analyse des temps de calcul donnés dans les tableaux IV et V inspire les remarques suivantes :

– il est avantageux de développer une version du simplexe particularisée aux PFCCL. Pour résoudre les 30 problèmes, SGFLOW et SGFLOW 1 utilisent respectivement 55 % et 35 % du temps de calcul de FLOW.

– l'efficacité de la méthode des pivots combinés dépend du nombre de morceaux composant les coûts de transit et de la charge du réseau. L'expérience montre que le nombre de parties remplies à l'optimum est un des facteurs déterminants. Le problème P11 possède le même réseau que P12, muni de capacités plus grandes; ce réseau est peu chargé ce qui explique la mauvaise performance de SGFLOW 1.

5. CONCLUSION

La résolution de PFCCL en un temps minimal nécessite le développement d'un code spécialisé. La méthode des pivots combinés peut être facilement

introduite dans tout code implantant un algorithme primal pour le PFCCL. Cette idée simple permet, selon les caractéristiques des réseaux, de réduire jusqu'à 50 % du temps de calcul.

BIBLIOGRAPHIE

1. R. K. AHUJA, J. L. BATRA et S. K. GUPTA, *A Parametric Algorithm for Convex Cost Network Flow and Related Problems*, E.J.O.R., vol. 16, 1984, p. 222-235.
2. A. I. ALI, R. V. HELGASON, J. L. KENNINGTON et H. S. HALL, *Primal Simplex Network Codes : State-of-the-Art Implementation Technology*, Network, vol. 8, 1978, p. 315-339.
3. R. BARR, F. GLOVER et D. KLINGMAN, *Enhancement of Spanning Tree Labelling Procedures for Network Optimization*, I.N.F.O.R., vol. 17, 1979, p. 16-34.
4. G. H. BRADLEY, G. G. BROWN et G. W. GRAVES, *Design and Implementation of Large Scale Primal Transhipment Algorithms*, Management Sciences, vol. 24, 1977, p. 1-34.
5. W. H. CUNNINGHAM, *A Network Simplex Method*, Mathematical Programming, vol. 11, 1976, p. 105-110.
6. G. M. FAIR et J. C. GEYER, *Water Supply and Waste Water Disposal*, Wiley, New York, 1954.
7. L. R. FORD et D. R. FULKERSON, *Flows in Networks*, Princeton, New Jersey, 1962.
8. F. GLOVER, D. KARNEY et D. KLINGMAN, *Implementation and Computational Comparisons of Primal, Dual and Primal-Dual Computer Codes for Minimum Cost Network Flow Problem*, Network, vol. 4, 1974, p. 191-210.
9. P. L. HAMMER et M. SEGAL, *Area Transfers in Local Area* (unpublished work).
10. E. HOBSON, D. L. FLETCHER et W. O. STADLIN, *Network Flow Linear Programming Techniques and their Application to Fuel Scheduling and Contingency Analysis*, I.E.E.E. PAS-103, 1984, p. 1684-1691.
11. T. C. HU, *Minimum Cost Flow in Convex Cost Network*, Naval Res. Logist. Quart. vol. 13, 1966, p. 1-8.
12. P. A. JENSEN et J. W. BARNES, *Network Flow Programming*, Wiley, New York, 1980.
13. J. L. KENNINGTON et R. V. HELGASON, *Algorithms for Network Programming*, Wiley, New York, 1980.
14. D. KLINGMAN, A. NAPIER et J. STUTZ, *NETGEN: a Program for Generating Large Scale Capacited Assignment, Transportation and Minimum Cost Flow Network Problems*, Management Sciences, vol. 20, 1974, p. 814-821.
15. C. L. MONNA et M. SEGAL, *A Primal Algorithm for Finding Minimum Cost Flow in Capacited Networks with Applications*, Bell System Technical Journal, vol. 61, 1982, p. 949-968.
16. J. M. MULVEY, *Testing of Large-Scale Network Optimization Program*, Mathematical Programming, vol. 15, 1978, p. 291-314.