RAIRO. RECHERCHE OPÉRATIONNELLE

MAURICE MILGRAM
BERNARD DUBUISSON

Un algorithme heuristique de décomposition d'un graphe

RAIRO. Recherche opérationnelle, tome 11, n° 2 (1977), p. 175-199

<http://www.numdam.org/item?id=RO_1977__11_2_175_0>

R.A.I.R.O. Recherche Opérationnelle/Operations Research
(vol. 11, n° 2, mai 1977, p. 175 à 199)

UN ALGORITHME HEURISTIQUE DE DÉCOMPOSITION D'UN GRAPHE

par Maurice MILGRAM [1] et Bernard DUBUISSON [2]

Résumé. — *Nous présentons dans cet article un algorithme qui fournit une décomposition d'un graphe en sous-graphes faiblement interconnectés. Nous montrons que le résultat est optimal pour des graphes de taille assez petite.*

I. INTRODUCTION

I.1. Pourquoi décomposer des systèmes complexes

Le but de cet article est d'exposer une méthode de décomposition de systèmes de grande taille en sous-systèmes de taille réduite. La nécessité d'une telle décomposition apparaît par exemple lorsqu'on veut répartir un ensemble de composants électroniques sur des cartes ou des plaquettes en minimisant les connexions intercartes; un autre exemple est la segmentation d'un programme pour le placer sur un support paginé. On peut aussi signaler comme domaine d'application, la Commande Hiérarchisée, qui vise à résoudre le problème de la commande d'un système de grande dimension par des commandes locales coordonnées; plusieurs études ont montré que la coordination de la commande des différents sous-systèmes convergerait d'autant mieux que ceux-ci étaient faiblement interconnectés.

I.2. Hypothèses

L'étude d'un système complexe comporte plusieurs étapes : la modélisation, l'identification, le contrôle éventuel d'une commande.

Nous supposons ici le problème formalisé, plus précisément, nous supposons que le système à étudier est représenté par un graphe. Le choix de cette représentation peut être délicat, voici deux possibilités :

a) On peut considérer le graphe où les sommets représentent les variables et les arcs les équations joignant deux variables qui figurent dans cette équation.

([1]) Assistant, Université de Technologie de Compiègne, Dept MAI, Benjamin Franklin, Compiègne Cedex.

([2]) Maître de Conférences, Université de Technologie de Compiègne.

b) Si on connaît bien le système physique (par exemple, un processus de fabrication) on peut représenter par les sommets des unités physiques et par des arcs valués des flux de matière ou d'informations entre ces unités. Nous considérons donc, dans la suite, donné un graphe représentatif du système à décomposer. [Ref. 1,2].

Un graphe orienté $G = (X, U)$

où $U \subseteq X^2$

est dit connexe si et seulement si :

$$\forall a, b \in X \quad \exists x_0, x_1, \ldots, x_k \in X \quad a = x_0 \quad x_k = b$$
$$\forall i = 0, \ldots, k-1 \quad (x_i, x_{i+1}) \in U \quad \text{ou} \quad (x_{i+1}, x_i) \in U$$

Un graphe non orienté $G = (X, E)$

où $E \subseteq P_2(X) \cup P_1(X) \quad (P_i(X) = \{ A \subseteq X ; |A| = i \})$

est dit connexe si et seulement si :

$$\forall a, b \in X \quad \exists x_0, \ldots, x_k \in X \quad a = x_0 \quad b = x_k$$
$$\forall i = 0, \ldots, k-1 \quad \{ x_i, x_{i+1} \} \in E$$

On appelle composante connexe d'un graphe un sous-graphe connexe maximal. Étant donnée une partie $A \subseteq X$, on appelle arc (resp. arête) de liaison un arc (resp. arête) ayant une extrémité dans E et l'autre dans le complémentaire de E noté $C_x E$ ou $\overline{E}$.

I.3. Formulation mathématique

Énonçons le problème mathématique que nous essayerons de résoudre avec l'algorithme :

Soit $G = (X, U)$ un graphe et des fonctions de pondération $f : X \to R^+$ et $g : U \to R^+$; soit $\theta \in R^+$ un réel fixant la taille maximum d'un sous-graphe ; on cherche une partition $P = (X_1, X_2, \ldots, X_k)$ de X telle que :

(1) $\forall i = 1, \ldots, k \quad X_i$ connexe et $\sum_{x \in X_i} f(x) \leqslant \theta$ (condition d'admissibilité)

(2) P minimise la fonction coût :

$$C(X_1, \ldots, X_k) = \sum_i \sum_{\substack{x \in X_i \\ y \in \overline{X}_i}} g(x, y) \qquad \text{(condition d'optimalité)}$$

Une partition vérifiant (1) et (2) est appelée solution du problème $P(X, U, f, g, \theta)$.

I.4. Méthodes existantes

Le problème de la partition optimum a déjà fait l'objet d'études. Les difficultés sont propres à tous les problèmes combinatoires et un compromis doit être cherché entre l'optimalité qui coûte cher en temps de calcul et une certaine sous-optimalité. Aucun algorithme optimal en temps polynomial ne figure à ce jour parmi les méthodes existantes.

Nous ne citerons que cinq méthodes, ce sont les plus connues.

Ces méthodes ne résolvent pas exactement le problème posé précédemment ; donc nous n'avons pas effectué de comparaison avec la méthode proposée.

I.4.1 *Méthode des « vecteurs propres »* [*Ref. 3*]

Cette méthode est très spécifique car le critère et les contraintes sont adaptés à un problème précis lié aux calculateurs électroniques.

Soit un graphe simple (i. e. non orienté) $G = (X, E)$ avec $|X| = n$ et E un ensemble de paires de sommets, une paire de E est appelée arête. On considère maintenant une matrice $M(G)$ réelle et d'ordre (n, n). Cette matrice a pour terme général $m_{i,j} = 0$ si $\{ x_i, x_j \} \notin E$ et $m_{i,j} = 1$ si $\{ x_i, x_j \} \in E$.

Les vecteurs propres de cette matrice jouent un rôle dans certains problèmes liés au graphe G. Par exemple, le rapport entre la plus grande et la plus petite valeur propre de $M(G)$ donne une borne du nombre chromatique de G.

La méthode en question repose principalement sur le résultat suivant : soit une partition de X en k sous-ensembles $X_1, X_2, \ldots, X_k$ de même taille et τ_i le nombre d'arêtes sortantes du groupe X_i, soit : $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n$ la suite décroissante des valeurs propres de $M(G)$. Alors :

$$\sum_{i=1}^{k} \tau_i \geqslant -\frac{n}{K}(\lambda_1 + \lambda_2 + \ldots + \lambda_K)$$

L'algorithme utilise cette borne et s'articule autour de 3 procédures : manipulation du graphe, groupement, expansion. Les auteurs admettent le caractère exponentiel de l'algorithme ce qui n'est pas forcément grave si on a une constante de temps assez petite.

I.4.2. *Méthode des colorations* [*Ref. 4*]

Le critère proposé est ici quelque peu différent. Soit $G = (X, E)$ un graphe simple et $P = (X_1, \ldots, X_r)$ une partition de X. On suppose que $|X| = n$ et $|X_i| = m_i$ et on définit $\Delta_i = \frac{n}{r} - m_i$ (écart à la moyenne) soit alors :

$$J(P) = S + \frac{1}{2}\sum_{i=1}^{r} \Delta_i^2$$

où S est le nombre d'arêtes ayant leurs extrémités dans des classes différentes. Le coefficient $\frac{1}{2}$ est introduit car tout écart positif à la moyenne d'un côté engendre un déplacement négatif dans la somme des autres écarts, tout écart est donc comptabilisé deux fois.

Ce critère intègre donc la contrainte de « taille maximum » du problème $P(G, f, g, \theta)$ sous forme d'une pénalisation par la « variance » de la taille.

La méthode utilise le graphe complémentaire $\overline{G} = (X, \overline{E})$ avec par définition

$$\{ x, y \} \varepsilon \overline{E} \Leftrightarrow \{ x, y \} \notin E$$

Le résultat essentiel est que si p est le nombre d'arêtes « coupées » dans $\overline{G}$ lors d'une partition P qui en « coupe » S dans G on a :

$$p = \text{Constante} - \left(S + \frac{1}{2} \sum_i \Delta_i^2 \right)$$

Minimiser le critère revient ici à maximiser p.

D'autre part on sait que le nombre chromatique d'un graphe croît avec le nombre d'arêtes; si G est un graphe partiel de G_2 on aura $\gamma(G_1) \leqslant \gamma(G_2)$ où $\gamma(G)$ désigne le nombre chromatique. On sait que l'ensemble des sommets de $\overline{G}$ ayant une couleur donnée est un ensemble S stable de $\overline{G}$, c'est-à-dire que $x, y \varepsilon S \Rightarrow \{ x, y \} \notin \overline{E} \Rightarrow \{ x, y \} \varepsilon E$. Les sous-ensembles de sommets ayant la même couleur dans une coloration de G fournissent des groupes de sommets de G fortement interconnectés et avec lesquels on va construire la partition cherchée.

Remarquons que le problème de la coloration d'un graphe G de n sommets en $\gamma(G)$ couleurs exige actuellement un temps $t(n)$, où $t(n)$ croît plus vite que tout polynôme.

I.4.3. *Méthode des groupes minimaux* [*Ref. 5*]

Cette méthode est originale en ce sens qu'elle s'appuie sur une notion nouvelle, celle de « groupe minimal ». Elle est bien adaptée aux problèmes des réseaux électriques et a été utilisée pour le tracé des connexions par fils isolés. Du point de vue vitesse, elle se range dans la catégorie non polynomiale mais avec une constante de temps assez faible.

Soit $G = (X, E)$ un graphe simple. Pour toute partie $Y \subseteq X$ calculons un nombre $k(Y)$ = nombre d'arêtes ayant une extrémité dans Y et l'autre dans $X - Y$. Une partie $Y \subseteq X$ sera appelée groupe minimal si

$$\forall Y' \neq \emptyset \qquad Y' \subset Y \Rightarrow k(Y') > k(Y).$$

La propriété fondamentale des groupes minimaux est la suivante : « Si Y_1 et Y_2 sont deux groupes minimaux et si $Y_1 \cap Y_2 \neq \emptyset$ alors $Y_1 \subset Y_2$ ou $Y_2 \subset Y_1$ ».

On peut aussi facilement construire tous les groupes minimaux grâce au résultat suivant : « Si $Y_1, Y_2, \ldots, Y_n$ sont n groupes minimaux disjoints (avec $n > 2$) en sorte que si $2 \leqslant m \leqslant n - 1$, la réunion de m de ces n parties ne soit pas un groupe minimal, alors $S = Y_1 \cup Y_2 \cup \ldots \cup Y_n$ est un groupe minimal ». Partant de cette propriété on énumère alors tous les groupes minimaux et on calcule une partition à partir de ceux-ci.

I.4.4. *Méthode des échanges inter-classes* [*Ref. 6*]

Cette méthode a été développée à la fin des années 60 et était orientée vers la segmentation de programmes trop gros pour être contenus « en bloc » en mémoire centrale. L'idée générale en est très simple, nous nous contenterons d'ailleurs du cas d'une bipartition. Soit donc un graphe simple $G = (X, E)$ ayant $2n$ sommets que l'on veut partager en deux sous-graphes $G_1 = (X_1, E_1)$ et $G_2 = (X_2, E_2)$ avec $|X_1| = |X_2| = n$ exactement. On part d'une bipartition quelconque et on se fixe un nombre λ dont le choix est crucial. La phase suivante consiste à choisir $A_1 \subset X_1$ et $A_2 \subset X_2$ avec $|A_1| = |A_2| = \lambda$ de manière à ce que la partition : (X_1', X_2') définie par :

$$X_1' = (X_1 - A_1) \cup A_2 \qquad X_2' = (X_2 - A_2) \cup A_1$$

améliore le critère, c'est-à-dire le nombre d'arêtes ayant leurs extrémités dans des classes distinctes.

On réitère ce processus d'échange jusqu'à ce qu'on ne puisse améliorer le critère. On obtient ainsi un minimum local d'autant meilleur que λ est grand mais dont le temps de calcul croît très vite avec λ.

Cet algorithme est très efficace pour des problèmes de taille raisonnable. Il se comporte comme un algorithme optimal si la taille du problème autorise un λ assez grand et son défaut est surtout d'être mal adapté aux multi-partitions.

I.4.5. *Méthode du flot maximal* [*Ref. 7*]

Cette méthode fournit une bipartition (X_1, X_2) d'un graphe $G = (X, E)$ minimisant le nombre d'arêtes coupées et telle que $A_1 \subset X_1$ et $A_2 \subset X_2$ où (A_1, A_2) est un couple de parties disjointes de X fixées à l'avance. Les sommets de A_1 et A_2 sont donc classés a priori ce qui diminue la généralité de l'algorithme. On commence par transformer le graphe G en un réseau : $R = (Y, U, c)\, c : U \to R^+ U \{ + \infty \}$ avec

$$Y = [X - (A_1 \cup A_2)] \times \{ e, S \} \cup \{ \alpha_1, \alpha_2 \}$$

On a contracté les groupes A_i en un sommet unique α_i et chaque nœud x donne naissance à deux sommets du réseau : (x, e) et (x, s) (e = entrée, s = sortie).

(x, e) reçoit tous les arcs entrant en x.

(x, s) reçoit tous les arcs sortant en x.

α_1 est la « source » et α_2 le « puits » du réseau. Voici un exemple pour illustrer cette transformation :

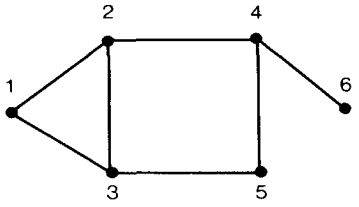


Figure 1 *a.*

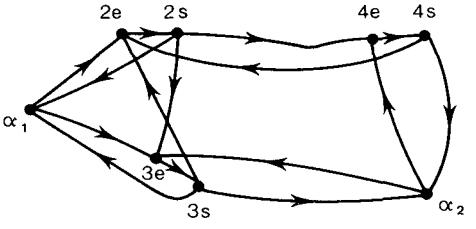


Figure 1 *b.*

$$A_1 = \{ 1 \} \qquad A_2 = \{ 5, 6 \} \qquad G = (X, E) \qquad R = (Y, U, c)$$

Tous les arcs du réseau ont une capacité $+ \infty$ sauf les arcs $((x, e), (x, s))$ qui ont une capacité égale à 1. La valeur maximum d'un flot entre α_1 et α_2 est d'après le théorème de Ford-Fulkerson la valeur minimum d'une coupe c'est-à-dire d'une famille d'arcs qui disconnecte le réseau en séparant α_1 et α_2. Cette capacité étant finie, les arcs de la coupe sont du type $((x, e), (x, s))$ et leur suppression équivaut dans G à la suppression du nœud x au lieu de l'arc $((x, e), (x, s))$. Toute coupe de R induit donc dans G un ensemble d'articulation, c'est-à-dire un ensemble de sommets dont la suppression coupe le graphe en deux. Il semble que l'on a traité un problème différent du problème abordé dans les autres méthodes mais on peut s'y ramener de la manière suivante. Soit $G = (X, E)$ un graphe, on construit le graphe $L(G) = (E, A)$ dont les nœuds sont les arêtes de G et où A contient les paires $\{ \alpha, \beta \}$ d'arêtes de G ayant dans G un sommet commun. Il est clair que si F est un ensemble d'articulation de $L(G)$, le graphe $G = (X, E - F)$ est disconnecté. Donc, étant donné un graphe G, on calcule son graphe associé $L(G)$ puis on transforme $L(G)$ en un réseau R. La coupe minimale de R donne un ensemble d'articulation de $L(G)$ et donc un graphe partiel disconnexe de G. Les composantes connexes sont les classes de la partition cherchée. Cette méthode est très rapide mais impose le choix du couple (A_1, A_2) a priori et se limite aux bipartitions, sauf transformations de l'énoncé.

II. ORGANISATION DE L'ALGORITHME

L'originalité de l'algorithme repose sur la division en deux phases successives :

Phase 1 : Réduction de la taille du problème

Phase 2 : Recherche de la solution optimale du problème réduit.

La phase 1 se décompose en deux procédures baptisées CONCENTR et ALPHAC.

La phase 2 comprend la procédure EXPLORE et la procédure NEST.

Il est plus commode d'exposer la phase 2 en premier.

II.1. Algorithme de réduction

II.1.1. *Recherche des* (θ, α)*-cliques maximales*

Soit $G = (X, E)$ un graphe. Une partie $A \subset X$ est appelée une clique si $\forall a, b \varepsilon \ a \neq b \Rightarrow \{ a, b \} \varepsilon E$. Une clique possède donc toutes les arêtes possibles. Nous allons généraliser la définition d'une clique.

DÉFINITION : Soit $\alpha \varepsilon [0, 1]$ un nombre réel. Une partie $A \subset X$ sera appelée une α-clique si et seulement si :

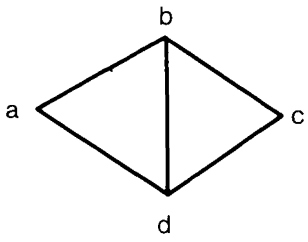
$$t(A) = \frac{2\lambda}{n(n-1)} = \alpha \qquad \text{où} \quad \begin{cases} \lambda = \text{nombre d'arêtes de } A \\ n = \text{Card } A \end{cases}$$

REMARQUE :

Soit un ensemble de n sommets et λ arêtes.

Comme le nombre maximum d'arêtes avec n sommets est $\frac{1}{2}n(n-1)$, on a $\dfrac{\lambda}{\frac{1}{2}n(n-1)} \leqslant 1$ l'égalité n'ayant lieu que dans le cas d'une clique. (Tous les sommets sont liés par des arêtes.)

Exemple :



$\frac{2.5}{4.3} = 0.833 = \alpha$

$A = \{ a, b, c, d \}$ est une 0.833-clique.

Figure 2.

Dans l'exemple : $t(A) = 0.833\ldots$ (Nous dirons que $t(A)$ est le taux de présence des arêtes).

Notons $m(x, A)$ = nombre d'arêtes de x vers un sommet de A.

Essayons d'évaluer $t(A \cup \{ x \})$. Avec Card $A = n$, Nombre d'arêtes de $A = \lambda$

$$t(A \cup \{ x \}) = \frac{2(\lambda + m(x, A))}{(n + 1)n}$$

Si on veut $t(A \cup \{ x \}) \geqslant \beta$ il faut :

$$2\lambda + 2m(x, A) \geqslant \beta n^2 + \beta n \tag{1}$$

Supposons que $t(A) \geqslant \beta$ alors :

$$2\lambda \geqslant \beta n^2 - \beta n \tag{2}$$

Soit en faisant (1)-(2)

$$m(x, A) \geqslant \beta \, . \, n$$

Ce critère permet de trouver récursivement des ensembles A qui sont des β-cliques maximales.

Si on exige des parties ayant au plus θ sommets, on incluera un test sur le nombre de sommets.

Avant de donner la procédure, nous allons généraliser ces définitions pour tenir compte du fait suivant : dans la suite, des réductions successives vont être opérées sur le graphe et les sommets du graphe réduit vont provenir de la contraction de plusieurs sommets du graphe initial ; de même les arêtes du graphe réduit seront pondérées pour conserver un maximum d'information sur le graphe original. Supposons donc que le graphe soit pondéré aux nœuds et aux arêtes par des entiers

$$G = (X, E, f, g) \quad \begin{bmatrix} f : X \to N \\ g : E \to N \end{bmatrix}$$

Alors on pose :

$$t(A) = 2 \sum_{\substack{\{x,y\} \in E \\ x, y \in A}} g(\{ x, y \}) / \left(\sum_{x \in A} f(x) \right) \left(\sum_{x \in A} f(x) - 1 \right)$$

Pour alléger l'écriture on posera :

$$\sum_{x \in A} f(x) = p(A) \qquad \text{(poids du sous-ensemble } A\text{)}$$

De plus, nous allons introduire un critère supplémentaire, appelé « liaison » et qui a été rencontré dans la « méthode des groupes minimaux ».

Soit A un sous-ensemble de X, nous poserons :

$$l(A) = \sum_{\substack{x \in A \\ y \in X - A \\ \{x,y\} \in E}} g(\{ x, y \})$$

Ce critère va intervenir pour choisir, en dernier ressort, entre les parties ayant le même « taux de présence » celle qui est le moins liée au reste du graphe.

Nous pouvons donner la procédure de recherche des ensembles que nous appelerons (α, θ)-cliques maximales et de liaison minimum.

Remarquons que : $t(E \cup \{ x \})$ maximum $\Leftrightarrow$ $m(x, E)$ maximum dans le cas où $f = 1$ et $g = 1$ sinon on a la relation plus complexe : si $n = p(E)$

$$t(E \cup \{ x \}) \text{ maximum} \Leftrightarrow \frac{2m(x, E)}{f(x)^2 + (2n - 1)f(x)} \text{ maximum}$$

en remplaçant $f(x)$ par 1

$$\frac{2m(x, E)}{1 + (2n - 1)} = \frac{m(x, E)}{n}$$

et comme n est fixé, $m(x, E)$ doit être maximum.

Voici l'organigramme de la procédure que nous nommerons ALPHA $(G, f, g, \theta, \alpha ; E)$ ou simplement ALPHA $(\alpha ; E)$.

(G, f, g) est le graphe valué.

$\alpha \in [0, 1]$ est fixé extérieurement.

A = la meilleure (α, θ)-clique maximale pour le critère l (ou l'ensemble $\emptyset$) telle que $p(A) \neq 0 \quad p(A) > 2$

En effet, les ensembles $A = \{ a, b \} \in E$ et tels que

$$f(a) = f(b) = 1 \qquad g(\{ a, b \}) = 1$$

vérifient

$$t(A) = \frac{2 \cdot 1}{2(2 - 1)} = 1 \text{ et sont donc des 1-cliques.}$$

Cependant ces ensembles ne sont pas très significatifs. Ils sont éliminés dans le test 5-*a* de l'organigramme ci-dessous.

ORGANIGRAMME DE LA PROCÉDURE ALPHA

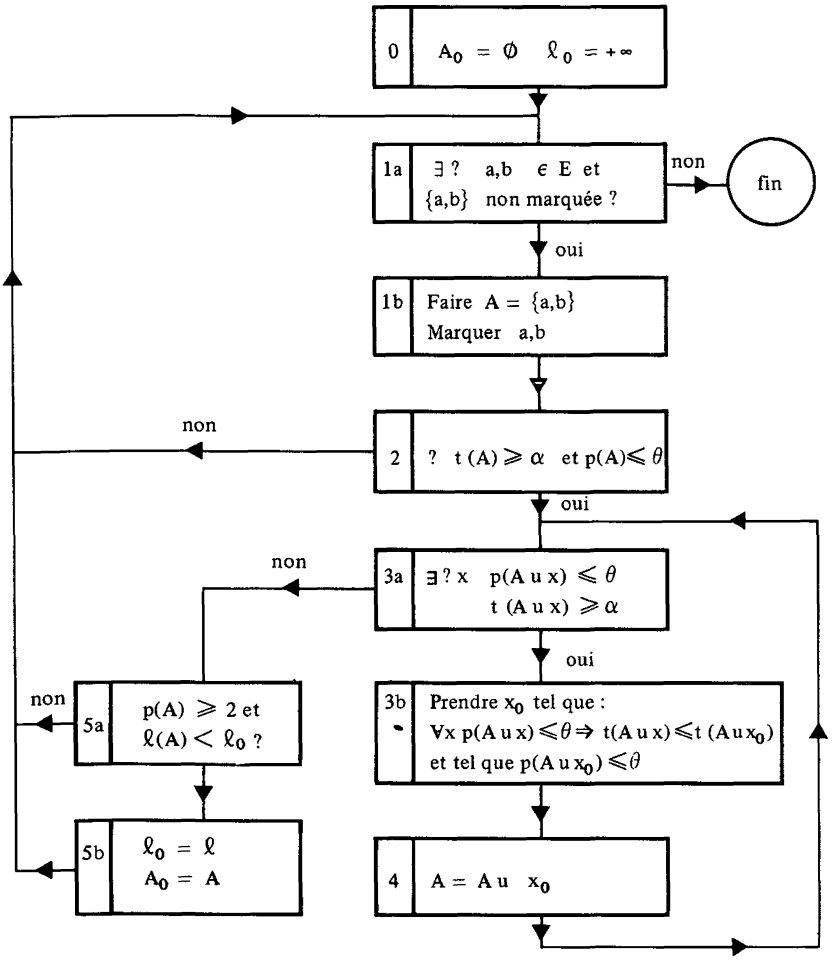


Figure 3.

COMMENTAIRES DE L'ORGANIGRAMME DE LA PROCÉDURE ALPHA

(0) Initialisation de A_0 et l_0
(1a) Recherche d'une arête non encore utilisée. Sert de TEST d'arrêt à la procédure
(1b) Initialisation de A
(2) TEST : « A » est-il une (α-θ)-clique?
(5a) (5b) (4) Calcul itératif d'une (α, θ)-clique maximale contenant A
(5a) TEST : « Va-t'on stocker cette (α, θ)-clique? »
(5b) Stockage de l'ensemble A.

II.1.2. *Procédure de contraction du graphe*

Soit A une (α, θ)-clique de $G = (X, E)$.

Supposons que $\alpha = 1$. Soit $P = (X_1, X_2, \ldots, X_l)$ une partition optimale de X; il semble raisonnable de penser que A sera contenue dans un X_i, autrement dit, les arêtes de A auront leurs extrémités dans une même classe de la partition.

Si α est « voisin » de 1, ce raisonnement est plausible.

L'idée de la contraction est triple.

(1) Transformer le graphe G ayant n sommets et m arêtes en graphe G' avec n' sommets et m' arêtes tel que :

$$n > n' \qquad \text{et} \qquad m > m'$$

(2) Garder invariante la structure des (α, θ)-cliques entre G et G', plus précisément, si $\phi : P(X) \to P(X')$ est l'application induite par l'application $\Pi : X \to X'$ de contraction définie plus loin.

On exige que :

$$t'[\phi(A)] = t(A)$$
$$p'[\phi(A)] = p(A)$$

(3) Si on a deux parties A et B disjointes et si l'on désire contracter G selon A et B, l'ordre est indifférent. Les applications p et t de $P(X) \to R^+$ sont construites à partir des applications f et g respectivement de $X \to N$ et de $E \to N$. Pour construire G' il faudra donc veiller à conserver les informations contenues dans f et g.

D'autre part, si on suppose qu'une partie A est contenue dans une classe de la partition, le problème n'est plus que de savoir dans quelle classe, ceci revient à considérer A comme un seul sommet de G'.

Il reste à statuer sur les arêtes incidentes à A dans G et sur les fonctions f' et g'.

Nous poserons :

$G = (X, E, f, g)$ A est une partie connexe de X

$G' = (X', E', f', g')$ sera le contracté de G selon A et nous noterons $G' = G/A$ avec :

$$X' = (X - A) \cup \{ a \} \qquad (\text{avec } a \notin X)$$

$$E' = \{ E \cap P_2(X - A) \} \cup E'' \qquad P_2(V) = \{ W \subseteq V / 1 \leqslant |W| \leqslant 2 \}$$

avec $E'' = \{ \{ x, a \} \mid x \in X ; \exists u \, u \in A ; \{ x, u \} \in E \}$

REMARQUE : E'' contient la boucle $\{ a \}$

$$f'(x) = \begin{cases} f(x) & x \in X - A \\ p(A) & x = a \end{cases}$$

$$g'(\{ x, y \}) = \begin{cases} g(\{ x, y \}) & (x, y) \in (X - A) \times (X - A) \\ \sum_{u \in A} g(\{ x, u \}) & \text{si} \quad y = a \text{ et } x \neq a \\ \sum_{\substack{u \in A \\ v \in A}} g(\{ u, v \}) & \text{si} \quad x = y = a \end{cases}$$

La figure suivante illustre une telle contraction :

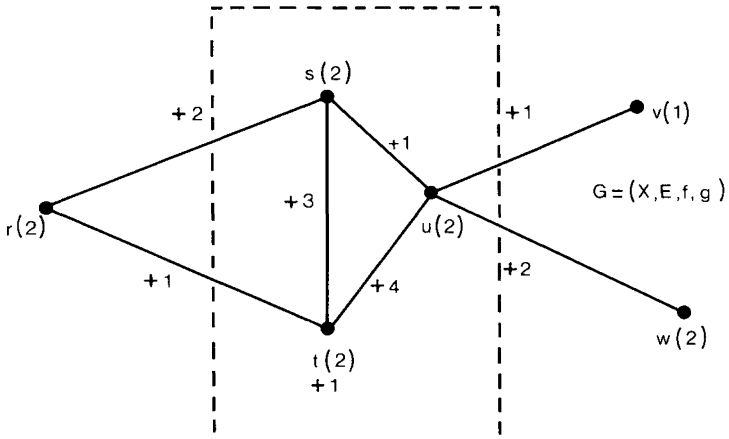


Figure 5 *a*

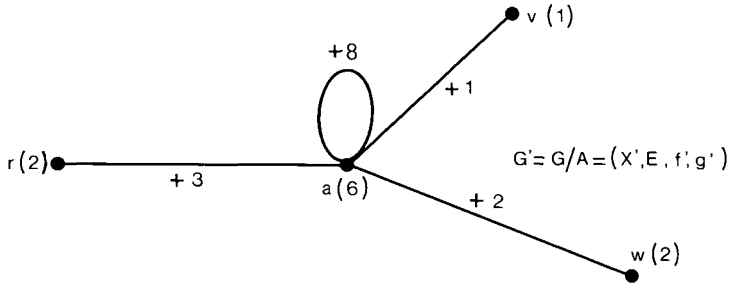


Figure 5 *b*

Légende : $A = \{ s, t, u \}$

Les nombres entre ( ) indiquent les poids des sommets ; les autres nombres indiquent les poids des arêtes.

Nous définissons maintenant $\Pi : X \to X'$ et $\emptyset : P(X) \to P(X')$ par :

$$\Pi(x) = \begin{cases} x & \text{si } x \in X - A \\ a & \text{si } x \in A \end{cases}$$

Remarquons que $\{ x, y \} \in E \Rightarrow \{ \Pi(x), \Pi(y) \} \in E'$ et donc que Π est un morphisme dans la catégorie des graphes.

On en déduit une définition de $\emptyset$ par :

$$\forall V \subseteq X \qquad \emptyset(V) = \{ \Pi(x) \mid x \in V \}.$$

Notons que :

$$f'(y) = \sum_{\Pi(x) = y} f(x) \tag{1}$$

$$g'(x, y) = \sum_{\Pi(t) = y} \sum_{\Pi(S) = x} g \{ S, t \} \tag{2}$$

(1) et (2) découlent des définitions même de f et g.

THÉORÈME 1 : *Conservation du taux de présence et du poids*

$$\forall V \subseteq X \qquad t'(\emptyset(V)) = t(V)$$
$$p'(\emptyset(V)) = p(V)$$

avec t' et p' définis comme t et p mais sur G'.

Preuve : Soit $V \subseteq X$. Avec (1) :

$$p'(\emptyset(V)) = \sum_{y \in \emptyset(V)} f'(y) = \sum_{\Pi(x) \in \emptyset(V)} f'[\Pi(x)] = \sum_{x \in V} f(x) = p(V)$$

Soit $n(V)$ le poids total des arêtes ayant leurs extrémités dans V :

$$n(V) = \sum_{\substack{x, y \in V \\ \{x, y\} \in E}} g \{ x, y \}$$

On définit de même $n'(V') = \sum_{\substack{x, y \in V' \\ \{x, y\} \in E'}} g' \{ x, y \}$

Comme pour la fonction p' on aura :

$$n'(\emptyset(V)) = \sum_{\substack{x, y \in \emptyset(V) \\ \{x, y\} \in E'}} g'(\{ x, y \}) = \sum_{\substack{s, t \in V \\ \{s, t\} \in E}} g'(\Pi(s), \Pi(t)) = \sum_{\substack{s, t \in V \\ \{s, t\} \in E}} g(s, t) = n(V)$$

Comme

$$t'(V') = \frac{2n'(V')}{p'(V')[p'(V') - 1]}$$

on a :

$$t'(\phi(V)) = \frac{2n'(\phi(V))}{p'(\phi(V))[p'(\phi(V)) - 1]} = \frac{2n(V)}{p(V)[p(V) - 1]} = t(V)$$

THÉORÈME : *Si l'élément remplaçant une partie A est a, B est b, etc... On a :* $(G/A)/B = (G/B)/A$ quels que soient $A, B \subseteq X$ avec $A \cap B = \phi$.

Preuve : On va construire un graphe noté $G/[A, B]$ qui sera isomorphe aux deux graphes précédents.

Notons :

Π_A la projection de $G \to G/A$

et Π_B la projection de $G \to G/B$

Soit alors $G/[A, B] = (X_1, E_1)$

$$X_1 = [X - (A \cup B)] \cup \{ a, b \} \qquad a, b \notin X$$

$$E_1 = \{ \{ \Pi_{A \cup B}(x), \Pi_{A \cup B}(y) \} \mid \{ x, y \} \in E \}$$

$$\Pi_{A \cup B}(x) = \begin{cases} x & x \in X - A \cup B \\ a & x \in A \\ b & x \in B \end{cases}$$

or : $\{ [(X - A) \cup \{ a \}] - B \} \cup \{ b \} = [X - A \cup B] \cup \{ a, b \}$

puisque $A \cap B = \phi \qquad A, B \subseteq X \quad$ et $\quad a, b \notin X.$

Donc $G/[A, B]$ et $(G/A)/B$ ont les mêmes sommets.

D'autre part, $\Pi_{A \cup B} = \Pi_{B/A} \circ \Pi_A$ si on décide de noter $\Pi_{B/A}$ l'application de

$$(X - A) \cup \{ a \} \to (X - (A \cup B)) \cup \{ a, b \}$$

définie par :

$$\Pi_{B/A}(x) = \begin{cases} x & \text{si} \quad x \notin B \\ b & \text{si} \quad x \in B \end{cases}$$

Alors on a bien égalité de l'ensemble des arêtes.

En fait, les valuations elles-mêmes sont conservées.

En résumé, (1) nous avons défini un procédé de contraction des graphes valués et deux fonctions : « poids » et « taux de présence », qui sont

compatibles; (2) le procédé de contraction est commutatif pour des parties disjointes de l'ensemble des sommets.

Les deux diagrammes suivants rendent compte de ces deux propriétés :

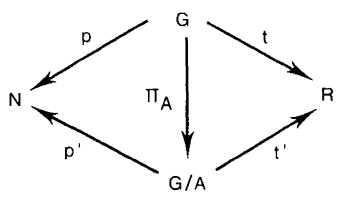


Figure 6 *a*

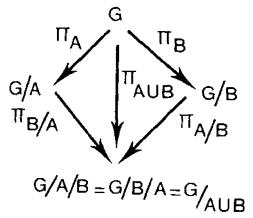


Figure 6 *b*

REMARQUE : Si on considère $j_A : A \to G$ l'injection canonique d'un sous-graphe dans son graphe et $k_A : A \to \{ a \}$ l'application constante on a bien $j_A \in \text{Mor}(A, G)$ et $k_A \in \text{Mor}(A, a)$ on peut alors vérifier que G/A est la somme amalgamée de G et $\{ a \}$ munis des morphismes j_A et k_A.

On a le diagramme commutatif :

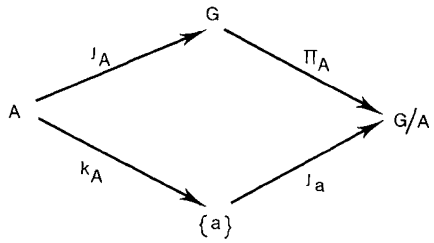


Figure 7.

II.1.3. *Organisation de la procédure de contraction*

Nous n'avons pas encore dit comment s'effectue le choix de α dans la procédure de contraction ni quand on doit arrêter cette procédure pour passer à la recherche d'une solution proprement dite.

En fait à chaque application d'une procédure ALPHA [α],

- soit on réduit la taille du problème P (en nombre d'arêtes par exemple);
- soit on ne la réduit pas.

Dans la première hypothèse on peut, si la taille cherchée n'est pas atteinte réappliquer ALPHA [α].

Dans la seconde il faut diminuer α ou bien cesser de réduire la taille du problème, car en-dessous d'un seuil α_{min}, les (α, θ)-cliques sont des ensembles non significatifs.

La figure suivante illustre l'organisation de ces choix :

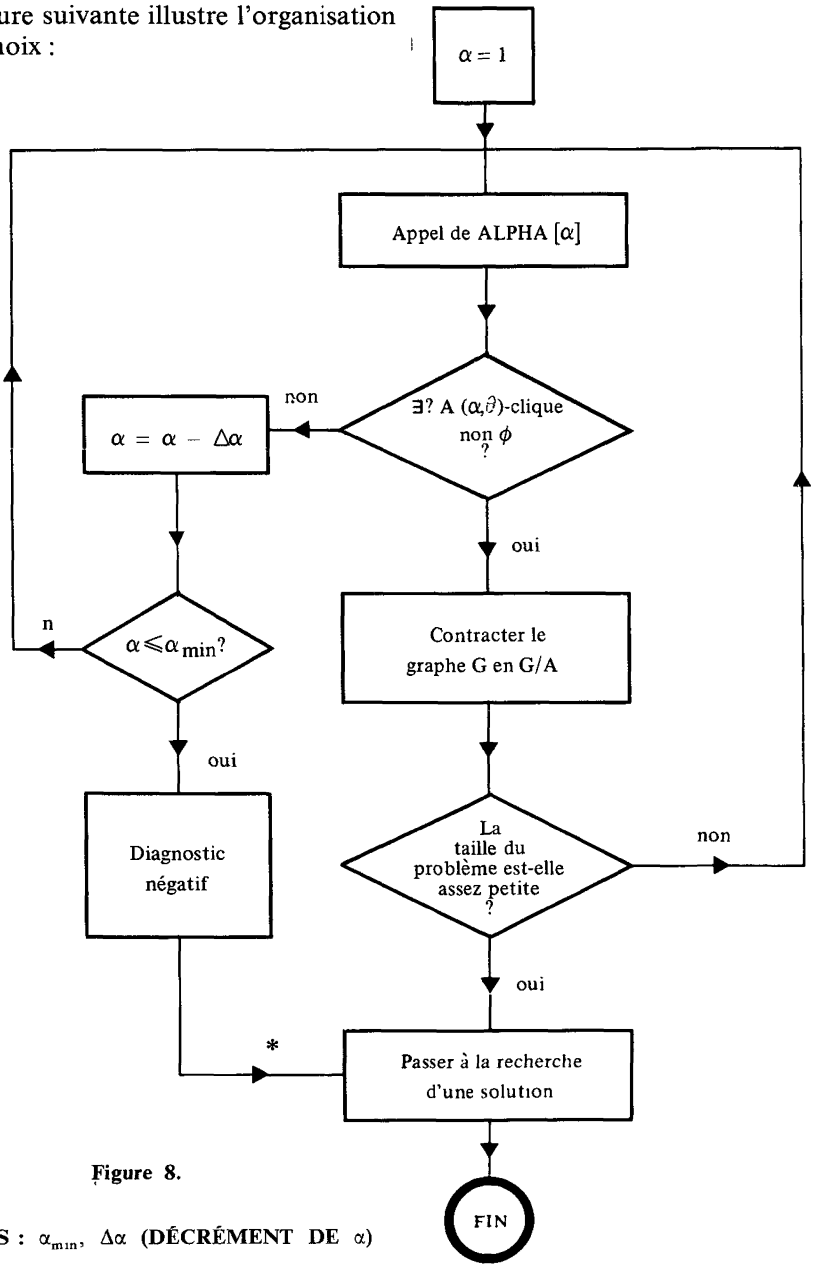


Figure 8.

ENTRÉES : α_{min}, $\Delta\alpha$ **(DÉCRÉMENT DE** α**)**

(*) Malgré le diagnostic on entreprend néanmoins la « Recherche d'une Solution » car il peut se faire que l'on obtienne ainsi une solution sous-optimale utilisable dans la suite.

II.2. Algorithme d'exploration

Nous allons exposer dans ce paragraphe la méthode de recherche de la solution optimale d'un problème réduit. Dans le premier sous-paragraphe nous donnons un test pour savoir si une solution potentielle vérifie la contrainte du problème. Nous exposons ensuite le codage des partitions adopté puis les procédures d'exploration.

II.2.1. *Test d'admissibilité*

Soit $G = (X, E, f, g)$ un graphe simple valué et $P = (X_1, X_2, \ldots, X_r)$ une partition de ses sommets.

Nous dirons que P est admissible si :

$$\forall i = 1, r \sum_{x \in X_i} f(x) \leqslant \theta$$

θ est un réel fixé et, on voit que cette condition peut aussi s'écrire :

$$\forall i = 1, r \qquad P(X_i) \leqslant \theta.$$

Le test doit donc nous dire si toutes les composantes connexes du graphe partiel, dont les arêtes sont celles qui ont leurs 2 extrémités dans un même X_i, ont un poids moindre que θ.

Nous allons :

1) Supprimer de $G = (X, E)$ toutes les arêtes joignant des sommets qui ne sont pas dans une même classe X_i ; on obtient un graphe $G' = (X, E')$.

2) Marquer tous les sommets de G' à l'aide d'une pile d'exploration.

Décrivons les règles de constitution de la pile :

Soit $A = (a_1, a_2, \ldots, a_h)$ la pile (hauteur $= h(A)$).

[I] Si $h(A) = 0$, prendre un sommet quelconque non marqué x et faire

$$a_1 = x \qquad h(A) = 1.$$

S'il n'y a plus de sommets non marqués, sortir avec un test = positif.

[II] Si $h(A) = h$, chercher a_{h+1} non marqué tel que $\{ a_{h-\lambda}, a_{h+1} \} \in E'$

$$0 < h \leqslant \theta \qquad \text{pour} \quad \lambda = 0, 1 \ \ldots, k - 1$$

Si aucun élément de la pile n'a de sommet voisin non marqué, détruire la pile et aller en I avec $h(A) = 0$.

Si $p(A) > \theta$, faire test = négatif et sortir.

Si on considère :

$$E'' = \{ \{ a_{h-\lambda}, a_{h+1} \} \mid h \geqslant 0 \}$$

On a $E'' \subseteq E'$ et $G'' = (X, E'')$ est un graphe partiel de $G' = (X, E')$. D'autre part, G'' ne possède pas de cycle car un sommet n'est marqué qu'une fois.

Rappelons qu'un arbre est un graphe connexe sans cycle. Une réunion d'arbres disjoints est une forêt. Ici, le graphe $G'' = (X, E'')$ est un élément maximal dans l'ensemble des forêts de G' car il comprend tous les nœuds de G'.

On voit donc que le test consiste à construire une forêt maximale de G. Si chaque arbre de cette forêt a une taille inférieure à θ, le test est positif.

L'intérêt de cette remarque est le suivant : si on enlève une arête à un arbre, on le disconnecte, donc une forêt maximale est le graphe partiel le plus petit qu'il faille engendrer pour ce test (qui nécessite la création de toutes les composantes connexes). D'autre part toutes les forêts maximales d'un graphe ayant le même nombre d'arêtes, l'exploration aura donc une longueur indépendante des choix arbitraires qui y sont faits.

En résumé, ce test répond à la question : « La partition P est-elle admissible ? » de façon optimale.

II.2.2. *Procédure d'exploration*

La procédure EXPLORE est une méthode de résolution du problème $P(X, E, f, g)$ par exploration d'un sous-ensemble de l'ensemble des partitions de X. Nous donnons d'abord un codage pratique de ces partitions.

Dans le paragraphe suivant, le théorème 1 établit l'exhaustivité d'une procédure de marquage des mots codes de ces partitions.

Enfin, nous montrons que la procédure EXPLORE marque tous les mots codes sauf éventuellement des mots codes de partitions moins bonnes (au sens du critère P) que celle trouvée dans la procédure.

II.2.2.1. Codage

Soit $F \subseteq E$, on note $G(F) = (X, F)$ le graphe partiel de G ayant E pour ensemble d'arêtes.

Nous allons coder les parties S de E à l'aide de mots de longueur h sur l'alphabet $\{ T, F \}$.

Soit C l'ensemble de ces mots :

$$M \in C \Leftrightarrow M = L_1 L_2 \ldots L_h \qquad \text{et} \qquad \forall i \, L_i \in \{ T, F \}.$$

Soit donc S une partie de $E = \{ e_1, \ldots, e_h \}$ on définit un mot M de C par :

$$M = L_1 L_2 \cdots L_h \qquad \text{et} \qquad \forall i = 1, h \, L_i = \begin{cases} F & \text{si} \quad e_i \in S \\ T & \text{si} \quad e_i \notin S \end{cases}$$

Il est clair que la correspondance $S \leftrightarrow M$ est un codage de $P(E)$ dans C.

Soit une partie S de E et $G(S) = (X, S)$ le graphe partiel de G associé à S.

Les composantes connexes de $G(S)$ sont des parties connexes de G; en effet si a et b sont reliés par une chaîne d'arêtes de S et puisque $S \subseteq E$, a et b seront reliés par une chaîne dans G.

On peut donc, à toute partie S de E associer la partition de X en composantes connexes de $G(S)$, chaque classe de la partition étant connexe dans G.

On sait ainsi associer à un mot $M \in C$ une partition $P(M) = (X_1, \ldots, X_r)$ de X en parties connexes de G.

Inversement, soit $Q = (Y_1, Y_2, \ldots, Y_r)$ une partition de X en parties connexes de G. Soit S l'ensemble des arêtes de E ayant leurs deux extrémités dans un même Y_i.

Considérons enfin la partition $P = (X_1, \ldots, X_S)$ de X en composantes connexes de $G(S)$. On a :

$$P = Q.$$

En effet soit $\gamma = [x_1, x_2, \ldots, x_n]$ une chaîne de $G(S)$. γ est aussi une chaîne de G et par construction de S, cette chaîne est contenue dans un Y_i.

Les composantes X_i contiennent donc des réunions de Y_j. Supposons que Y_i ne soit pas une composante connexe de $G(S)$, ceci signifie qu'il existe $y \notin Y_i$ et $y' \in Y_i$ avec une chaîne de y à y' dans $G(S)$. Chaque chaîne de $G(S)$ est contenue dans un Y_i et donc notre hypothèse est impossible. Les Y_i étant des composantes connexes de $G(S)$ et contenues dans des X_j qui en sont aussi on a bien :

$$Q = P.$$

Conclusion :

A tout mot M de C on peut associer une partition de X en parties connexes de G.

A toute partition de X en parties connexes de G on sait associer au moins une partie de S de E et donc un mot M de C.

Un mot M sera dit admissible si la partition $P(M)$ associée vérifie la condition 1 du problème P c'est-à-dire :

$$P(M) = (X_1, \ldots, X_r) \quad \forall i = 1, r \sum_{x \in X_i} f(x) \leqslant \theta$$

Nous définissons le poids d'un mot par :

$$\Pi(M) = \sum_{i \mid L_i = T} g(u_i) = \mathscr{C}(X_1, X_2, \ldots, X_k)$$

(où $P = (X_1, \ldots, X_k)$ est la partition associée à M).

Par convention, le mot vide ($\emptyset$), c'est-à-dire de longueur nulle, a un poids :

$$\Pi(\emptyset) = 0.$$

II.2.2.2. EXPLORE

Le problème $P(X, U, f, g, \theta)$ revient à chercher un mot M_{OPT} admissible de longueur h et de poids minimum Π_{OPT}. Il nous faut pour cela une procédure de construction de mots.

Règles de dérivations sur l'ensemble des mots :

$$\begin{aligned} M &= L_1 L_2 \quad\quad L_{k-1} L_k \\ M_1 &= L_1 L_2 \ . \ L_{k-1} \text{« } T \text{»} \\ \textit{Père}\ (M) &= L_1 L_2 \cdot \quad L_{k-1} \\ \textit{Fils}\ (M) &= L_1 L_2 \quad\quad L_{k-1} L_k \text{« } T \text{»} \\ \textit{Sœur}\ (M_1) &= L_1 L_2 \quad . \ L_{k-1} \text{« } F \text{»} \end{aligned}$$

On note $[M, M']$ le mot obtenu par concaténation de M et M'.

Nous introduisons aussi un ensemble de mots :

$$\mathcal{M}_h(M_0) = \{\ [M_0, M];\ \text{longueur}\ (M) \leqslant h\ \}$$

La procédure PROC 1 a pour objet d'explorer tous les mots de $\mathcal{M}_h(M_0)$ c'est-à-dire les mots possédant M_0 comme facteur gauche, de longueur majorée par longueur $(M_0) + h$.

Si on convient que : $T \leqslant F$ cet ordre sur l'alphabet induit un ordre lexicographique sur les mots.

Par exemple $TFT \leqslant TFF$.

C'est dans cet ordre lexicographique que seront marqués les mots de $\mathcal{M}_h(M_0)$.

Soit maintenant une procédure PROC 1 de marquage définie par :

[PROC 1] Entrées : M_0 ; h

[1] $M \leftarrow M_0$
[2] Marquer M
[3] Si ($\textit{Fils}\ (M) \notin \mathcal{M}_h(M_0)$ ou $\textit{Fils}\ (M)$ marqué), aller en [4], sinon $M \leftarrow \textit{Fils}\ (M)$ et aller en [5]
[4] Si ($\textit{Sœur}\ (M) \in \mathcal{M}_h(M_0)$), $M \leftarrow \textit{Sœur}\ (M)$;
sinon $M \leftarrow \textit{Père}\ (M)$
[5] Si ($M = M_0$), aller en [6]
sinon : Aller en [2]
[6] Stop

THÉORÈME 1 : *PROC 1 marque tous les éléments de $\mathcal{M}_h(M_0)$ en un nombre fini d'étapes.*

Démonstration : Nous allons démontrer le théorème par récurence :

Soit la proposition

$$H(n) \Leftrightarrow \left\{ \forall M_0 \quad \begin{array}{l} \text{PROC 1 marque tous les mots de } \mathcal{M}_n(M_0) \\ \text{dans l'ordre lexicographique} \end{array} \right.$$

$H(0)$ est évidente. Supposons que $H(n)$ établie et prouvons que $H(n+1)$ est vraie.

a) $M_{n+1}(M_0) = \{ M_0 \} \cup M_n(M_1) \cup M_n(M_2)$
avec $M_1 = \text{Fils}(M_0) = M_0 T$
$M_2 = \text{Sœur}(M_0) = M_0 F$

En effet :

$M \in \mathcal{M}_{n+1}(M_0) \Leftrightarrow M = [M_0, M']$ et longueur $(M') \leqslant n+1$
– si longueur $(M') = 0$, on a $M = M_0$
– si longueur $(M') = 1$, on a $M = M_0 L M'$ avec $L \in \{ T, F \}$
$L = T \Rightarrow M \in \mathcal{M}_n(M_1)$
$L = F \Rightarrow M \in \mathcal{M}_n(M_2)$

b) Après avoir marqué M_0, PROC 1 va marquer M_1. D'après l'hypothèse $H(n)$ appliquée à $\mathcal{M}_n(M_1)$, PROC 1 marque tous les éléments de $\mathcal{M}_n(M_1)$ et le dernier étant « $M_1 F^n$ », il n'a pas de fils et la PROC 1, [4], aboutit de nouveau en M_1 dont tous les descendants sont marqués. Le [4] fait alors marquer $M_2 = \text{Sœur}(M_1)$.

c) Comme pour $\mathcal{M}_n(M_1)$, $\mathcal{M}_n(M_2)$ est marqué dans l'ordre lexicographique. Le dernier mot « $M_2 F^n$ » étant marqué et M_2 n'ayant pas de sœur, PROC 1 remonte à M_0.

d) On a de plus $\forall X \in \mathcal{M}_n(M_1) \, \forall Y \in \mathcal{M}_n(M_2)$ « X précède Y dans l'ordre lexicographique » car :

$$X = M_0 T M' \qquad \text{et} \qquad Y = M_0 F M''.$$

Conclusion

Tous les mots de $\mathcal{M}_{n+1}(M_0)$ ont été marqués dans l'ordre lexicographique et la récurence est terminée.

COROLLAIRE

La procédure PROC 1 appliquée avec $M_0 = \emptyset$ marque tous les mots de longueur inférieure ou égale à h. C'est le cas que nous considérons dans la suite, mais le théorème, plus général, est plus facile à établir.

Définissons maintenant une nouvelle procédure : PROC 2

[PROC2] Entrées : M_0 ; h ; Π_0
]1] $\Pi \leftarrow \Pi_0$; $M \leftarrow M_0$
[2] Marquer M ; Si : (long $(M) = n$ et M admissible)
Faire : $\Pi \leftarrow \Pi(M)$; aller en [3]
[3] Si : (Fils $(M) \notin \mathcal{M}_h(M_0)$ ou Fils (M) marqué ou $\Pi(\text{Fils}(M)) > \Pi_0$) ; Aller en [4]
Sinon : $M \leftarrow$ Fils (M) et aller en [5]
[4] Si : (Sœur $(M) \in \mathcal{M}_h(M_0)$) ; Alors : $M \leftarrow$ Sœur (M)
Sinon : $M \leftarrow$ Père (M)
[5] Si : $(M = M_0)$ allez en [6]
Sinon : allez en [2]
[6] Stop.

THÉORÈME 2 : *La procédure PROC 2 marque tous les mots de* $\mathcal{M}_h(M_0)$ *sauf des mots M de poids supérieur à un poids optimal,* Π_{OPT} *d'un mot* M_{OPT} *dont la partition associée est solution du problème* $P(X, U, f, g, \theta)$.

Démonstration :

Si un mot M n'est pas marqué, il existe un des ascendants $\bar{M}$ (M éventuellement) qui vérifie : $\Pi(\bar{M}) > \Pi_0$ dans la règle [3] de PROC 2. En effet, le théorème 1 garantit le marquage par PROC 1 de tous les mots de $\mathcal{M}_h(M_0)$, la seule différence étant le test $\Pi(\bar{M}) > \Pi_0$, celui-ci a fonctionné sur au moins un ascendant de M.

D'autre part :

$$\Pi(M) \geqslant \Pi(\bar{M}) \quad (\text{car } g \text{ est à valeur positive})$$

Donc :

$$\Pi(M) > \Pi_0.$$

Nous savons d'autre part que la séquence des « Π_0 » est décroissante strictement et que Π_{OPT} est le dernier terme de cette séquence. D'où :

$$\Pi(M) > \Pi_0 \geqslant \Pi_{OPT}$$

Il est clair que le mot M_{OPT} dont Π_{OPT} est le poids fournit la solution optimale du problème $P(X, U, f, g, \theta)$.

II.2.3. *Exemple*

Nous avons traité l'exemple représenté à la page suivante. Le nombre de sommets est 44, le nombre d'arêtes est 62. On exige $\theta = 15$ (taille maximum de chaque classe). La solution trouvée donne 3 classes et une fonction coût égale à 10 (nombre d'arêtes coupées).

Le temps de calcul sur IBM 370/165 de 9 secondes au maximum lorsque la procédure EXPLORE travaille avec 20 arêtes et de 1,3 secondes au maximum si EXPLORE travaille avec 15 arêtes. Nous donnons des temps maximum au sens où le temps de calcul croît avec la valeur initiale de la fonction coût M_0. On peut la calculer dans le « plus mauvais des cas ».

Si $g_{max} = \underset{(x,y)}{\text{Sup}}\, g(x, y)$ et d_{max} le nombre maximum de voisins d'un sommet. Alors :

$$M_0 = n \times g_{max} \times (d_{max} - 1) \quad (\text{où } n = |X|).$$

Dans cet exemple $n = 44$ $g_{max} = 1$ $d_{max} = 4$, donc $M_0 = 132$.

Les 3 figures suivantes exposent les résultats obtenus. Sur la figure 9 nous avons entouré d'un trait pointillé les (α, θ)-cliques utilisées dans la phase de concentration (procédure ALPHA).

Sur la figure 10 se trouve le graphe réduit sur lequel la procédure EXPLORE va travailler. Les nouveaux sommets sont cerclés et le poids de chaque sommet est indiqué ainsi que celui de chaque arête.

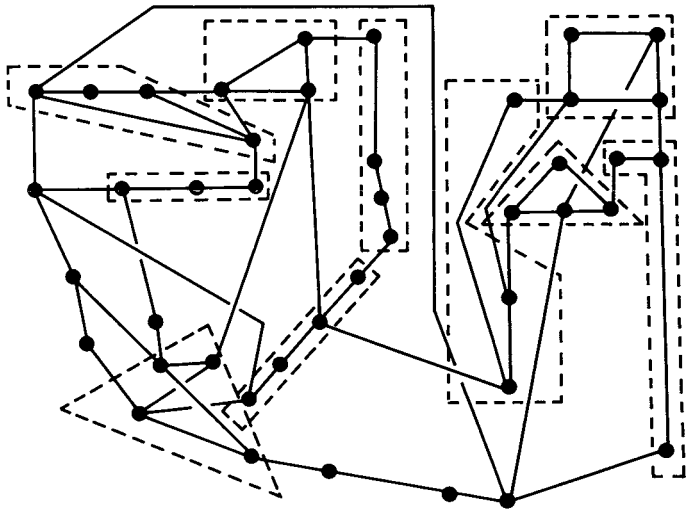


Figure 9.

Ce graphe schématise une usine de production d'acide sulfurique. Il est extrait d'un article de T. K. Pho et L. Lapidus [8]. La décomposition obtenue semble pertinente, une analyse plus fine reste nécessaire. Le paramètre α a varié de 1 à 0.5.

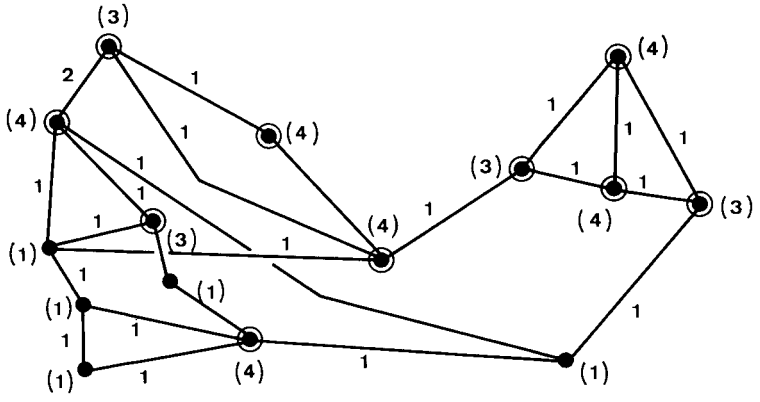


Figure 10.

Enfin sur la figure 11 nous avons repris le graphe initial en indiquant la partition optimale obtenue; (3 classes) les arêtes en trait discontinu sont celles qui joignent deux classes distinctes.

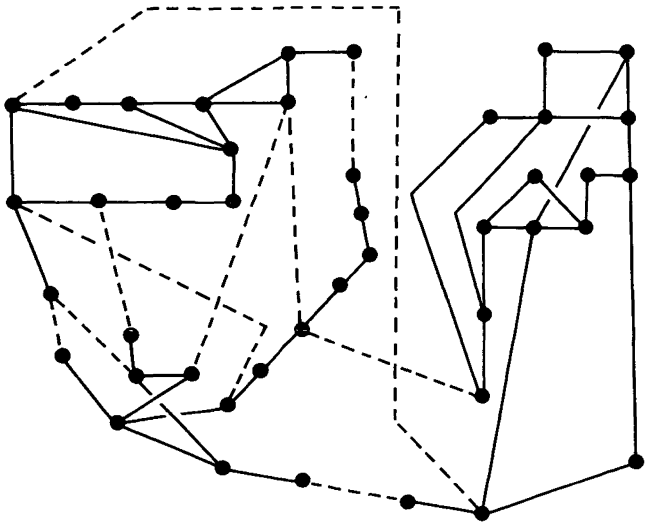


Figure 11.

III. CONCLUSION

Nous avons présenté un algorithme heuristique qui résout un problème de nature combinatoire. La solution trouvée est optimale si la phase de réduction de l'algorithme n'est pas utilisée. Le volume de mémoire utilisé est approximativement donné par la formule :

$$V(n) = 150 + n^2 \,.\, 10^{-2} \text{ en K octets}$$

où n est le nombre de sommets du graphe.

L'évolution du temps de calcul est exponentielle avec une valeur de l'ordre de 2 secondes pour des graphes de 30 sommets et de l'ordre de 180 secondes pour des graphes de 130 sommets.

Nombre de sommets du graphe	20	30	45	126
Temps de calcul en secondes	1,2	2	9	180

Cet algorithme est très souple d'utilisation, il peut être très facilement modifié pour tenir compte des contraintes et/ou des choix « a priori » les plus variés. Actuellement, nous envisageons de développer une version utilisable « en temps réel » pour étudier l'évolution temporelle de la décomposition.

BIBLIOGRAPHIE

1. C. BERGE, *Théorie des graphes et ses applications*, Dunod, 1956.
2. C. BERGE, *Graphes et Hypergraphes*, Dunod, 1970.
3. W. E. DONATH-A. J. HOFFMAN, *Algorithms for partitioning of graphs and computer logic based on eigen vectors of connection matrices*, IBM Technical Report.
4. G. I. ORLOVA, *Optimum partition of a graph into several subgraphs* Engineering Cybernetic, Vol. **10**, 103, 1972.
5. F. LUCCIO-M. SAMI, *On the decomposition of networks into minimally interconnected subnetworks*, IEE transactions on Circuit Theory, CT-16, Vol. **2**, 1972.
6. B. W. KERNIGHAN, *An effective heuristic procedure for partioning graphs*, BSTJ, Vol. **49**, 291, 1970.
7. M. RICHETIN, *Algorithme de décomposition optimale et sous-optimale des graphes*, Notes internes CH-LAAS 73 1 33.
8. T.K. PHO-L. LAPIDUS, *An optimum Tearing for recycle systems*, AIChE Journ.; Vol. 19, n° 6, p. 1170, 1973.