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

M. MINOUX

Résolution des problèmes de multiflots en nombres entiers dans les grands réseaux

Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle, tome 9, n° V3 (1975), p. 21-40

<http://www.numdam.org/item?id=RO_1975__9_3_21_0>

R.A.I.R.O.
(9e année, octobre 1975, V-3, p. 21 à 40)

RÉSOLUTION DES PROBLÈMES DE MULTIFLOTS EN NOMBRES ENTIERS DANS LES GRANDS RÉSEAUX (*)

par M. MINOUX (1)

Résumé. — *Dans le domaine de la gestion et de la planification des réseaux de télécommunications, on se heurte invariablement au problème suivant : trouver un multiflot compatible sur un graphe non orienté dont les arêtes sont munies de capacités (problème de l'admissibilité). Il est résolu, classiquement, par programmation linéaire. Mais, dans ce cas précis, l'algorithme du simplexe s'avère peu approprié : d'une part c'est un outil trop lourd étant donnée la taille des problèmes traités; d'autre part, la solution qu'il fournit n'est généralement pas entière.*

Cet article décrit une méthode heuristique permettant de résoudre des problèmes de multiflots compatibles en nombres entiers sur des graphes de grandes dimensions, en un temps de calcul et en un volume-mémoire réduits sur calculateur électronique. Les résultats de calcul obtenus, tant sur des problèmes réels que sur des exemples de petite taille, indiquent des écarts très faibles (en valeur relative) entre la solution heuristique et la solution optimale théorique. La généralisation au problème du multiflot compatible et de coût minimal est suggérée.

I. INTRODUCTION

Un réseau de télécommunications sera considéré ici comme un graphe non orienté $G = [X, U]$ dont les nœuds $x \in X$ correspondent aux centres de commutation ou de modulation (autocommutateurs, centres de transit...) et dont les arêtes $u \in U$ correspondent aux différentes artères de transmission installées entre paires de nœuds (x, y); chacune de ces artères (arêtes) porte des moyens de transmission (faisceau hertzien, câble coaxial, guide d'onde...) de capacités connues.

D'autre part, pour un couple de nœuds (x, y) on définit la *demande en circuits* entre x et y : c'est le nombre de circuits téléphoniques (supports physiques des communications) devant exister entre x et y. Si K est le nombre

(*) Reçu août 1974.

(1) Centre National d'Études des Télécommunications.

de telles demandes, on notera d^k la valeur (en nombre de circuits) de la demande k relative au couple $k = (x, y)$ $(1 \leqq k \leqq K)$. On observera que ces demandes peuvent être considérées comme des flots sur G, d'origines et de destinations diverses, et possédant la propriété d'être *indépendants* (non miscibles). En effet, les circuits téléphoniques étant le support physique des communications, il n'est pas question de combiner un circuit Lyon-Marseille avec un Paris-Nice : ce sont deux entités bien distinctes (heureusement pour les abonnés !). Mathématiquement, cela implique que les lois des Kirchhoff soient vérifiées séparément pour chaque flot (structure multiflot).

Définissons maintenant ce que l'on appelle le *routage* d'une demande d^k entre x et y : c'est l'opération qui consiste à affecter à la demande k, d^k circuits (au total) pris sur une ou plusieurs chaînes joignant x à y dans G (uni-routage = une seule chaîne; multiroutage = plusieurs chaînes). Ainsi, pour router une demande de 600 circuits entre Paris et Marseille on pourra, par exemple, réserver 600 circuits sur un câble coaxial Paris-Lyon et 600 circuits sur un faisceau hertzien Lyon-Marseille. Évidemment, le choix de l'itinéraire Paris-Lyon-Marseille suppose qu'il existe effectivement 600 circuits au moins sur chaque tronçon. Si les capacités installées apparaissaient insuffisantes, il faudrait envisager de détourner tout ou partie de la demande sur un autre itinéraire, par exemple Paris-Clermont-Lyon-Marseille.

Organiser le routage des circuits existants, à un instant donné, prévoir le routage des nouveaux circuits au fur et à mesure de l'apparition des besoins, telles sont les tâches principales des services techniques chargés de la gestion d'un réseau de télécommunications.

A moyen terme, on s'intéresse au choix et à la prévision des investissements : par exemple, on recherchera sur une période de 4 à 5 ans la somme minimale (actualisée) des investissements nécessaires pour satisfaire la demande en circuits à chaque année de la période.

A court terme, il s'agit de définir un programme de routages pour l'année à venir, en partant de celui de l'année en cours et en tenant compte :

— des nouvelles capacités installées;

— des nouvelles demandes en circuits (pour plus de détails, *cf.* référence [1]).

Dans un cas comme dans l'autre, on se heurte au *problème de l'admissibilité* c'est-à-dire : un réseau étant donné (par sa structure, les capacités de ses artères, et les demandes en circuits point à point) comment savoir s'il existe un routage compatible avec les capacités installées; et comment mettre en évidence un tel routage lorsqu'il existe ?

Étant donnés la taille actuelle du réseau français et surtout son fort taux de croissance, les méthodes manuelles qui étaient employées jusqu'ici deviennent de jour en jour plus impraticables.

Cet article a pour objet de décrire une méthode automatique de résolution, applicable aux réseaux de grandes dimensions, et particulièrement bien adaptée à la programmation sur calculateur électronique.

II. LE PROBLÈME DU MULTIFLOT COMPATIBLE

En Recherche Opérationnelle, le problème de l'admissibilité est bien connu : c'est celui de la recherche d'un multiflot compatible sur un graphe non orienté muni de capacités (*voir* la référence [2], par exemple). Malheureusement, ici, l'analogie avec les problèmes de flots classiques (uniflots) semble n'être que formelle : d'une part, le théorème du flot maximal et de la coupe minimale (référence [16]) ne se généralise pas; d'autre part les solutions fournies par programmation linéaire ne sont généralement pas entières.

Ainsi, il n'existe pas de critère simple permettant de reconnaître si un réseau donné est admissible ou non : si on désire connaître la réponse avec certitude, et si les flots ne sont pas astreints à être entiers, le recours à la programmation linéaire, sous une forme ou sous une autre, est inévitable (références [11], [12], [14]).

Dans le problème qui nous intéresse, cependant, la contrainte d'intégrité est indispensable (il n'est pas concevable de router 1/3 de circuit téléphonique !). Le problème devient alors un programme en nombres entiers de grandes dimensions, pour lequel on sait qu'il n'existe pas de bon algorithme.

Deux approches heuristiques ont été envisagées.

La première utilise la programmation linéaire et procède à un arrondi de la solution optimale continue (ceci est particulièrement justifié du fait que, dans les problèmes que nous traitons, les flots sont toujours des nombres entiers très supérieurs à l'unité).

La seconde est une méthode de relaxation, qui maintient à chaque itération l'intégrité de la solution. Nous verrons pour quelles raisons la première méthode a dû être écartée au profit de la seconde.

1° Le problème en nombres entiers

Désignons par $N = |X|$ le cardinal de l'ensemble X et par $M = |U|$ celui de l'ensemble U.

Pour tout $u \in U$ on notera C_u la capacité (en nombre de circuits) de l'arête u.

On rappelle, d'autre part, que circulent sur le graphe $G = [X, U]$, K flots indépendants de valeurs respectives : $d^1, d^2, \ldots, d^K$.

Pour k variant de 1 à K :

- on notera $s(k)$ et $t(k)$ respectivement l'origine et l'extrémité du flot k [$s(k) \in X$, $t(k) \in X$];
- on désignera par $p(k)$ le nombre de chaînes élémentaires joignant l'origine $s(k)$ à l'extrémité $t(k)$ du flot k;
- pour j variant de 1 à $p(k)$ on appellera P_j^k la j-ième chaîne élémentaire joignant $s(k)$ et $t(k)$.

Dans ce qui suit, on confondra dans une même notation la chaîne P_j^k et le M-vecteur P_j^k de composantes :

$$P_j^k(u) = 1 \quad \text{si} \quad u \in P_j^k \text{ (chaîne)},$$
$$P_j^k(u) = 0 \quad \text{si} \quad u \notin P_j^k \text{ (chaîne)}.$$

Enfin, on notera x_j^k la fraction du flot k empruntant la chaîne P_j^k ($x_j^k \leqq d^k$).

Suivant ces notations, le problème est de trouver des nombres $x_j^k \geqq 0$ vérifiant les contraintes :

a) Contraintes de capacité

$$\sum_{k=1}^{K} \sum_{j=1}^{p(k)} P_j^k(u) . x_j^k \leqq C_u \qquad (u = 1 \ldots M) \tag{1}$$

(elles expriment que la somme des flots passant sur l'arête u doit être inférieure ou égale à la capacité de cette arête).

b) Contraintes de demandes

$$\sum_{j=1}^{p(k)} x_j^k = d^k \qquad (k = 1, \ldots, K) \tag{2}$$

(elles expriment que la somme des flots de $s(k)$ à $t(k)$ doit être égale à la valeur d^k imposée au flot k).

c) Contraintes d'intégrité

x_j^k entiers, pour $k = 1, \ldots, K$ et $j = 1, \ldots, p(k)$.

2° Résolution du problème en continu

On sait que la recherche d'une solution réalisable d'un système d'équations ou d'inéquations linéaires peut toujours se ramener à un problème de programmation linéaire. Pour le cas qui nous intéresse, introduisons dans chaque contrainte de capacité (1) :

— une variable d'écart s_u ($s_u \geqq 0$, $u = 1, \ldots, M$);

— une variable artificielle y_u ($y_u \geqq 0$, $u = 1, \ldots, M$) et écrivons (1) sous la forme

$$\sum_{k=1}^{K} \sum_{j=1}^{p(k)} P_j^k(u).x_j^k + s_u - y_u = C_u. \qquad (1')$$

La variable s_u représente le nombre de circuits excédentaires sur l'arête u, et la variable y_u le nombre de circuits manquants. Un routage pour lequel la somme des variables artificielles est nulle constitue donc une solution (en continu) du problème de l'admissibilité.

Pour déterminer une telle solution, on voit qu'il suffit de rechercher une solution optimale du programme linéaire suivant (PL) :

$$\text{(PL)} \quad \begin{cases} \text{minimiser : } z = \sum_{u=1}^{M} y_u \\ \text{sous les contraintes :} \\ \sum_{k=1}^{K} \sum_{j=1}^{p(k)} P_j^k(u).x_j^k + s_u - y_u = C_u, \qquad (1') \\ \sum_{j=1}^{p(k)} x_j^k = d^k, \qquad (2) \\ x_j^k \geqq 0, \quad s_u \geqq 0, \quad y_u \geqq 0. \end{cases}$$

Évidemment, si la solution optimale z^* de (PL) est strictement positive, c'est que le problème de l'admissibilité n'a pas de solution (on dit aussi que le réseau donné n'est pas admissible).

Le programme (PL) possède une forte structure : la plus grande partie de la matrice des contraintes est en effet formée de contraintes de choix multiples :

$$\sum_{j=1}^{p(k)} x_j^k = d^k,$$

d'où une structure classique « en escalier ».

On sait que l'on peut appliquer à ce genre de problème :

— soit l'algorithme de décomposition de Dantzig et Wolfe (références [3] et [4], p. 189-197);

— soit l'algorithme de Dantzig et Van Slyke (*Generalized upper bounding*, références [5] et [6]).

La théorie n'est donc pas démunie, et l'on pourrait songer ici à lui emprunter une de ses méthodes.

L'algorithme de décomposition a été expérimenté (référence [7]) et on a pu vérifier, une fois de plus, sur des problèmes réels ses deux principaux inconvénients :

- convergence très lente au voisinage de l'optimum;
- fractionnement extrême de la solution fournie.

La seconde méthode, quant à elle, évite ces écueils en travaillant directement sur la matrice des contraintes de (PL); d'autre part, en exploitant la structure « en escalier » de (PL) elle permet de réduire la dimension de la base à une taille raisonnable, de l'ordre de $M \times M$ (références [6] et [14]). Mais des réseaux à 500 arêtes sont fréquents en pratique, et le stockage, la mise à jour et l'inversion de matrices 500×500 sont des opérations coûteuses en temps calcul et en nombre de mots-mémoire.

3° Vers une solution heuristique

Ces arguments de temps et d'encombrement n'ont pas été les seuls pour nous inciter à abandonner l'approche utilisant la programmation linéaire. Comme la solution optimale continue n'est généralement pas entière, il faut encore prévoir une procédure d'arrondi, laquelle, vue la taille des problèmes, ne peut être que de nature heuristique. D'autre part, comme les données (capacités, valeurs des flots) sont rarement précises à un circuit près, il semble bien inutile d'employer un algorithme donnant la solution optimale continue avec un degré de finesse extrême.

III. UNE MÉTHODE HEURISTIQUE POUR LE PROBLÈME DU MULTIFLOT COMPATIBLE

Comme dans le programme linéaire (PL), l'objectif est de minimiser la fonction z (somme des circuits manquants).

L'idée est la suivante : minimiser z par relaxations successives en recherchant à chaque étape le meilleur routage pour une demande d^k ($1 \leqq k \leqq K$), les routages des $K-1$ autres demandes ($d^1, d^2, \ldots, d^{k-1}, d^{k+1}, \ldots, d^K$) étant fixés. Il est commode, pour la clarté de l'exposé, de se restreindre tout d'abord à la recherche d'un *uniroutage* donnant à z une valeur minimale. (On rappelle qu'un uniroutage est un routage dans lequel chaque flot s'écoule sur une chaîne unique.)

Si on se donne, pour chaque flot k, un routage $R^k \in \{ P_j^k \}$ choisi parmi les chaînes P_j^k il est facile de calculer la valeur de z correspondante.

En définissant les capacités résiduelles $\bar{C}_u$ des arêtes après routage des flots 1 à K sur les chaînes $R^1, R^2, \ldots, R^K$ par :

$$\bar{C}_u = C_u - \sum_{k=1}^{K} R^k(u).d^k,$$

on peut écrire :

$$z = - \sum_{u/\bar{C}_u<0} \bar{C}_u.$$

De la sorte, z peut être considérée comme une fonction $\Phi(R^1, R^2, \ldots, R^K)$ des routages.

La recherche du minimum absolu de Φ est un problème très combinatoire (même pour un graphe de taille modérée, il existe des milliers de chaînes entre deux nœuds x et y donnés). Par contre, il est facile de trouver le minimum de chacune des différentes restriction Φ^k de Φ à un flot k unique.

Supposons, par exemple, que l'on désire rechercher le minimum de $\Phi^1(R^1) = \Phi(R^1, R_0^2, R_0^3, \ldots, R_0^K)$ pour $R_0^2, R_0^3, \ldots, R_0^K$ fixés.

Les capacités résiduelles $\bar{C}_u$ après routage des flots 2 à K sont :

$$\bar{C}_u = C_u - \sum_{k=2}^{K} R_0^k(u).d^k \tag{3}$$

et la valeur de z correspondante est

$$z_0 = - \sum_{u/\bar{C}_u<0} \bar{C}_u.$$

Cherchons maintenant le meilleur routage possible pour le flot 1. Alors, pour une arête $u \in U$ quelconque de G :

- si la capacité résiduelle $\bar{C}_u$ calculée en (3) est telle que $\bar{C}_u - d^1 \geqq 0$, la valeur z_0 de z n'est pas augmentée par le passage du flot 1 sur l'arc u;
- si par contre $0 \leqq \bar{C}_u < d^1$ alors la valeur z_0 sera augmentée de la quantité : $d^1 - \bar{C}_u$;
- enfin, si $C_u < 0$, la valeur z_0 sera augmentée de la quantité d^1.

Au total, le routage du flot 1 sur la chaîne P_j^1 ($1 \leqq j \leqq p(1)$) donne à z la valeur :

$$z = z_0 + \sum_{u \in P_j^1} \gamma_u,$$

avec :

$$\begin{cases} \gamma_u = 0 & \text{si } \bar{C}_u \geqq d^1, \\ \gamma_u = d^1 - \bar{C}_u & \text{si } 0 \leqq \bar{C}_u < d^1, \\ \gamma_u = d^1 & \text{si } \bar{C}_u < 0. \end{cases}$$

Par suite, le meilleur routage du flot 1 est celui qui minimise : $\sum_{u \in P_j^1} \gamma_u$: *c'est donc la plus courte chaîne de* $s(1)$ *à* $t(1)$, *les arêtes* $u \in U$ *de* G *étant munies des longueurs* γ_u.

Ainsi, minimiser une restriction Φ^k de Φ à un flot k donné, se ramène à la détermination d'une plus courte chaîne entre $s(k)$ et $t(k)$ dans G. La recherche du minimum de z pourra donc se faire, très simplement, par une méthode de relaxations successives dans laquelle on minimise à chaque étape une restriction Φ^k de Φ. D'une façon plus détaillée, la procédure adoptée est la suivante :

1) Partir d'un routage initial (uniroutage) pour chacun des flots $k = 1, \ldots, K$.

Numéro de l'itération : $i = 0$.

2) $i = i + 1$.

3) Pour chaque flot $k = 1, \ldots, K$.

- calcul des capacités résiduelles $\bar{C}_u$ $(u \in U)$;
- calcul des longueurs γ_u $(u \in U)$;
- le meilleur routage du flot k est la plus courte chaîne de $s(k)$ à $t(k)$ en fonction des longueurs γ_u.

4) Si tous les flots ont été examinés ($k = K$) aller en 5). Sinon, retourner en 3).

5) Si la valeur de z a été améliorée par rapport à l'itération précédente, retourner en 2).

Sinon, un optimum (local) de z a été atteint.

Fin de la procédure.

La convergence de la méthode est facile à démontrer : les capacités et les valeurs des flots étant des nombres entiers, la fonction z est entière à chaque itération. Il suffit alors de remarquer que z est monotone, strictement décroissante au cours des itérations (tant que le critère d'arrêt n'a pas été atteint) et bornée inférieurement par 0. (La solution obtenue n'est évidemment qu'un optimum local de z.)

Signalons enfin que, lorsqu'on cherche à résoudre un problème de multiflot compatible classique (sans contrainte d'intégrité), une telle heuristique est extrêmement intéressante : en effet, en fournissant une très bonne solution de départ à la méthode du simplex, le nombre d'itérations nécessaires ensuite pour obtenir l'optimalité est considérablement diminué.

IV. UN EXEMPLE NUMÉRIQUE

Afin de rendre plus concrets les développements qui précèdent, nous croyons intéressant d'illustrer le fonctionnement de notre heuristique sur un petit exemple.

Le graphe G de la figure 1 comprend 4 nœuds et 5 arcs de capacités respectives :

Arc n°	Origine, extrémité	Capacité
1	1,2	$C_1 = 8$
2	1,3	$C_2 = 52$
3	2,3	$C_3 = 38$
4	2,4	$C_4 = 32$
5	3,4	$C_5 = 10$

D'autre part, nous supposerons que 2 flots ($K = 2$) doivent circuler sur ce graphe, de valeurs respectives :

Flot n°	Origine, extrémité	Valeur du flot
1	1,4	$d^1 = 35$
2	2,3	$d^2 = 20$

Le routage de départ choisi (arbitrairement) est le suivant :

- le flot 1 routé sur la chaîne ¦ 1, 4 ¦;
- le flot 2 routé sur la chaîne ¦ 3 ¦

pour lequel la fonction z prend la valeur $z = 30$.

Itération 1

a) *Examen du flot* 1

(origine 1; extrémité = 4; valeur 35)

En revenant à une situation où, seul, le flot 2 est routé sur la chaîne $\{3\}$ on obtient les capacités résiduelles :

$$\bar{c}_1 = 8; \quad \bar{c}_2 = 52; \quad \bar{c}_3 = 18; \quad \bar{c}_4 = 32; \quad \bar{c}_5 = 10$$

(voir *fig.* 2 *a*).

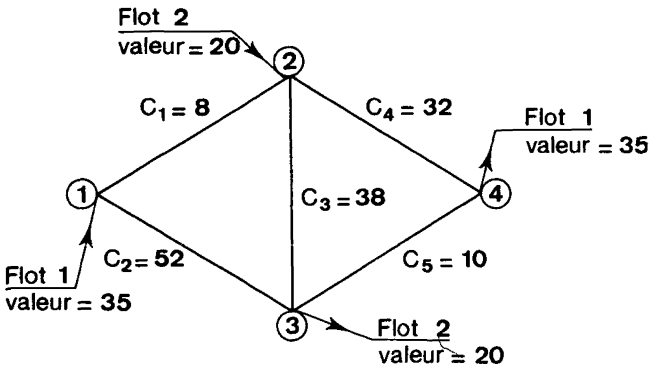


Figure 1

Recherche d'un biflot compatible ($K = 2$) sur un graphe à quatre nœuds. Le flot 1 a pour origine le nœud 1, extrémité le nœud 4 et pour valeur $d^1 = 35$. Le flot 2 a pour origine 2, extrémité 3 et pour valeur $d^2 = 20$. On a indiqué également, la capacité des arêtes en regard de chacune d'elles.

Les longueurs γ_u sont alors définies par :

$$\begin{cases} \gamma_1 = 35-8 = 27, \\ \gamma_2 = 0, \\ \gamma_3 = 35-18 = 17, \\ \gamma_4 = 35-32 = 3, \\ \gamma_5 = 35-10 = 25 \end{cases}$$

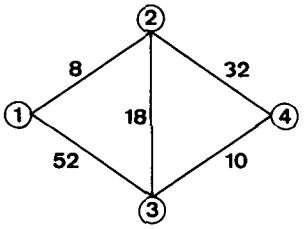
(voir *fig.* 2 *b*).

La recherche de la plus courte chaîne de 1 à 4 donne alors : $\{2, 3, 4\}$, de longueur totale 20.

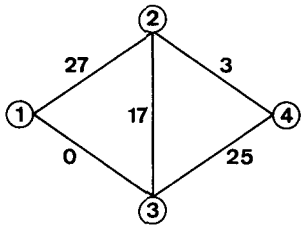
La fonction z passe, avec ce nouveau routage pour le flot 1, de $z = 30$ à $z = 20$.

b) *Examen du flot* 2

(origine 2; extrémité 3; valeur 20)



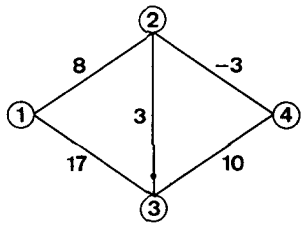
(*a*) Capacités résiduelles obtenues après routage du flot 2 sur l'arête { 3 }.



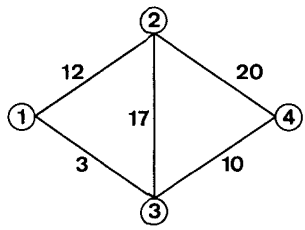
(*b*) Détermination des longueurs γ sur toutes les arêtes du graphe. La plus courte chaîne entre 1 et 4 en fonction de ces longueurs est la chaîne { 2, 3, 4 } de longueur 20.

Figure 2

Examen du flot 1 à l'itération 1



(*a*) Capacités résiduelles après routage du flot 1 sur la chaîne { 2, 3, 4 }.



(*b*) Détermination des longueurs γ. La plus courte chaîne entre 2 et 3 est { 1, 2 } de longueur 15.

Figure 3

Examen du flot 2 à l'itération 1

Lorsque le flot 1 est seul routé sur la chaîne { 2, 3, 4 } les capacités résiduelles des arcs sont les suivantes :

$$\bar{c}_1 = 8; \quad \bar{c}_2 = 17; \quad \bar{c}_3 = 3; \quad \bar{c}_4 = -3; \quad \bar{c}_5 = 10.$$

D'où $z_0 = +3$ (*fig.* 3 *a*).

Les longueurs γ_u sont alors :

$$\begin{cases} \gamma_1 = 20-8 = 12, \\ \gamma_2 = 20-17 = 3, \\ \gamma_3 = 20-3 = 17, \\ \gamma_4 = 20, \\ \gamma_5 = 20-10 = 10. \end{cases}$$

La plus courte chaîne de 2 à 3 est alors : ¦ 1, 2 ¦, de longueur 15 (*fig.* 3 *b*) et la valeur de z devient : $z = z_0 + 15 = 18$.

Itération 2

Nous laissons au lecteur le soin de vérifier que l'on ne peut plus améliorer maintenant la valeur de z. Il constatera que la chaîne ¦ 2, 3, 4 ¦ reste optimale pour le flot 1 et la chaîne ¦ 1, 2 ¦ reste optimale pour le flot 2.

La procédure s'arrête donc et la meilleure valeur de z obtenue est $z = 18$.

V. FRACTIONNEMENT DES FLOTS ET ROUTAGE DE DÉPART

1° Fractionnement

L'examen des résultats obtenus, au paragraphe précédent, sur l'exemple de la figure 1 montre que la restriction à la recherche d'un *uniroutage admissible* est une sérieuse limitation.

En effet, dans la solution obtenue certaines arêtes ont une capacité mal utilisée tandis que certaines autres sont beaucoup trop chargées.

Au lieu de faire passer la totalité du flot 1 sur la chaîne ¦ 2, 3, 4 ¦ on voit que l'on pourrait en faire passer 10 unités sur la chaîne ¦ 2, 5 ¦. Ceci aurait pour effet de libérer 10 unités de capacité sur l'arête ¦ 3 ¦.

En décomposant, encore, le flot 2 en deux fractions de 8 et 12 unités respectivement, on arrive à satisfaire toutes les contraintes de capacité en faisant passer la deuxième fraction sur l'arête ¦ 3 ¦. On obtient alors un *multiroutage admissible.*

Ainsi, le fractionnement conduit à une meilleure utilisation des capacités, et on peut s'attendre par conséquent à obtenir de meilleurs valeurs de z (*cf.* les résultats numériques du paragraphe VI).

Par ailleurs, dans la méthode heuristique du paragraphe III, on remarquera que ceci n'entraîne aucune modification de principe : il suffit de traiter chaque fraction comme un flot indivisible et indépendant des autres fractions.

Deux types de fractionnement ont été expérimentés :

Le premier décompose un flot de valeur n en n flots de valeur 1 (fractionnement en parties égales). C'est celui qui fournit les résultats les plus fins, mais au prix d'un temps calcul accru.

Le second est le fractionnement en puissances de deux : il décompose un flot de valeur 25, par exemple, en 5 flots seulement de valeurs respectives : 1, 2, 4, 8 et 10 (la dernière fraction est le résidu modulo $2^4 - 1$).

Remarquons enfin que, dans un cas comme dans l'autre, on a le choix de l'unité de base. Dans les problèmes de télécommunications, on en profitera pour prendre comme unité de base les unités naturelles de regroupement des circuits : le groupe primaire (12 circuits) le groupe secondaire (60 circuits), etc.

2° Routage de départ

Ce point n'est pas très critique, mais on a évidemment intérêt à choisir un routage de départ donnant à la fonction z une valeur aussi faible que possible.

La méthode que nous avons employée consiste à examiner les flots les uns après les autres, en commençant par ceux de plus grande valeur. Chaque flot k est alors routé sur la plus courte chaîne (en nombre d'arêtes) joignant $s(k)$ à $t(k)$. Lorsqu'il existe deux chaînes de même longueur, la chaîne de capacité résiduelle maximale est retenue (la capacité d'une chaîne est ici définie comme le minimum des capacités des arêtes qui la constituent). Avant de procéder à l'examen du flot suivant, les capacités résiduelles des arêtes du graphe G sont mises à jour.

Il faut noter que, dans la pratique, le routage de départ fait souvent partie des données du problème. C'est le cas, par exemple, pour la gestion annuelle dans un réseau téléphonique interurbain (*cf.* introduction).

VI. RÉSULTATS OBTENUS SUR DES EXEMPLES PRATIQUES

L'heuristique a été programmée en FORTRAN sur calculateur BGE 6000, et, depuis juin 1973, a fait l'objet de nombreuses expériences. Entre autres, on a procédé à un certain nombre d'essais « en vraie grandeur » visant à déterminer un plan annuel du routage sur le réseau interurbain français. Dans

chacun de ces exemples le graphe G considéré comportait :

$$N = 87 \text{ noeuds},$$
$$M = 150 \text{ arêtes},$$

et était parcouru par $K = 365$ flots d'origines et destinations diverses. Donnons, (pour le réseau de 1973) les principales caractéristiques des résultats fournis :

- fractionnement (en puissances de deux) avec unité de base de 60 circuits conduisant à un nombre total de 873 fractions;

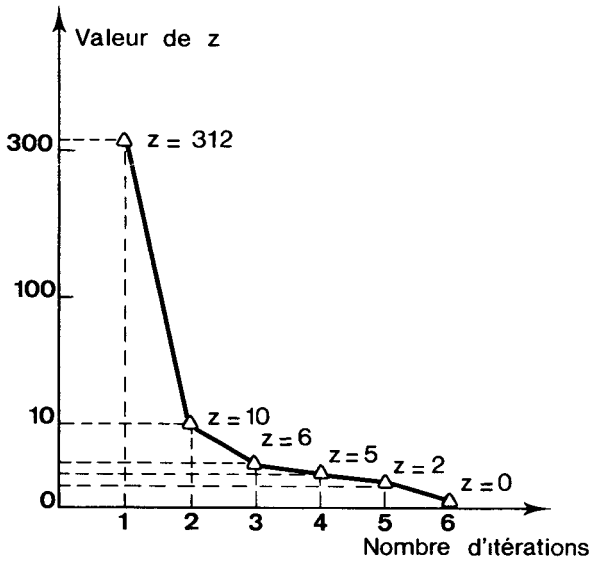


Figure 4

Décroissance de la fonction z (somme des capacités manquantes) au cours des itérations sur un exemple à 87 nœuds et 150 arêtes (réseau interurbain 1973).

- nombre d'itérations pour obtenir l'admissibilité = 6. On trouvera à la figure 4 la courbe de décroissance de la fonction z (somme des capacités manquantes) au cours des itérations. On remarquera que plus des 95 % du gain sont obtenus dès la première itération;
- temps d'exécution sur BGE 6000 : 30 secondes (lectures et écritures comprises);
- occupation en mémoire centrale : 25 000 mots;
- pourcentage des demandes routées au plus court chemin (en nombre d'arêtes) dans la solution : 80 %;

• nombre de flots : uniroutés : 327
biroutés : 30
triroutés : 8
total : 365

Indépendamment des résultats obtenus sur quelques problèmes réels de ce type, il était intéressant d'étudier de façon systématique le comportement de l'heuristique et sa sensibilité à certains paramètres. Pour cela, une série de 5 problèmes-témoins a été construite sur un graphe G comprenant 12 nœuds et 25 arêtes. Dans chacun de ces cas seules diffèrent les valeurs des capacités attribuées aux arêtes; la structure du graphe G et les demandes point à point restant les mêmes. Par ailleurs, ces cinq problèmes correspondent tous à des réseaux dimensionnés « au plus juste », c'est-à-dire qu'ils appartiennent à une catégorie de problèmes particulièrement difficiles à résoudre.

Les expériences effectuées avaient pour objet :

a) l'étude de la rapidité de convergence;

b) l'étude de l'influence du fractionnement et de l'ordre d'examen des demandes sur la qualité du résultat fourni (valeur de z).

Commentons brièvement, pour chacun de ces points, les résultats obtenus.

• Les résultats concernant l'influence du fractionnement sur la rapidité de convergence sont consignés dans le tableau I. On remarquera que le nombre

TABLEAU I

Le nombre d'itérations jusqu'à convergence de la méthode heuristique en fonction de l'unité de base choisie pour le fractionnement (type de fractionnement : en puissances de 2).

Numéro du problème →	1	2	3	4	5
Pas de fonctionnement	3	3	2	2	4
Unité de base : 300	6	4	3	5	5
Unité de base : 60	8	4	2	6	7
Unité de base : 12	6	3	8	9	5

d'itérations requises croît, généralement, avec le nombre de fractions induites, et que dans tous les cas ce nombre reste faible (inférieur à 10).

• Pour ce qui concerne la qualité du résultat fourni (valeur finale de z) on se reportera au tableau II où l'avantage du fractionnement apparaît très nettement.

TABLEAU II

Étude de la valeur finale de la fonction z (somme des capacités manquantes) pour différentes valeurs de l'unité de base choisie. On a indiqué également les valeurs minimales de z obtenues, dans chaque cas, par programmation linéaire. Avec le fractionnement, on observe que les plus grands écarts en valeurs relatives (ramenés à la somme totale des capacités des arêtes) sont inférieurs à 1 %.

Numéro du problème →	1	2	3	4	5
Somme totale des capacités des arêtes →	172000	227000	193000	124000	136000
Pas de fonctionnement	587	4704	3335	2525	9615
Unité de base : 300	0	1	66	309	24
Unité de base : 60	0	218	0	655	569
Unité de base : 12	478	0	48	309	72
Résultats obtenus par programmation linéaire :	$Z^* = 0$ (admissib.)	$Z^* = 0$ (admissib.)	$Z^* = 0$ (admissib.)	$Z^* = 178{,}3$ (non adm.)	$Z^* = 6{,}1$ (non adm.)

D'autre part, le tableau II permet de comparer les résultats fournis par notre heuristique, d'une part, et les solutions optimales continues obtenues par programmation linéaire d'autre part. On notera d'abord que la réponse au problème de l'admissibilité est souvent correcte ainsi par exemple sur les problèmes 1 et 3 pour une unité de fractionnement égale à 60. On notera ensuite que lorsque cette réponse n'est pas exacte, les différences en valeur relative entre les valeurs finales de z fournie par l'heuristique, et l'optimum théorique z^* en continu sont extrêmement faibles (< 1 %). Ceci montre que la solution heuristique est très proche de l'optimum en nombres entiers.

TABLEAU III

Étude de l'influence de l'ordre d'examen des flots sur la valeur finale de z (somme des capacités manquantes). Les différences sont ramenées en pourcentage à la somme totale des capacités des arêtes du graphe G (résultats pour un fractionnement en puissances de 2 et multiples de 60).

Numéro du problème →	1	2	3	4	5
Somme des capacités des arêtes →	172000	227000	193000	124000	136000
Ordre des valeurs croissantes	775	0	16	309	23
Ordre des valeurs décroissantes	0	218	0	655	569
Différence en valeur absolue	775	218	16	346	546
Différence en pourcentage	0,4 %	0,1 %	0,01 %	0,3 %	0,3 %

• Enfin, si l'on se réfère au tableau III, on pourra se convaincre que l'influence de l'ordre d'examen des différents flots à chaque itération n'est pas critique. Tantôt les résultats sont favorables à l'ordre des valeurs croissantes, tantôt c'est l'ordre des valeurs décroissantes qui l'emporte. Dans tous les cas, les différences en valeur relative sont très faibles (inférieures à 0,5 %).

VII. EXTENSION AUX PROBLÈMES DE MULTIFLOTS COMPATIBLES AVEC COÛTS

1° Le principe heuristique que nous avons présenté se généralise sans difficulté au cas où les arêtes du réseau cesseraient d'être équivalentes; c'est-à-dire où l'ont attacherait plus d'importance à vérifier les contraintes de capacité sur certaines arêtes plutôt que sur certaines autres. Ceci peut être fait en affectant à chaque arête $u \in U$ un « poids » π_u et en recherchant le minimum de la somme des capacités manquantes pondérée par les π_u; autrement dit, le

minimum de la fonction

$$z' = - \sum_{u/\bar{c}_u < 0} \pi_u . \bar{c}_u .$$

Pour minimiser z', il suffit de rechercher successivement, pour chaque flot k ($k = 1, \ldots, K$) une plus courte chaîne en affectant à chaque arête $u \in U$ de G une longueur $\pi_u . \gamma_u$, où les nombres γ_u sont ceux qui ont été définis au paragraphe III.

2° Une autre extension intéressante mérite d'être mentionnée. On se donne pour chaque arête $u \in U$ le coût de passage $w_u > 0$ d'une unité de flot sur l'arête u. Le problème est alors de trouver un routage admissible, et de coût total minimal (problème du multiflot compatible de coût minimal); soit de minimiser :

$$\sum_{u \in U} w_u . x_u$$

sur l'ensemble des routages *admissibles* (x_u est la quantité de flot sur l'arête u). Pour ce faire, on décidera de pénaliser une capacité manquante ($\bar{C}_u < 0$) d'une certaine quantité $w_u + \lambda$ (où λ est un paramètre positif), et on minimisera la fonction :

$$z'' = \sum_{u/x_u \leqq c_u} w_u . x_u + \sum_{u/x_u > c_u} w_u . x_u + \lambda . (x_u - c_u).$$

Il suffira alors de rechercher, à chaque examen de flot, une plus courte chaîne en fonction de nouvelles longueurs γ_u. Pour le flot $k = 1$ par exemple, et en désignant toujours par $\bar{C}_u$ ($u \in U$) les capacités résiduelles des arêtes après routage des flots 2 à K, celles-ci doivent être définies de la façon suivante :

- $\gamma_u = w_u . d^1$ si $\bar{C}_u \geqq d^1$;
- $\gamma_u = w_u . d^1 + \lambda . (d^1 - \bar{C}_u)$ si $0 \leqq \bar{C}_u < d^1$;
- $\gamma_u = (w_u . + \lambda) . d^1$ si $\bar{C}_u < 0$.

Ce procédé s'apparente tout à fait à une *phase* 1 *paramétrée* dans la méthode du simplex (référence [3]) où l'on minimise une combinaison linéaire de la fonction économique et de la fonction artificielle.

Remarquons, ici encore, que si l'on cherche à résoudre un problème de multiflot de coût minimum classique (c'est-à-dire sans contrainte d'intégrité) cette heuristique peut être avantageusement utilisée pour fournir une bonne solution de départ à l'algorithme du simplex (et, par conséquent, pour en accélérer la convergence).

VIII. CONCLUSION

La recherche d'un multiflot compatible, en nombres entiers, constitue un problème fondamental dans la gestion et la planification des réseaux de télécommunications.

La méthode de résolution qui a été exposée permet de traiter des réseaux comportant une centaine de nœuds et plusieurs centaines d'artères en un temps de calcul de l'ordre de la minute, sur calculateur électronique (type BGE 6000). Les expériences de calcul effectuées sur de petits exemples, permettent d'affirmer que les résultats fournis s'écartent très peu (en valeur relative) des résultats obtenus par programmation linéaire (en relaxant les contraintes d'intégrité) et par conséquent, sont proches de l'optimalité en nombres entiers.

Les applications de cette heuristique au domaine des télécommunications sont, d'ores et déjà, très nombreuses. Citons encore :

— l'automatisation des opérations de reroutage des circuits lors d'une panne sur une artère du réseau;

— la gestion de la sécurité à moyen terme.

Mais nous pensons aussi qu'elle pourrait trouver des applications intéressantes dans d'autres domaines où se posent des problèmes de multiflots compatibles ou de coût minimal : réseaux de calculateurs, transports routiers, réseaux de téléinformatique (avec maillage), etc.

REMERCIEMENTS

Je remercie le comité de lecture pour ses suggestions concernant la mise en forme de cet article.

BIBLIOGRAPHIE

[1] M. MINOUX, *Planification à court et à moyen terme d'un réseau de télécommunications*, Annal. Télécomm., 29, nos 5-6, novembre-décembre 1974.

[2] J. A. TOMLIN, *Minimum cost multicommodity networks flows*, Operations Research, 14 (1), février 1966, p. 45-51.

[3] G. B. DANTZIG, *Linear programming and extensions*, Princeton University Press, 1963.

[4] T. C. HU, *Integer Programming and network flows*, Addison-Wesley, 1969.

[5] L. S. LASDON, *Optimization theory for large systems*, Macmillan Series for Operations Research, 1970.

[6] G. B. DANTZIG et R. M. VAN-SLYKE, *Generalized upper bounding techniques*, *J. Computer System Science*, 1, 1967, p. 213-226.

[7] M. MINOUX, *Le problème de l'admissibilité d'un réseau parcouru par des flots multiples*, Note technique ITD/CES/59, CNET, mars 1972.

[8] K. ONAGA et O. KAKUSHO, *On feasibility conditions of multicommodity flows in networks, I. E. E. E. Transactions on circuit theory*, 18, n° 4, juillet 1971, p. 425-429.

[9] L. FRATTA, M. GERLA et L. KLEINROCK, *The flow deviation method: an approach to store and forward communication network design*, Networks, 3, n° 3, 1973, p. 97-133.

[10] H. FRANK et W. CHOU, *Routing in computer networks*, Networks, 1, n° 2, 1971, p. 99-112.

[11] L. R. FORD et D. R. FULKERSON, *A suggested computation for maximal multi-commodity network flows*, Management Science, 5, 1958, p. 97-101.

[12] R. SAIGAL, *Multicommodity flows in directed networks*, University of California, Berkeley, Ph. D., 1968.

[13] W. S. JEWELL, *Multicommodity network solutions* in *Théorie des Graphes*, Rome, 1966, Dunod, Paris, 1967.

[14] J. K. HARTMAN et L. S. LASDON, *A generalized upper bounding algorithm for multicommodity network flow problems*, Networks, 1, n° 4, 1971, p. 333-354.

[15] T. C. HU, *Multicommodity network flows*, J. Orsa, 11 (3) ,1963, p. 344-360.

[16] L. R. FORD et D. R. FULKERSON, *Flows in networks*, Princeton University Press, Princeton, 1962.