RAIRO. RECHERCHE OPÉRATIONNELLE

A. QUILLIOT

Décomposition en matrices graphiques de matrices en $\{0, 1, -1\}$: application à la résolution de programmes linéaires entiers

RAIRO. Recherche opérationnelle, tome 27, n° 3 (1993), p. 293-306

<http://www.numdam.org/item?id=RO_1993__27_3_293_0>

Recherche opérationnelle/Operations Research
(vol. 27, n° 3, 1993, p. 293 à 306)

DÉCOMPOSITION EN MATRICES GRAPHIQUES DE MATRICES EN {0, 1, –1} : APPLICATION A LA RÉSOLUTION DE PROGRAMMES LINÉAIRES ENTIERS (*)

par A. QUILLIOT ([1])

Résumé. – *Reconnaître une matrice graphique constitue un problème connu pour être polynômial. A partir d'une nouvelle caractérisation de ces matrices, nous déduisons une méthode de décomposition d'une matrice quelconque en {0, 1, –1} en matrices graphiques et nous montrons comment dans certains cas, cette méthode peut s'insérer à l'intérieur d'un processus de résolution de programmes linéaires entiers basé sur un mécanisme de transfert des contraintes d'intégrité depuis les variables de départ vers un ensemble plus réduit de variables de contrôle.*

Mots clés : Matrices graphiques ; programmation linéaire entière ; totale unimodularité.

Abstract. – *Recognizing a network matrix is known to define a polynomial problem. From a new characterization of these matrices, we deduce a decomposition scheme for general {0, 1, –1}-matrices, and we show how his scheme can be part of a resolution process for linear integer programs based upon the idea of transfering the integrality constraints from the original unknown vector to a somewhat smaller control vector.*

Keywords: Network matrices; linear integer programming; total unimodularity.

I. INTRODUCTION

Les matrices graphiques, connues pour former la base des matrices unimodulaires (théorème de Seymour [14, 15]), sont reconnaissables polynômialement et ont été largement étudiées (*voir* [1, 10, 11, 13, 16]). Nous nous intéressons ici au problème suivant :

Une matrice A en $\{0, 1, -1\}$ quelconque étant donné, comment à chaque colonne associer un arc d'un certain arbre de telle sorte que chaque ligne de A « apparaisse alors le plus possible » comme représentative d'un certain chemin, c'est-à-dire de telle sorte que A apparaisse le plus possible comme une matrice graphique ?

(*) Reçu en décembre 1991.
([1]) Université Blaise-Pascal, Plateau des Cézeaux, 63174 Aubière.

Recherche opérationnelle/Operations Research, 0399-0559/93/03/$ 4.00

Nous verrons qu'à ce problème, *NP*-complet, correspond une interprétation assez simple en terme de processus de résolution pour des programmes linéaires entiers. Du fait même de cette interprétation, il s'avère que ce qui importe dans ce problème est non pas sa résolution exacte, mais l'obtention en un temps rapide d'une solution approchée. A partir d'une caractérisation des matrices graphiques proche de celle fournie par P. Duchet [9] pour les matrices (hypergraphes) d'intervalles, nous présenterons un algorithme glouton conçu selon cet objectif.

II. LE PROBLÈME DE L'ARBRE-DÉCOMPOSITION D'UNE MATRICE EN {0, 1, -1} (ADP)

Quelques définitions :

Arbre : Dans toute la suite, nous nommerons arbre un graphe orienté sans cycle ni boucle ; un A-chemin d'un tel arbre sera alors l'ensemble des arcs associé à un chemin élémentaire signés + ou − selon que leur orientation coïncide ou non avec celle du chemin ; un sommet x d'un tel arbre sera dit bout d'un arc e si il est origine ou extrémité de e.

Hypergraphe signé : Nous nommons hypergraphe signé tout couple $H = (X, E)$, où E est une famille de parties (arêtes signées) de $X.\{-, +\}$ telle que dans aucune de ces parties un même sommet x dans X ne figure accompagnés de 2 signes différents. Un couple $(x, +)$ $((x, -))$ sera usuellement noté x^+ (x^-). La notion de sous-hypergraphe signé induit par un sous-ensemble de X se définira comme pour les hypergraphes ordinaires.

Commentaire : Il est clair que la notion d'hypergraphe se confond de fait avec celle de matrice en $\{0, 1, -\}$. Nous privilégierons la première du fait des analogies existant entre notre démarche et celles ayant prévalu dans le cadre des travaux sur les hypergraphes (matrices) d'intervalles (*voir* [9]).

Arbre-représentation : $H = (X, E)$ étant donc un hypergraphe signé, tout arbre $r = (S, X)$ sera nommé arbre-représentation de H. Pour un tel arbre, on posera :

– si e dans E, alors $\tau_H(e, r) = \text{Min} \mid I \mid$ tel que e peut s'écrire

$e = \underset{i \text{ dans } I}{\text{UNION}} e_i$, où les e_i sont les A-chemins de r.

– $\tau_{H^*}(r) = \sum \tau_H(e, r) - |E|$, r Arbre-représentation de H.

Exemple-base : Supposons H représenté par la matrice A suivante :

	a	b	c	d	e
$E1$	1	−1	0	0	1
$E2$	−1	0	0	1	−1
$E3$	0	1	1	−1	0
$E4$	0	1	−1	0	−1

Pour l'arbre-représentation r de H ci-dessous (*fig.* 1)

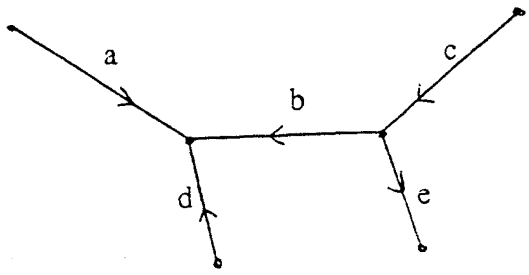

Figure 1

nous aurons :

$$\tau_H(E1, r) = 1; \quad \tau_H(E2, r) = 2; \quad \tau_H(E3, r);$$
$$\tau_H(E4, r) = 2 \quad \text{et} \quad \tau_H^*(r) = 2.$$

Le problème ADP (arbre décomposition problème) consistera alors à, un hypergraphe signé H étant donné, trouver un arbre-représentation r de H qui minimise la quantité $\tau_{H^*}(r)$.

Remarque : H induit une matrice graphique si et seulement si il existe un arbre-représentation r de H tel que $\tau_{H^*}(r) = 0$. Une telle arbre-représentation est alors nommée bonne arbre-représentation.

III. UNE INTERPRÉTATION DU PROBLÈME ADP EN TERME DE PROGRAMMATION LINÉAIRE ENTIÈRE

Considérons un programme linéaire entier P :

$x \geq 0$ dans Z^n, $A\ x = b$, A étant une m^*n-matrice en $\{0, 1, -1\}$;

A peut être considérée comme associée à un hypergraphe signé $H(A) = (C, L)$, C et L étant respectivement les ensembles de colonnes et de lignes de A.

Supposons obtenue une arbre-décomposition r de $H(A)$ telle que $\tau_{H^*}(R)$ soit sensiblement plus petit que $|C|$.

Tout vecteur ligne l de A peut alors s'écrire :

$$l = l_1 + l_2 .+...+ l_{k(l)}$$

où les l_i correspondent à des A-chemins de r ;

la somme $\sum_{l \text{ dans } L} k(l)$ étant petite comparée à n.

Nous introduisons alors, pour tout l dans L, des variables de contrôle $t_{l,i}$, i in l ... $k(l)$, soumises aux contraintes suivantes :

(CCO) : Pour tout l in L : $\sum_{i \text{ dans } l \ldots k(l)} t_{l,i} = b_l$.

Ayant initialisé ces variables, nous pouvons résoudre le programme P^* suivant :

– Vecteur inconnu $x \geq 0$ dans Z^n.

Pour tout l dans L, i dans l ... $k(l)$, $l_i.x = t_{r,i}$.

La matrice de P^* est une matrice graphique, donc totalement unimodulaire : il s'ensuit que P^* peut être relaché de sa contrainte d'intégrité puis résolu par l'algorithme du simplexe (par exemple) (*voir* [13]).

Si P^* ainsi relaché admet une solution alors nous avons fini sinon il vient par dualité l'existence de :

$$\mu = (\mu_{l,i}, \text{ dans } L, i \text{ dans } 1 \ldots k(l))$$

tel que :

Pour tout c dans C :

$$\sum_{l \text{ dans } L} \sum_{i \text{ dans } l \ldots k(l)} \mu_{l,i} \cdot l_i \geq 0;$$

$$\sum_{l \text{ dans } L} \sum_{i \text{ dans } l \ldots k(l)} t_{l,i} \cdot \mu_{l,i} < 0;$$

et nous pouvons ajouter la « coupe de Benders » :

$$\sum_{l \text{ dans } L} \sum_{i \text{ dans } 1 \ldots k(l)} t_{l,i} \cdot \mu_{l,i} \geq 0; \quad (\alpha)$$

aux relations contenues dans le programme (CCO) ;

Exemple : Reprenons les données relatives à l'exemple-base (matrice de contraintes A + arbre-représentation r), ainsi que le vecteur $b = (5, 7, 8, 10)$; en introduisant les variables de contrôles t, u, nous pouvons réécrire le programme linéaire $A\ x = b$, $x \geq 0$ entier, sous la forme :

$$
\begin{array}{rcrcrcrcrcl}
x_1 & - & x_2 & + & & & & & x_5 & = & 5 \\
-x_1 & & & & & + & x_4 & & & = & t \\
& & & & & & & - & x_5 & = & -t \\
& & x_2 & + & x_3 & - & x_4 & & & = & 8 \\
& & x_2 & & & & & - & x_5 & = & u \\
& & & - & x_3 & & & & & = & 10 - u
\end{array}
$$

Pour une initialisation à $t = 5$, $u = 5$, nous voyons que le système ci-dessus n'a pas de solution fractionnaire et permet de faire apparaître la nouvelle contrainte (α) $t \geq 30$.

En procédant de la sorte, nous transférons la contrainte d'intégrité depuis les variables de P sur les variables de contrôle de (CCO). Deux types de stratégies peuvent être alors envisagées :

1° approche : par génération successive de contraintes sur les variables de contrôle. On obtient l'algorithme suivant :

```
Algorithme DECOMPOSE
Input : Le programme P ;
Output : La faisabilité de P (Booleen)
début
Trouver une « bonne » arbre-représentation de H(A) ;
Initialiser les variables de contrôle (en prenant par exemple
t_{l,i} = partie entière inférieure ou supérieure de
    b_l. (Nombre de 1 ou – 1 dans l_i /nombre de 1 ou – 1 dans l).
Not Succes; Not Echec;
Tant que Not Stop faire
   début
   Résoudre P* ; (par relaxation de la contrainte d'intégrité)
   Si une solution existe alors succès sinon
      début
      Former (α) et l'insérer dans (CCO) ;
      Ajuster les variables de contrôle dans (CCO) ;
      Si pas de solution alors échec ;
      fin ;
   fin ;
Si Succes alors Faisabilité sinon Not Faisabilité ;
fin.
```

Remarque : On notera encore que les contraintes dans (CCO) étant elles aussi en $\{0, 1, -1\}$, il peut être envisagé de répéter la méthode pour la résolution de (CCO).

Performances : Elles dépendent pour une large part de la façon dont on traitera l'instruction soulignée « Ajuster les variables de contrôle dans

(CCO) », qui implique la résolution d'un système d'inégalités linéaires entières pour lequel on dispose d'une solution qui satisfait toutes les contraintes sauf une. Nous avons effectué des tests en utilisant pour traiter cette instruction une procédure par « Branch and Cut » et compté le nombre de contraintes additionnelles sur le vecteur de contrôle qui étaient générées avant l'obtention du résultat. Nous avons alors obtenu :

Nombre de variables initiales : 30.

p = nombre de trous ($\tau_{H^*}(r)$) = nombre de variables de contrôle ;

q = nombre de contraintes (α) générées au cours de l'algorithme.

Moyennes sur 5 tests effectués pour chaque valeur de p :

p	5	10	15	20
q	6,5	15,8	29	51

Ces tests font alors apparaître que si l'on applique au programme de départ et au traitement de l'instruction d'ajustement la même routine de « Branch and Cut » (façon de procéder peu favorable car n'exploitant pas les spécificités du système à résoudre à l'intérieur de l'instruction d'ajustement), alors il y a gain de temps CPU jusqu'à $p = 10$.

2° approche : Par Branch and Cut sur le vecteur de contrôle.

On procède alors de la façon suivante :

Un nœud de l'arbre des sous-problèmes est créé en imposant une contrainte

$$t_{l,i} \leq (\leq, =)\ \beta, \qquad (l \text{ dans } L,\ i = l \ldots k(l)) ;$$

Ce nœud est stérilisé soit si le programme défini par :

– Vecteur inconnu $x \geq 0$ dans R^n, $t = (t_{l,i},\ l \text{ dans } L,\ i = 1 \ldots k(l))$;

Pour tout l dans L, i dans $1 \ldots k(l)$, $l_i.\ x = t_{r,i}$.

Pour tout l dans L, $\sum_{i=1 \ldots k(l)} t_{l,i} = b_l$;

Contraintes sur les t issues des nœuds précédant et du nœud courant.

ne possède pas de solution (retour arrière sur échec) soit si la résolution laisse apparaître une solution entière (succès).

Les tests effectués sur les mêmes exemples que pour la première approche permettent d'obtenir (en posant N = nombre de nœuds générés au cours de la recherche)

p	5	10	15	20
N	9	28	69	95

IV. UNE CARACTÉRISATION EN TERMES D'HYPERGRAPHES DES MATRICES GRAPHIQUES

Remarquons tout d'abord que si l'hypergraphe signé $H = (X, E)$ est en fait un graphe simple (toutes les arêtes sont positives et de cardinalité 2), alors résoudre le problème ADP consiste à rechercher une famille d'arêtes de cardinalité minimale qui soit transversale à tous les cycles impairs du graphe, ce qui permet d'affirmer [12] :

PROPOSITION 1 : *ADP est NP-dur.*

Effectuons maintenant quelques rappels sur la notion de représentation par intervalles.

Rappel : Un hypergraphe d'intervalles est un hypergraphe dont les sommets peuvent être totalement ordonnés de façon à faire apparaître chaque arête comme un intervalle.

THÉORÈME (Duchet [9]) : *Soit $H = (X, E)$ un hypergraphe tel que l'union de deux arêtes d'intersection non vide est aussi une arête ; un tel hypergraphe est d'intervalles si et seulement si il n'existe pas de sous-hypergraphe partiel de H qui soit un graphe complet à 3 sommets.*

Le but de ce paragraphe va dès lors être de présenter un analogue de ce résultat dans le contexte des matrices graphiques, dont le principal intérêt sera en fait de sous-tendre une heuristique gloutonne de traitement du problème ADP.

DÉFINITIONS : Soit $H = (X, E)$ un hypergraphe signé.

– Pour e dans E, nous posons $T(e) = \{x^{t(s)}, x^s \text{ dans } e\}$, t étant la transformation définie par $t(+) = -$ et $t(-) = +$ et $P(e) = \{x \text{ dans } X \text{ tel qu'il existe } s \text{ dans } \{+, -\} \text{ avec } x^s \text{ dans } e\}$.

– Nous disons que H est stable par inverse si pour tout e dans E, $T(e)$ est aussi dans E.

– Nous disons que H est intersection-compatible si pour tout e, e' dans E on a :

$e \star e' = \text{Nil}$ ou bien $| e^\star e' | = | P(e)^\star P(e') |$ (nous utilisons ici le symbole de multiplication $\star$ pour désigner l'opérateur d'intersection)

(c'est-à-dire que les sommets apparaissant à la fois en e et en e' sont signés tous de façon identique ou tous de façon opposée).

– Nous disons que H est stable par union connexe si pour tout e, e' dans E d'intersection non vide on a l'existence de a ,b disjoints et inclus dans e et a', b' disjoints et inclus dans e' tels que :

$$e = a + b + (e^\star\ e')\ ;$$
$$e' = a' + b' + (e^\star\ e')\ ;$$

$a + b' + (e^\star\ e')$ et $a' + b + (e^\star\ e')$ sont dans E ;

$a + (T(b'))$ et $a' + T(b)$ sont dans E.

Nous obtenons alors :

THÉORÈME I : *Soit donc $H = (X, E)$ comme ci-dessus, stable par inverse et par union connexe et intersection-compatible. H est alors représentatif d'une matrice graphique si et seulement si*

il n'existe pas $\{e_i,\ i$ *dans* $1 \ldots 3\}$ *inclus dans* E, $\{x_i,\ i$ *dans* $1 \ldots 3\}$ *inclus dans* X, s_1, s_2, s_3 *dans* $\{-, +\}$ *tels que pour tout* i *dans* $1 \ldots 3$, $x_i\ s_i$ *est dans* $(e_{i+1}$ UNION $e_{i+2}) - e_i$ *(l'addition étant prise ici modulo* 3). $(\star)$

Démonstration : La partie « seulement si » de l'équivalence est aisée à vérifier.

Afin de prouver la réciproque, nous considérons $H = (X, E)$ satisfaisant $(^\star)$ ainsi que les hypothèses contenues dans l'énoncé ci-dessus, et, raisonnant par induction sur X, supposons qu'a été construit un arbre (Y, S), S inclus dans X, qui soit une bonne arbre-représentation du sous-hypergraphe de H signé induit par S. Un sommet x_0 étant donné dans X–S, il reste à prouver qu'il est possible de trouver y dans Y et une partition du sous-ensemble S_y formé des arcs de S dont un des bouts est y en 2 sous-ensembles A et B de telle sorte que l'arbre formé (et dont on dira qu'il est obtenu en insérant x_0 en y, A, B) en éclatant y en deux sommets y_A et y_B reliés par un arc $x_0 = [y_A, y_B]$ et respectivement bouts des arcs de A et B soit une bonne arbre-représentation du sous-hypergraphe de H induit par $S + \{x_0\}$.

Quelles sont donc alors les conditions que doit satisfaire un tel triplé y, A, B ?

Afin de les préciser, considérons le processus de marquage suivant, qui interviendra explicitement au cours de l'algorithme décrit au paragraphe V :

– Pour tout e dans E, y dans Y tels que:

$S^\star\ P\ (e) <> \text{Nil}\ ;\ x_0^+$ ou x_0^- dans e ;

y n'est bout d'aucun arc dans e ;

alors, on donne à y une marque RUPTURE.

– Pour tout e dans E tel que x_0^+ est dans e, tout y dans Y, tout s dans S tels que :

(origine(s) = y et s^- dans e) ou (extrémité(s) = y et s^+ dans e)

alors on donne à s, considéré comme associé à y, une marque DEBUT.

– Pour tout e dans E tel que x_0^+ dans e, tout y dans Y, tout s dans S tels que : (origine(s) = y et s^+ dans e) ou (extrémité(s) = y et s^- dans e), on donne à s, considéré comme associé à y, une marque FIN.

– Pour tout e dans E tel que x_0 non dans $P(e)$, tout y dans Y, tout s, s' dans S tels que :

s^+, s'^+ dans e, extrémité(s) = y = origine (s')

ou

s^+, s'^- dans e, extrémité(s) = extrémité (s') ou origine(s) = origine(s')

alors le couple [s, s'] reçoit une marque ENSEMBLE.

Il nous faut alors choisir y, A, B de telle sorte que y n'ait pas reçu de marque RUPTURE, qu'aucun arc s dans A n'ait été marqué FIN, qu'aucun s dans B n'ait été marqué DEBUT (de telle sorte que l'insertion de x_0 en y, A, B ne « déchire » aucune arête signée de H contenant x_0^+ ou par passage à l'inverse x_0^-), qu'aucun couple [s, s'] avec s dans A et s' dans B n'ait été marqué ENSEMBLE (de façon à ne « déchirer » aucune arête signée de H qui ne contient ni x_0^+ ni x_0^-).

Exemple : Reprenons H tel que dans l'exemple-base, et supposons déjà construit l'arbre-représentation partielle suivante (*fig.* 2) :

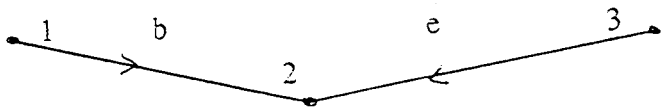

Figure 2

Nous avons donc $S = \{b, e\}$ et supposons $x_0 = c$.

Le sommet 3 de l'arbre reçoit une marque RUPTURE du fait de l'arête signée $E3$.

Le couple d'arcs [b, e] est marqué ENSEMBLE du fait de l'arête signée $E2$ (en 2).

L'arc b en 2 est marqué DEBUT du fait de $E3$ et FIN du fait de $E4$.

L'arc b en 1 est marqué FIN du fait de $E3$ et DEBUT du fait de $E4$.

On voit que :

– Du fait de l'intersection-compatibilité, il ne peut exister d'arc de l'arbre (Y, S) qui soit à la fois marqué DEBUT et FIN par rapport à un même sommet.

– La stabilité par union connexe fait que si les traces sur S de deux arêtes signées e et e' de H contenant x_0^+ se représentent dans (Y, S) par des chemins non vides et ne partageant aucun sommet, alors il existe une des 4 arêtes

signées issues de la définition de la stabilité par union connexe dont la trace S ne définit pas un A-chemin de (Y, S). Il s'ensuit que globalement, les chemins non vides de (Y, S) associés aux arètes signées de H contenant x_0^+ ont une intersection non vide qui est un chemin élémentaire de (Y, S). Soit donc $\Gamma = (y_0, ..., y_k)$ ce chemin, formé des arcs $s_1, ..., s_k$. On pose $\Gamma^* = \{s_i^{e_i}\}$, ou e_i est le signe de s_i dans Γ, supposé orienté de y_0 vers y_k.

Aucun sommet de Γ n'est marqué rupture, cependant qu'en chaque y_i, les ensembles D_i et F_i des arcs marqués DEBUT et FIN sont tels que :

$$D_i = \{s_i\} \quad \text{pour } i > 0 \qquad \text{et} \qquad F_i = \{s_i{+}1\} \quad \text{pour } i < k.$$

Supposons à présent qu'il ne soit pas possible de satisfaire la condition relative aux couples d'arcs marqués ENSEMBLE par rapport à un des sommets de Γ : cela signifie l'existence d'une suite $(u_{i\,j})$, i dans $0...k$, j dans $0...l(i)$, d'arcs de (Y, S) telle que :

Pour tout i dans $0...k$, j dans $0...l(i)$–1, $[u_{i,j}, u_{i,j+1}]$ est marqué ENSEMBLE par rapport à y_i.

$u_{0,0}$ est dans D_0 ; $u_{k,l(k)}$ est dans F_k. Pour tout i dans $0...k$, $u_{i,l(i)} = s_{i+1} = u_{i+l,0}$. On pose $u = u_{0,0}$, signé selon sa position par rapport à l'origine de Γ ; (+ ssi « entrant »). On pose $v = v_{k,l(k)}$ signé selon sa position par rapport à l'origine de Γ ; (+ ssi « sortant »).

LEMME 1 : *Soient y dans Y, s ,s', s'' dans S_y tels que* $[s, s']$ *et* $[s', s'']$ *sont marqués* ENSEMBLE *par rapport à y. Alors il en est de même pour* $[s, s'']$.

Démonstration : Partant de e et e' dans E permettant le marquage ENSEMBLE des couples $[s, s']$ et $[s,' s'']$, il suffit d'utiliser la stabilité par union connexe pour voir que l'une des 4 arètes signées s'en déduisant permet d'obtenir le marquage de $[s, s'']$ FIN.

LEMME 2 : *Soient y dans Y, s, s' dans S_y tels que* $[s, s']$ *est marqué* ENSEMBLE *par rapport à y, s'' dans S sans bout commun avec s, et e'' dans E tel que $P(e'')$ contient s" et s' et ne contient ni x_0^- ni x_0^+.*

Alors il existe e' dans E, ne contenant x_0^- ni x_0^+ et tel que $P(e')$ contient s et s''.

Même type de démonstration que pour le Lemme 1, basée sur l'application de la stabilité par union connexe à e'' et à l'arète signée e associée au marquage ENSEMBLE de $[s, s']$. FIN.

Des 2 lemmes ci-dessus et de l'intersection compatibilité nous déduisons l'existence de e dans E, ne contenant ni x_0^+ ni x_0^- et contenant $\Gamma^* + \{u, v\}$. Mais nous voyons alors que le marquage DEBUT de $u_{0,0}$ et FIN de s_1 par

rapport à y_0 permet de déduire l'existence de e' dans E contenant x_0^+ et u et de e'' dans E contenant x_0^+ et v.

Dans ce cas, nous pouvons écrire :

Il existe e dans E tel que : $\Gamma^* + \{u, v\}$ inclus dans e, x_0^+ non inclus dans e.

Il existe e' et e'' dans E s'écrivant respectivement $e' = \Gamma^* + \{u, x_0^+\} + A'$ et $e'' = \Gamma^* + \{v, x_0^+\} + A''$.

Si A' ne contient pas v et A'' ne contient pas u, alors nous voyons apparaître une configuration triangle et nous concluons. Sinon (nous nous plaçons dans ce cas), il existe aussi f, f' et f'' dans E s'écrivant :

$$f = \Gamma^* + \{x_0^+, u, v\} + A ;$$
$$f' = \Gamma^* + \{x_0^+\} + B' ; \qquad u \text{ non dans } B' ;$$
$$f'' = \Gamma^* + \{x_0^+\} + B'' ; \qquad v \text{ non dans } B''.$$

Appliquant la stabilité par intersection à f' et e nous voyons que si Γ^* est non vide, alors il doit exister une arête signée de H qui contient x_0^+ et contient soit u, soit v, ce qui fournit une contradiction sur la définition de Γ^*.

Nous pouvons donc supposer (cas non trivial) que Γ est réduit à un sommet (Γ^* vide).

Appliquons la stabilité par intersection à f et f' ; nous voyons apparaître une arête signée $\{x_0^+, v\} + B\$, B\$$ inclus dans B', et nous remplaçons f' par cette arête, ou bien 2 arêtes signées $\{u, x_0^+\} + B'$ et $\{v\} + B'^{-1}$. Dans ce dernier cas, nous regardons l'intersection de $\{v\} + B'^{-1}$ avec $\{x_0^+\} + B'$ et concluons à l'existence dans E de $\{x_0^+, v\}$ (et nous remplaçons f' par cette dernière arête), ou de $B' + \{x_0^+, v^{-1}\}$ ce qui entraîne une contradiction sur la notion de intersection-compatibilité. Nous pouvons donc supposer en fin de compte v dans B' et parallèlement u dans B'', ce qui nous fournit une configuration de type triangle et nous permet de conclure. FIN.

V. UNE HEURISTIQUE GLOUTONNE POUR LE PROBLÈME ADP

La démonstration du théorème I suggère alors l'algorithme suivant, qui consiste en une construction de l'arbre-représentation cherchée arc par arc, sans retour arrière :

Algorithme ARBRE :
Input : L'hypergraphe signé H = (X, E) ;
Output : Une arbre-représentation r de H.
début
Créer un arbre G = (Y, S) à un seul sommet : Not Echec ;
Tant que S <> X faire

début
Choisir x_0 dans X–S ;
Appliquer le processus de marquage par rapport à x_0 décrit au début de la démonstration du théorème *I* ;
Pour tout y dans Y non marqué RUPTURE faire
début
A(y) := {x dans S_y/Nombre de marques DEBUT >= Nombre de marques FIN} ;
q(A(y)) : = Σ Nombre de marques FIN ;
x dans A(y)
B(y) = {x sans S_y/Nombre de marques FIN > Nombre de marques DEBUT] ;
q(B(y)) : = Σ Nombre de marques DEBUT ;
x dans B(y)
ql(y) : = Cardinalité d'une coupe minimale entre A(y) et B(y) dans le graphe simple défini par les couples de ENSEMBLE ; (* pour le calcul par flots d'une coupe minimale, voir [12]*)
PENALITE(y) : = q(A(y)) + q(B(y)) + ql(y) ;
fin ;
Choisir y minimisant PENALITE ;
Insérer x en y, A(y), B(y) ;
Si PENALITE 0 alors Echec ;
fin ;
fin.

Exemple : Reprenons les données de l'exemple-base : l'algorithme ci-dessus place dans l'ordre les arcs (sommets de l'hypergraphe signé), *b, e, c, a,* et *d* (on procède en sélectionnant à chaque fois le sommet signé le plus contraint) et aboutit à l'arbre-représentation *r* suivante (*fig.* 3) (qui est optimale) : $\tau_{H^*}(r) = 1$.

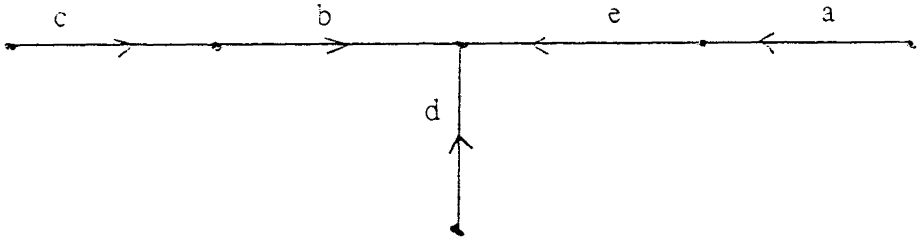

Figure 3

THÉORÈME II : *L'algorithme* ARBRE *reconnaît un hypergraphe signé représentatif d'une matrice graphique dès lors que celui-ci est stable par inverse et union connexe.*

Démonstration : Elle découle immédiatement de celle du théorème I. FIN.

Complexité et performances : On vérifiera aisément que ARBRE est polynomial ; en fait il peut être aisément implémenté de façon à traiter en un temps très court des matrices de l'ordre de 100*100. On observe alors d'assez grandes disparités dans les résultats obtenus, la quantité $\tau_H^\star(r)$ pouvant s'avérer nettement plus grande que la cardinalité de l'ensemble de sommets X. Quant à la perte par rapport à l'optimum théorique, nous ne savons pas exactement l'évaluer dans la mesure où la résolution exacte de ADP semble très difficile à obtenir dès lors que la taille de X dépasse la dizaine d'éléments. Nous avons cependant réalisé, pour une dizaine d'exemples mettant en jeu des matrices à 50 colonnes, le test suivant :

– Nous faisons tourner l'algorithme ARBRE sur l'hypergraphe signé H en question.

– Puis, à partir du résultat obtenu, nous appliquons tant que possible le procédé d'amélioration locale très simple qui consiste à trouver un arc de l'arbre-représentation r et une nouvelle position pour cet arc dans la représentation de manière à améliorer la performance $\tau_{H^*}(r)$.

Nous sommes alors en mesure d'évaluer jusqu'à quel point ARBRE génère des *optimae* locaux par rapport à la notion de voisinage définie ci-dessus. Nous obtenons :

Numéro de tests	1	2	3	4	5	6	7	8	9	10
Nombre de lignes	20	30	30	40	40	50	50	60	70	80
$\tau_H{}^*(r)$ (ARBRE)	7	12	8	25	79	22	101	125	24	68
$\tau_H{}^*(r)$ (LOCAL)	6	11	7	23	72	20	89	109	22	56

BIBLIOGRAPHIE

1. L. Auslander et H. Trent, Incidence Matrices and Linear Graphs, *J. of Maths and Mecha*, 1959, *8*, p. 827-835.
2. L. Auslander et H. Trent, On the Realization of a Linear Graph Given its Algebraic Specification, *J. of Acoustical Society of America*, *33*, p. 1183-1192.
3. P. Baptiste et J. Favrel, Résolution de problèmes d'ordonnancements par graphes d'intervalles et treillis de galois. *R.A.I.R.O.*, 1984, *18*, 4.
4. J. F. Benders, Partitionning Procedure for Solving Mixed Variables Programming Problems. *Numerische Mathematik*, 1962, *4*, p. 238-252.
5. C. Berge, Graphes et hypergraphes (chap. 5, 6), *Dunod*, 1974.
6. R. Bixby et W. Cuningham, Converting Linear Programs to Network Problems, *Maths of Operat. Research*, 1980, *5*, p. 321-357.
7. M. Chein et M. Habib, The Jump Number of Dags and Posets : an Introduction, *Ann. of discrete math*, 1980, *9*, p. 189-194.
8. V. Chvatal, Linear programming, *Freeman*, N.Y., 1983.
9. P. Duchet, Problèmes de représentations et noyaux, *Thèse d'État*, Paris-VI, 1981.

10. I. HELLER et A. HOFFMAN, On Unimodular Matrices, *Pacific Journ. of Math.*, 1962, *12*, p. 1321-1327.
11. A. HOFFMAN et J. KRUSKAL, Integral Boundary Points of Convex Polyedra, in *Linear Inequalities and Related Systems*, H. KUHN and A. TUCKER éds., Princeton Univ. Press, 1956, p. 223-246.
12. C. PAPADIMITRIOU et K. STEIGLITZ, Combinatorial optimization (chap. 3, 4, 5), Prentice Hall, 1982.
13. A. SCHRIJVER, Theory of Linear and Integer Programming (chap. 19, 20), Wiley Interscience, 1986.
14. P. SEYMOUR, Recognizing graphic matroids, *Combinatorica*, 1985, *1*, p. 75-78.
15. P. SEYMOUR, Decomposition of Regular Matroids, *J.C.T. B.*, 1980, *28*, p. 305-359.
16. W. TUTTE, An Algorithm for Determining Whether a Given Binary Matroid is Graphic, *Proc. of the American Math. Society*, 1960, *11*, p. 905-917.