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

YVES TABOURIER

Un algorithme pour le problème d'affectation

Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle, tome 6, n° V3 (1972), p. 3-15

<http://www.numdam.org/item?id=RO_1972__6_3_3_0>

R.A.I.R.O.
(6e année, novembre 1972, V-3, p. 3 à 16)

UN ALGORITHME POUR LE PROBLEME D'AFFECTATION

par Yves TABOURIER (1)

Résumé. — *On propose pour le problème d'affectation un algorithme plus simple et généralement moins coûteux en mémoire que les algorithmes usuels. Conçu spécialement pour le cas où le tableau des coûts est rectangulaire et/ou peu rempli, cet algorithme semble néanmoins capable de venir battre sur leur propre terrain les techniques spécifiques du problème à tableau carré et plein (Méthode Hongroise, Algorithme de Ford-Fulkerson).*

INTRODUCTION

Le problème dit « problème d'affectation » est tout à fait classique et apparemment bien résolu. Il s'agit, sur un graphe biparti valué, de trouver un couplage maximum de valeur extremum. La « Méthode Hongroise » [1], [5], longtemps employée, tend à céder la place à la méthode voisine de Ford et Fulkerson [3]. On peut aussi utiliser l'algorithme primal de Balinski et Gomory [2].

Malheureusement, ces techniques supposent qu'un couplage maximum sature tous les sommets du graphe, ce qui n'est possible que si le tableau des coûts est carré, et ce qui n'est certain que si toutes les cases de ce tableau sont admissibles. Il en résulte que si le tableau initial est, par exemple, une matrice de 20×100 remplie à 30 %, il n'en faudra pas moins gérer un système de 10 000 coûts (100×100) au lieu des 600 réellement donnés ($20 \times 100 \times 0{,}30$).

Beaucoup d'algorithmes de théorie des flots sont exempts de ce défaut. Or personne, à notre connaissance, n'a jamais cherché à les adapter pour tirer tout le parti possible de la structure très particulière du problème d'affectation, afin de concurrencer de façon sérieuse les méthodes usuelles.

C'est ce que nous avons cherché à obtenir en construisant l'algorithme proposé ici. La première partie comporte la description de l'algorithme, avec

(1) Direction Scientifique du Groupe Metra.

la définition de tous les concepts de théorie des graphes nécessaires à l'exposé. La seconde partie est la justification théorique de la technique, et suppose du lecteur la connaissance de la théorie des flots. Une dernière partie présente les variantes et les essais. Ce sont ces essais qui ont montré, à notre surprise, que le nouvel algorithme venait battre les techniques usuelles sur leur terrain. Les bases du travail présenté ici ont été publiées dans [8].

I. PRESENTATION DE L'ALGORITHME

1. Définitions et notations

Soient X et Y deux ensembles finis disjoints : $X \cap Y = \varnothing$.

Soit $\Gamma \subset X \times Y$: $G = (X, Y, \Gamma)$ est appelé *un graphe biparti fini.*

Nous poserons :

$$\begin{cases} \forall x \in X, & \Gamma x = \{ y \in Y | (x, y) \in \Gamma \} \\ \forall y \in Y, & \Gamma^{-} y = \{ x \in X | (x, y) \in \Gamma \} \end{cases}$$

$\gamma = (x, y) \in \Gamma$ sera appelée une *arête* de G. $x \in X$ et $y \in Y$ seront appelés des *sommets* de G, le premier de la *classe* X, le second de la *classe* Y. Comme $X \cap Y = \varnothing$, *nous confondrons* sans risque d'ambiguïté les êtres (x, y) et $\{ x, y \}$ lorsque $(x, y) \in \Gamma$.

$W \subset \Gamma$ est *un couplage* de G si, et seulement si,

$$\gamma \ , \quad \gamma' \in W \ , \quad \gamma' \neq \gamma \Rightarrow \gamma \cap \gamma' = \varnothing.$$

La taille de W sera sa cardinalité $|W|$. Un couplage de taille maximum sera dit *maximum.*

On se donne maintenant une application v de Γ dans R :

$$v : \Gamma \to R, \text{ définissant les } \textit{valeurs des arêtes.}$$

Nous appellerons valeur du couplage W le nombre :

$$V(W) = \sum_{\gamma \in W} v(\gamma)$$

2. Énoncé du problème

Étant donné le graphe biparti fini $G = (X, Y, \Gamma)$ et l'application $v : \Gamma \to R$, trouver le couplage maximum de G qui rende extremum la fonction

$$V(W) = \sum_{\gamma \in W} v(\gamma).$$

3. Définitions complémentaires

a) *Sommets saturés et insaturés*

Un sommet a de $X \cup Y$ est dit insaturé dans W si $\nexists \gamma \in W \mid a \in \gamma$. Soit en revanche $(x, y) \in W$: x est dit saturé par y et y par x.

b) *Chaîne alternée (CA) ; CA saturante ; CA tronquée*

On appelle *chaîne alternée* (*CA*) dans G sur W une suite d'arêtes prises alternativement hors de W et dans W, chaque arête ayant une extrémité en commun avec la précédente et l'autre avec la suivante. Une chaîne alternée $\mu = (\gamma_1 \dots \gamma_p)$ sera dite *saturante* (*CAS*) si $\gamma_1 \notin W$, $\gamma_p \notin W$, et si le représentant de classe X de γ_1 et le représentant de classe Y de γ_p sont insaturés. Une chaîne $\mu = (\gamma_1 \dots \gamma_p)$ telle que $\gamma_1 \notin W$, $\gamma_p \in W$, le représentant de classe X dans γ_1 étant insaturé, sera appelée *chaîne alternée tronquée* (*CAT*) [1].

c) *Longueur d'une arête. Longueur d'une CA*

La longueur de l'arête (x, y) sera définie par :

$$\begin{cases} l(x, y) = v(x, y) & \text{si} \quad (x, y) \in \Gamma - W \\ l(x, y) = -v(x, y) & \text{si} \quad (x, y) \in W. \end{cases}$$

Si la suite d'arêtes $\mu = (\gamma_1 \gamma_2 \dots \gamma_p)$ forme une chaîne alternée, sa longueur $L(\mu)$ sera définie par $L(\mu) = \sum_{i=1}^{p} l(\gamma_i)$; si $\mu = \varnothing$, $L(\mu) = 0$.

d) *Inversion d'une CAS ou d'une CAT*

Soit $\mu = (\gamma_1 \dots \gamma_k \dots \gamma_p)$ une *CAS* ou une *CAT*. Inverser μ, c'est remplacer W par $[W \cup \mu - (W \cap \mu)]$, c'est-à-dire échanger le long de μ l'appartenance et la non-appartenance à W.

L'effet de l'inversion sur la valeur du couplage est de l'augmenter de $L(\mu)$. Si μ est une *CAS*, la taille du couplage augmente de 1 par saturation du sommet de classe X de la première arête. Si μ est une *CAT*, cette saturation est compensée par la désaturation du représentant de classe X de la dernière arête et la taille de X n'augmente pas. Si $\mu = \varnothing$, W est inchangé.

4. Algorithme proposé

a) *Algorithme principal*

Phase préparatoire. Poser $W = \varnothing$.

Phase A

---

(1) D'autre part la *chaîne vide* est une CAT.

Pour chaque $x \in X$, pris successivement :

— chercher $y_0 \in \Gamma x$ tel que

$$\begin{cases} v(x, y_0) = \underset{y \in \Gamma x}{Ext}\, v(x, y) \\ y_0 \text{ insaturé dans } W. \end{cases}$$

— Si y_0 a pu être trouvé, ranger (x, y_0) dans W. Passer dans tous les cas au sommet $x \in X$ suivant.

Phase B (à n'exécuter que si la phase A n'a pas saturé tous les sommets $x \in X$).

Pour chaque $x \in X$ non saturé en phase A :

— chercher la chaîne alternée saturante μ de longueur extrémale issue de x dans G sur W ou, à défaut, la *CAT* de longueur extrémale (cf. § *b*); (rappel : la chaîne vide est une *CAT* de longueur nulle);

— inverser μ et passer au suivant des x non saturés par la phase A.

Fin

Quand la phase B a été appliquée à tous les $x \in X$ non saturés en phase A, le couplage courant W est optimal.

b) *Sous-algorithme pour la recherche de* μ

Variables :

Outre le graphe avec les valeurs des arêtes, et la donnée du couplage courant W, on devra pouvoir écrire à côté de chaque sommet :

— une marque numérique $M(x)$ ou $M(y)$ pour la tenue à jour des longueurs,

— une croix indiquant si le sommet doit être exploré au cours de la prochaine demi-itération,

— pour les sommets de Y, l'« Origine » de sa marque, c'est-à-dire le sommet $x \in X$ qui a permis d'obtenir sa marque actuelle.

On utilise, d'autre part, deux variables sans indice, L (longueur de la plus courte *CAS* issue de x) et J (sommet $y \in Y$ qui a permis de trouver le L en cours).

Algorithme

Le sommet origine sera appelé x_0. Par commodité, *extrémal* signifie *minimal* (sinon remplacer $>$ par $<$, ∞ par $-\infty$, etc.).

Initialisation

Poser $M(a) = \infty \; \forall \, a \in X \cup Y$. Poser $M(x_0) = 0$ et faire une croix devant x_0.

Poser $L = \infty$.

Itération

1re *demi-itération*

Pour chaque $x \in X$ porteur d'une croix :

— effacer la croix et examiner chaque $y \in \Gamma x$:

. si $(x, y) \in W$ passer au $y \in \Gamma x$ suivant. Sinon :

. si $M(y) \leqslant M(x) + v(x, y)$, passer au y suivant. Sinon :

. poser $M(y) = M(x) + v(x, y)$ et Origine $(y) = x$. Alors :

. si y est saturé, le marquer d'une croix et passer au suivant. Sinon :

. si $L \leqslant M(y)$ passer au $y \in \Gamma x$ suivant. Sinon :

. poser $L = M(y)$, $J = y$ et passer au suivant.

2e *demi-itération*

Si aucun y ne porte de croix, aller à *Fin*. Sinon :

— pour chaque y porteur d'une croix :

. effacer la croix,

. x étant le sommet qui sature y, poser $M(x) = M(y) - v(x, y)$ et faire une croix devant x.

— Quand chaque y porteur d'une croix a été traité, reprendre au début de l'itération.

Fin

Plus aucun ajustement ne peut être fait. Si $L \neq \infty$, μ est une *CAS* de longueur L. Son dernier sommet est le $y \in Y$ qui se trouve inscrit en J. Si $L = \infty$, μ est une simple *CAT*, qui se termine sur le sommet $x \in X$ qui minimise $M(x)$. (Si $x = x_0$, μ est vide.) On a alors $L(\mu) = M(x)$.

Qu'il s'agisse d'une *CAS* ou d'une *CAT*, on remonte μ en passant d'un y au x inscrit comme origine de y, et d'un x au y qui le sature, jusqu'à ce que l'on rencontre x_0. μ peut naturellement être inversée à mesure qu'on la remonte.

II. JUSTIFICATION DE L'ALGORITHME

Il existe peut-être une preuve rapide et élégante de l'algorithme présenté en première partie. Nous ne l'avons pas cherchée, préférant guider le lecteur le long du chemin qui, d'un algorithme de théorie des flots, nous a amenés par améliorations successives à l'algorithme actuel.

1. Premier algorithme tiré de la théorie des flots

a) *Préambule*

Il est bien connu que le problème d'affectation peut être traité par des algorithmes de théorie des flots. D'ailleurs l'algorithme hongrois lui-même

peut être entièrement prouvé au sein de cette théorie. Notre première tâche est donc de transposer un algorithme de théorie des flots au problème d'affectation.

Nous avons porté notre choix sur un algorithme fondé sur la notion de *graphe d'écart*, [6] parce que la transposition des chemins de ce graphe conduit aux *chaînes alternés*, bien connues des praticiens de l'algorithme hongrois.

Ce que nous avons défini comme longueur d'une *CAS* est la longueur du chemin qui lui correspond dans le graphe d'écart. L'algorithme qui suit est la transposition de l'algorithme donné par B. Roy dans [6] sous la référence « algorithme IX.3 », et se passe donc de démonstration.

b) *Algorithme 1*

1) Poser $W = \varnothing$.

2) Construire la *CAS* de longueur extremum dans G sur W (au sens des longueurs définies en I.3, *c*). Si une telle chaîne n'existe pas, aller en 4. Sinon passer en 3.

3) Inverser μ; reprendre en 2 avec le nouveau W.

4) Éditer le couplage W obtenu au dernier pas 3. *Fin.*

REMARQUE IMPORTANTE

On est assuré qu'il n'y a pas de circuit absorbant dans le graphe d'écart du problème de flot associé. On pourrait dire qu'ici il n'y a pas de « cycle alterné absorbant ». Cela signifie que s'il y a une *CAS*, il y en a une de longueur extremale, et que l'on peut la chercher sans crainte par un algorithme d'ajustements progressifs. Cela signifie aussi qu'une telle *CAS* peut être considérée comme élémentaire, après élimination éventuelle des cycles de longueur nulle.

2. Graphes saturables : première amélioration

a) *Préambule*

Un *couplage* W de $G = (X, Y, \Gamma)$ sera dit *saturant* si $|W| = |X|$ (ce qui implique $|X| \leqslant |Y|$).

Le *graphe* $G = (X, Y, \Gamma)$ sera dit *saturable* s'il admet un couplage saturant. Nous verrons plus loin que l'on peut toujours se ramener à ce cas, ce qui permettra d'en exploiter les importantes propriétés.

b) *Translation et équitranslation du système de valeurs ; propriétés*

Soit t une application de X dans $R(t : X \to R)$. On appelle translation du système de valeurs fondée sur t, la transformation T_t qui, pour tout $(x, y) \in \Gamma$, remplace $v(x, y)$ par $v(x, y) + t(x)$.

Si $t(x)$ est constant sur X, alors T_t est appelée une *équitranslation.*

Propriété 1 (évidente)

Une équitranslation du système de valeurs laisse inchangées les solutions optimales du problème d'affectation.

Conséquence importante

Par changement de signe et équitranslation, on peut toujours se ramener à la recherche d'un *couplage maximum de valeur minimum*, les arêtes ayant des *valeurs non négatives*. C'est ce que nous supposerons désormais.

Propriété 2

Une translation T_t quelconque augmente d'une même longueur toutes les *CAS* issues d'un même $x \in X$.

Preuve

La première arête est augmentée de $t(x)$. Elle est suivie de couples d'arêtes de type (yx'), $(x'y')$ dont la longueur totale s'augmente de

$$[-t(x') + t(x')] = 0.$$

Conséquence

L'ensemble des plus courtes *CAS* issues de x est inchangé dans une translation.

Propriété 3

Si G est saturable, une translation T_t augmente de la même quantité la valeur de tous les couplages saturants, et laisse inchangé l'ensemble des couplages optimaux.

Preuve

Cette quantité unique est $\sum_{x \in X} t(x)$.

REMARQUE TRÈS IMPORTANTE

La conséquence de la propriété 2 autorise l'utilisation des translations comme un outil purement formel : dès lors que l'on se contentera de chercher la plus courte *CAS* issue d'un x désigné, *on n'aura pas besoin d'effectuer réellement la translation*. Notamment dans un algorithme opérationnel, on ne fera pas la translation destinée à rendre positives toutes les valeurs d'arêtes.

c) *Utilisation des translations*

Lemme 1

Si G est saturable, il existe une translation T_t du système de valeurs, telle que l'algorithme 1 sature les sommets $x \in X$ dans un ordre arbitraire.

Preuve

Montrons d'abord que, étant donné $A \subset X$, il existe une translation telle que les $x \in A$ soient saturés avant les $x \in X - A$. Pour tout $x \in X$, posons $M(x) = \underset{y \in \Gamma x}{\text{Max}}\, v(x, y)$, et $K = \sum_{x \in X} M(x)$, et soit alors T_t définie par

$$\begin{cases} t(x) = 0 \ \forall\, x \in A \\ t(x) = 2K + 1 \ \forall\, x \in X - A. \end{cases}$$

On montre aisément, par majoration et minoration sur les longueurs des chaînes élémentaires que, tant qu'il reste des x dans A, ils fournissent des *CAS* plus courtes et sont saturés en priorité.

Soit maintenant $x_1, x_2, \ldots, x_m$ ($m = |X|$) la liste des sommets de X dans l'ordre où on a décidé de les saturer. Posons $A_i = \{x_1, x_2 \ldots, x_i\}$ pour $i = 1, m - 1$. On peut définir une translation qui garantit la saturation de A_1 en priorité; puis, A_2 en priorité, et ainsi de suite. Aucune des translations ne détruit les effets des précédentes, et leur composition est évidemment une translation, ce qui établit le lemme.

d) *Algorithme 2* (pour un graphe *saturable*)

1) Poser $W = \varnothing$.

2) Pour chaque $x \in X$ dans un ordre arbitraire :

. construire la plus courte *CAS* μ issue de x dans G sur W,

. inverser μ,

. passer au x suivant dans l'ordre choisi.

3) Quand tous les $x \in X$ ont été passés en revue dans 2, le couplage saturant obtenu a une valeur minimum.

Preuve

Par le lemme 1, on peut trouver une translation T_t telle que l'algorithme 1 devienne l'algorithme 2. Par la remarque sur la propriété 2 des translations, on voit que l'algorithme 2 peut aussi bien chercher ses *CAS* avec les anciennes valeurs des arêtes qu'avec les nouvelles.

3. Introduction d'une « phase de déblayage »

Algorithme 3 (graphes *saturables*)

— *Initialisation.* Poser $W = \varnothing$.

— *Déblayage.* Pour chaque $x \in X$ pris successivement :

. chercher $y_0 \in \Gamma x$ tel que

$$\begin{cases} v(x, y_0) = \underset{y \in \Gamma x}{\text{Min}}\, v(x, y) \\ y_0 \text{ est insaturé dans } W; \end{cases}$$

. si on a pu trouver y_0, ranger (x, y_0) dans W;

. passer dans tous les cas au $x \in X$ suivant.

— *Finissage* (A n'exécuter que s'il reste des $x \in X$ insaturés après la phase de déblayage).

— Pour chaque $x \in X$ insaturé :

. construire la plus courte CAS μ issue de x;

. inverser μ. Passer au x insaturé suivant avec le nouveau W.

Fin

Le couplage W obtenu à la fin de la deuxième phase est optimal.

Preuve

Numérotons les sommets dans l'ordre où l'algorithme 3 les a saturés, soit $x_1, x_2, \ldots, x_p, x_{p+1}, \ldots, x_m$, les sommets x_1 à x_p ayant été saturés en phase de déblayage, et choisissons cet ordre pour appliquer l'algorithme 2 : les algorithmes 2 et 3 coïncident pour les sommets x_1 à x_p, parce que la plus courte chaîne issue d'un de ces x est composée de la seule arête (x, y_0) que la phase de déblayage range dans W. Les deux algorithmes coïncident formellement à partir de x_{p+1}, ce qui achève la démonstration.

4. Cas de graphes non saturables : Graphe G^*

Si l'on craint que G ne soit pas saturable, on peut se ramener au problème d'affectation sur un graphe G^* un peu différent, et saturable.

a) *Définition de G^**

Soit Y' un ensemble tel que $X \cap Y' = \varnothing$, $Y \cap Y' = \varnothing$ et $|Y'| = |X|$, et soit une *bijection* $f : X \to Y'$.

Posons, comme plus haut

$$M(x) = \operatorname*{Max}_{y \in \Gamma x} v(x, y) \; \forall \; x \in X, \quad \text{et} \quad C = 1 + \sum_{x \in X} M(x)$$

Nous définissons alors $G^* = (X, Y^*, \Gamma^*)$ ainsi :

$$\begin{cases} Y^* = Y \cup Y' \\ \Gamma^* x = \Gamma x \cup \{f(x)\} \; \forall \; x \in X, \quad \text{avec} \quad v(x, f(x)) = C \end{cases}$$

b) *Application S de l'ensemble des couplages de G dans l'ensemble des couplages saturants de G^**

Soit W un couplage de G. Son image par S est :

$$S(W) = W \cup \{(x, f(x)) \mid x \in X \text{ insaturé dans } W\}$$

$S(W)$ est clairement un couplage saturant de G^*.

c) *Propriété fondamentale de S*

Soit R le préordre total défini par :

$$WRW' \Leftrightarrow \begin{cases} |W| > |W'| \\ \quad \text{ou} \\ |W| = |W'| \quad \text{et} \quad V(W) \leqslant V(W') \end{cases}$$

on a le résultat suivant :

Lemme 2

S est une bijection telle que :

$$WRW' \Leftrightarrow V \circ S(W) \leqslant V \circ S(W')$$

Preuve

On montre que $C(|X| - |W|) \leqslant V \circ S(W) < C(|X| - |W| + 1)$ d'où $|W| > |W'| \Rightarrow V \circ S(W) < V \circ S(W')$, et le reste est évident.

Corollaire du lemme 2

La bijection S met en correspondance les couplages optimaux de G et ceux de G^*.

d) *Algorithme 4* (graphe G saturable ou non)

Initialisation

Construire G^* et poser $W^* = \varnothing$.

Déblayage et *finissage*

Comme algorithme 3, en remplaçant W par W^* et G par G^*.

Fin

La solution est $W = S^{-1}(W^*)$

5. De l'algorithme 4 à l'algorithme définitif

a) *Algorithme principal*

Il s'agit de rendre implicite le traitement des sommets de Y' et des arêtes $(x, f(x))$. Une telle arête ne peut exister dans une *CAS* μ que comme dernière arête. Si, au cours de l'algorithme, une telle *CAS* μ est la plus courte issue d'un sommet x' de X, c'est que toutes les chaînes issues de x' se terminent par une arête de ce type. Si nous retirons à toutes ces *CAS* leur dernière arête, il subsiste des *CAT*, dont les longueurs s'ordonnent selon la même hiérarchie. D'autre part, après l'inversion de μ, x est définitivement saturé par $f(x)$, car plus aucune *CAS* ne passera par ces sommets. Ces deux circonstances permettent de modifier l'algorithme 4 pour en tirer l'algorithme présenté en première partie.

b) *Sous-algorithme de recherche de* μ

L'absence de circuit absorbant autorise un algorithme de type dit « à ajustements progressifs » sans risque de bouclage. Nous avons profité de la structure du graphe pour faire ces ajustements dans un mouvement de va-et-vient entre les classes X et Y. Mises à part ces dispositions spéciales, l'algorithme est tout à fait classique.

III. MISE EN ŒUVRE ET PERFORMANCES

1. Variantes

a) *Ordre de saturation libre en phase de finissage*

L'ordre dans lequel la phase de finissage sature chaque sommet x n'a pas d'importance. Aussi, dans cette phase, on peut procéder comme dans l'algorithme 1 et chercher la plus courte *CAS* (ou *CAT*) dans G sans imposer son sommet initial (recherche des chaînes en parallèle). La modification du programme est très légère (quelques cartes à changer). Les gains de temps observés au cours des essais, sont trop faibles pour que nous les jugions significatifs.

b) *Cas de la matrice pleine (ou presque pleine)*

Dans ce cas on rangera les coûts sous forme matricielle, alors qu'une liste est plus avantageuse si le taux de remplissage est faible.

c) *Matrice carrée et pleine*

Dans ce cas tout x et tout y doit être saturé. On peut donc éventuellement remplacer la phase de déblayage par la phase initiale de la Méthode Hongroise. Les coûts sont sous forme matricielle.

2. Essais et comparaisons

a) *Présentation des essais*

Nous avons employé une matrice carrée et pleine de 51×51, représentant un problème concret d'affectation de personnel. Cette matrice nous a été aimablement fournie par un ingénieur de la Société concernée.

Nous avons testé les programmes FORTRAN suivants :

— Un algorithme Hongrois, fourni par la même personne.

— Un algorithme de Ford-Fulkerson programmé par l'auteur.

— L'algorithme décrit en première partie.

— Le même algorithme, modifié dans le sens de III 1 *c*.

Les temps sont ceux de l'Unité Centrale CDC 6600, chronométrés entre la fin de la lecture des données et le début de l'édition des résultats.

b) *Résultats*

ALGORITHME	TEMPS EN SECONDES
Hongrois	13,35
Ford-Fulkerson	5,90
Algorithme décrit en I	3,15
Modification III 1 c	2,85

c) *Commentaire*

Le problème proposé était difficile, en ce sens que la phase de déblayage de l'algorithme Hongrois ne fournit que 15 saturations (sur 51). Il aurait été intéressant de disposer de jeux de données variés pour suivre l'évolution des performances selon la difficulté du problème. Toutefois il ne faut pas oublier que c'est justement sur les problèmes difficiles que les gains de temps sont utiles.

CONCLUSION

Imaginé surtout pour le problème d'affectation à tableau rectangulaire et peu rempli, l'algorithme présenté ici nous paraît capable de venir battre sur leur propre terrain les techniques spécifiques du problème à tableau carré et plein. L'infériorité principale de la Méthode Hongroise réside sans doute dans la multiplicité des recherches infructueuses de chaînes alternées : cet écueil est évité par Ford-Fulkerson. La seconde cause de perte de temps, présente aussi dans Ford-Fulkerson, est la lourde transformation du système de coûts à chaque itération majeure. C'est pourquoi on peut penser que la méthode de Ford et Fulkerson reprendrait l'avantage pour des problèmes plus « faciles », et demandant par conséquent un petit nombre d'itérations majeures. Il reste naturellement que l'algorithme proposé peut être grandement amélioré par un choix heuristique judicieux de l'ordre de saturatoin des sommets.

BIBLIOGRAPHIE

[1] J. L. AGARD, *Méthode Hongroise*, Papier interne, Air France, 1965.

[2] M. L. BALINSKI et R. E. GOMORY, *A Primal Method for the Assignment and Transportation Problems*, Management Science, vol. 10, n° 3, 1964, 578-593.

[3] L. R. FORD Jr. et D. R. FULKERSON, *Flows in Networks*, Princeton U. Press, 1962.

[4] P. L. IVANESCU et S. RUDEANU, *A pseudo-boolean approach to matching problems, in* : « Théorie des graphes », Journées Internationales d'Études, Rome, juillet 1966. Publié par Dunod, Paris et Gordon and Breach, New York, 1967.

[5] H. W. KUHN, *The Hungarian Method for the Assignment Problem*, Nav. Res. Log. Quart., vol. 2, n° 1 et 2, 1965.

[6] B. ROY, *Algèbre moderne et théorie des graphes*, tome II, Chapitre 9, Dunod, Paris, 1970.

[7] I. M. STANCU-MINASIAN, *The Utilization of the Graph Teory in Solving the Assignment Problem, in* : « Economic Computation and Economic Cybernetics Studies and Research » (pp. 69-84). The Centre of Economic Computation and Economic, Cybernetics, Bucharest, 1970.

[8] Y. TABOURIER, *Sur un algorithme d'affectation*, Direction Scientifique Metra, Note de travail 139, avril 1971.