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

MICHEL SCHNEIDER

Méthodes pour recenser toutes les cliques maximales et θ-maximales d'un graphe

Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle, tome 7, n° V3 (1973), p. 21-33

<http://www.numdam.org/item?id=RO_1973__7_3_21_0>

R.A.I.R.O.
(7e année, septembre 1973, V-3, p. 21 à 33)

METHODES POUR RECENSER TOUTES LES CLIQUES MAXIMALES ET θ-MAXIMALES D'UN GRAPHE (1)

par Michel SCHNEIDER (2)

Résumé. — *Nous présentons dans cet article deux méthodes pour recenser toutes les cliques maximales et θ-maximales d'un graphe. La première méthode, quoique faisant appel à une technique classique d'exploration, est présentée sous une forme très générale qui la rend apte au recensement de plusieurs catégories remarquables de sous-ensembles maximaux d'un graphe. La deuxième méthode qui repose sur la notion de sous-graphe presque complet est beaucoup plus performante que la précédente lorsque le nombre de sommets par clique maximale est proche du nombre de sommets du graphe.*

INTRODUCTION

Soit un graphe $G = (X, \Gamma)$ où X est l'ensemble des sommets et Γ la relation binaire associée. On appelle clique dans G un sous-graphe de G dont tous les sommets sont adjacents entre eux. Une clique C_1 est dite supérieure, égale, ou inférieure à la clique C_2 suivant que le nombre de sommets de C_1 est supérieur, égal, ou inférieur au nombre de sommets de C_2. Une clique est dite maximale s'il n'existe aucune clique la contenant strictement. Les cliques maximales d'un graphe G n'ont pas toutes le même nombre de sommets. Le nombre maximum de sommets de G susceptibles de former une clique étant noté $\theta(G)$ on appellera clique θ-maximale une clique comprenant θ sommets.

Soit $\bar{G} = (X, \bar{\Gamma})$ le graphe complémentaire de G dans lequel deux sommets sont adjacents si et seulement s'ils ne le sont pas dans G. L'ensemble des cliques de G coïncident avec l'ensemble des sous-ensembles stables de $\bar{G}$. Il s'ensuit que tout problème concernant les cliques de G peut être traité comme un problème concernant les sous-ensembles stables de $\bar{G}$ et récipro-

(1) Ce travail a été réalisé au Département d'Informatique de l'Université de Montréal, Canada.

(2) Département de Mathématiques Appliquées, Université de Clermont-Ferrand, Aubière.

quement. Or la notion de sous-ensemble stable présente un grand intérêt dans de très nombreux problèmes concrets et a de ce fait suscité de nombreux travaux dont on trouvera l'essentiel dans [1] et [4]. Des algorithmes permettant de déterminer les sous-ensembles stables maximaux apparaissent dans [1], [2], [3], [4], [5].

La notion de cliques sans être aussi féconde se révèle cependant très utile dans tous les domaines où l'on considère une relation exprimant une similitude ou une hiérarchie et plus particulièrement dans les domaines suivants : Théorie des jeux, Systèmes de classification, Étude des réseaux. Mais la recherche directe des cliques n'a pas fait l'objet d'études extensives, sans doute à cause de la propriété de complémentarité signalée précédemment. Toutefois il n'est pas utile de recenser toutes les cliques du graphe, ce qui serait souvent fastidieux, puisque la seule connaissance des cliques maximales permet de déterminer toutes les autres cliques. De plus il apparaît qu'au niveau des applications la connaissance des cliques θ-maximales est très souvent suffisante. C'est pourquoi nous voudrions proposer deux méthodes pour recenser toutes les cliques maximales et θ-maximales d'un graphe.

La première méthode fait appel à une technique d'exploration désormais classique ([4], [5]). Nous nous sommes surtout attachés à en présenter une formulation générale adaptable à une large classe de problèmes concernant la détermination de sous-ensembles maximaux remarquables d'un graphe. Elle conduit à un algorithme particulièrement bien adapté à la recherche des cliques θ-maximales.

La deuxième méthode qui repose sur la notion de sous-graphe presque complet est à notre connaissance originale. L'algorithme correspondant autorise des temps d'exécution très inférieurs à ceux de l'algorithme précédent lorsque le nombre de sommets par clique maximale est proche du nombre de sommets du graphe.

1. METHODE DE RECHERCHE DIRECTE

1.1. Présentation générale

Soit un graphe $G = (X, \Gamma)$ et une propriété $\mathcal{P}$ définie sur X relativement à Γ. Soit V un sous-ensemble de X. $\mathcal{P}(V)$ est dite vraie si V vérifie $\mathcal{P}$ fausse sinon. Pour la suite la propriété $\mathcal{P}$ sera astreinte à suivre la condition suivante :

Condition 1 : Pour tout $V \subset X$ tel que $\mathcal{P}(V)$ est vraie, alors pour tout $V' \subset V$, $\mathcal{P}(V')$ est aussi vraie.

Problème : Déterminer tous les sous-ensembles de X pour lesquels la propriété $\mathcal{P}$ est vraie.

Définition 1 : $V \subset X$ sera dit $\mathfrak{T}$-majorant si

1) $\mathfrak{T}(V)$ est vraie

2) $\mathfrak{T}(V \cup \{ \alpha \})$ est fausse $\forall \alpha \in X - V$.

Compte tenu de cette dernière définition, le problème précédent se ramène à la recherche des sous-ensembles $\mathfrak{T}$-majorants de X.

Si G est symétrique et si $\mathfrak{T}(V)$ est vraie si et seulement si tous les sommets de V sont adjacents entre eux alors la recherche des sous-ensembles $\mathfrak{T}$-majorants est une recherche des cliques maximales.

Théorème 1

Numérotons les N éléments de X par les entiers de 1 à N. Soit un sous-ensemble $V_1 = \{ \alpha_1, \alpha_2, ..., \alpha_p \}$ de X où

1) α_1 est un élément de X tel que $\mathfrak{T}(\{ \alpha_1 \})$ est vraie

2) $\alpha_{i+1} (1 \leqslant i \leqslant p)$ est le premier élément après α_i dans l'ordre de numérotation tel que $\mathfrak{T}(\{ \alpha_1, \alpha_2, ..., \alpha_i, \alpha_{i+1} \})$ est vraie

3) il n'existe aucun élément β après α_p dans l'ordre de numérotation tel que $\mathfrak{T}(V_1 \cup \{ \beta \})$ est vraie

4) il n'existe aucun élément γ avant α_1 dans l'ordre de numérotation tel que $\mathfrak{T}(V_1 \cup \{ \gamma \})$ est vraie.

Alors V_1 est un sous-ensemble $\mathfrak{T}$-majorant.

La démonstration est simple. Supposons qu'il soit possible d'adjoindre un autre élément β à V_1 tel que $\mathfrak{T}(V_1 \cup \{ \beta \})$ soit vraie. β n'est dans l'ordre de numérotation, ni avant α_1, ni après α_p d'après les points 3) et 4). β se place donc entre les éléments α_k et $\alpha_{k+1} (1 < k < p)$ de V_1, ce qui contredit le point 2).

1.2. Recensement des sous-ensembles $\mathfrak{T}$-majorants de X appartenant à des classes remarquables

1.2.1. Recensement des sous-ensembles $\mathfrak{T}$-majorants de X contenant l'élément α

On pourra toujours supposer sans restreindre la généralité du problème que α est l'élément numéro 1. Un sous-ensemble $\mathfrak{T}$-majorant peut être obtenu en faisant au départ $V_\alpha = \{ \alpha \}$ et en appliquant de proche en proche la règle suivante.

Règle 1 : Faire $V_\alpha = V_\alpha \cup \{ \beta \}$ chaque fois que le premier élément (dans l'ordre de numérotation) non examiné est tel que $\mathfrak{T}(V_\alpha \cup \{ \beta \})$ est vraie.

Lorsque le dernier élément de X a été examiné, V_α respecte en tout point le théorème 1 et est donc $\mathfrak{T}$-majorant. On recherchera un autre sous-ensemble $\mathfrak{T}$-majorant en appliquant la règle 2.

Règle 2 :

— supprimer le dernier élément (appelons le γ) adjoint à V_α;

— appliquer à nouveau la règle 1 à partir de l'élément suivant immédiatement γ dans l'ordre de numérotation.

Si de nouveaux sommets ont été adjoints à V_α à l'aide de la règle 1 soit δ le sommet précédent immédiatement γ dans V_α. S'il n'existe aucun β tel que $\delta < \beta \leqslant \gamma$ et que $\mathfrak{F}(V_\alpha \cup \{ \beta \})$ soit vraie alors le nouveau V_α ainsi obtenu est $\mathfrak{F}$-majorant.

Il est inutile d'examiner les cas $\beta \leqslant \delta$ et $\beta > \gamma$ car d'après la constitution de V_α il n'existe aucun β satisfaisant simultanément les trois conditions :

i) $\beta \leqslant \delta$ ou $\beta > \gamma$

ii) $\beta \neq w \quad \forall w \in V_\alpha$

iii) $\mathfrak{F}(V_\alpha \cup \{ \beta \})$ vraie.

La règle 2 peut être appliquée tant que V_α contient au moins deux éléments. Lorsque V_α se réduit à $\{ \alpha \}$ on a épuisé toutes les possibilités et recensé tous les sous-ensembles $\mathfrak{F}$-majorants contenant α.

D'où l'algorithme suivant :

Algorithme 1 : Recensement des sous-ensembles $\mathfrak{F}$-majorants de X contenant le premier élément de X.

1) Créer une pile avec le premier sommet x_1.

Faire $\alpha = x_1$, $\beta = x_1$ et $y = x_1$. Passer en 2).

2) Placer sur la pile le premier sommet z venant après y dans l'ordre de numérotation et tel que l'ensemble $E \cup \{ z \}$ vérifie $\mathfrak{F}$, E désignant l'ensemble des sommets contenus dans la pile. Itérer cette procédure tant qu'il reste des sommets non encore examinés en faisant $y = z$. Si l'on a pu ainsi faire progresser la pile passer en 3) sinon passer en 5).

3) Soit E l'ensemble des sommets contenus dans la pile lorsqu'on parvient à ce stade. Dans le cas où $\alpha \neq \beta$ tester s'il existe un sommet γ compris entre α et β ou égal à β tel que l'ensemble $E \cup \{ \gamma \}$ vérifie $\mathfrak{F}$. Si la réponse est oui passer en 4). Si la réponse est non ou si $\alpha = \beta$, E est un ensemble $\mathfrak{F}$-majorant; passer alors en 5).

4) Soit t le sommet qui figure sur la pile lorsqu'on parvient à ce stade. Faire $y = t$, retirer ce sommet de la pile et retourner en 2).

5) Si la pile contient au moins deux éléments passer en 6) sinon tous les sous-ensembles $\mathfrak{F}$-majorants contenant x_1 ont été obtenus.

6) Soit t le sommet qui figure sur la pile lorsqu'on parvient à ce stade. Faire $y = t$ et $\beta = t$. Retirer alors ce sommet de la pile. Soit v le nouveau sommet qui figure sur la pile. Faire $\alpha = v$ et retourner en 2).

1.2.2. Recensement de tous les sous-ensembles $\mathfrak{F}$-majorants de X

Les sous-ensembles $\mathfrak{F}$-majorants de X peuvent être rangés en N classes ou la j-ième classe est définie comme la classe des sous-ensembles $\mathfrak{F}$-majorants qui contiennent le sommet de numéro j et qui ne contiennent pas les sommets de numéros 1, 2, ..., $j-1$.

Pour recenser les sous-ensembles $\mathfrak{F}$-majorants de la classe j on pourra utiliser l'algorithme 1 convenablement modifié :

— au départ au stade 1) il faut créer la pile avec le sommet d'indice j, où j est l'indicatif de la classe;

— au stade 4) il faut vérifier s'il existe un sommet γ de numéro inférieur à j tel que $\mathfrak{F}(E \cup \{\gamma\})$ est vraie. Si un tel sommet existe E n'est pas $\mathfrak{F}$-majorant.

L'algorithme 2 tient compte de ces modifications et comprend une boucle pour examiner les N classes.

Algorithme 2 : Recensement de tous les sous-ensembles $\mathfrak{F}$-majorants de X.

1) Faire $I = 1$ et passer en 2).

2) Si I est supérieur à N on a obtenu tous les sous-ensembles $\mathfrak{F}$-majorants de X; sinon créer une pile avec le sommet x_I puis faire $\alpha = x_I$, $\beta = x_I$, $y = x_I$ et passer en 3).

3) Placer sur la pile le premier sommet z venant après y dans l'ordre de numérotation et tel que l'ensemble $E \cup \{z\}$ vérifie $\mathfrak{F}$, E désignant l'ensemble des sommets contenus dans la pile. Itérer cette procédure tant qu'il reste des sommets non encore examinés en faisant $y = z$. Si l'on a pu ainsi faire progresser la pile passer en 4) sinon passer en 7).

4) Soit E l'ensemble des sommets contenus dans la pile lorsqu'on parvient à ce stade. Si $\alpha = \beta$ passer directement en 5). Dans le cas contraire tester s'il existe un sommet γ compris entre α et β ou égal à β tel que l'ensemble $E \cup \{\gamma\}$ vérifie $\mathfrak{F}$. Si la réponse est oui passer en 6), sinon passer en 5).

5) Tester s'il existe un sommet γ d'indice inférieur à I tel que $E \cup \{\gamma\}$ vérifie $\mathfrak{F}$. Si la réponse est oui passer en 6), sinon passer en 7).

6) Soit t le sommet qui figure sur la pile lorsqu'on parvient à ce stade. Faire $y = t$, retirer ce sommet de la pile et retourner en 2).

7) Si la pile contient au moins deux éléments passer en 8) sinon faire $I = I + 1$ et retourner en 2).

8) Soit t le sommet qui figure sur la pile lorsqu'on parvient à ce stade. Faire $y = t$ et $\beta = t$. Retirer alors ce sommet de la pile. Soit v le nouveau sommet qui figure sur la pile. Faire $\alpha = v$ et retourner en 3).

1.3. Recensement des cliques maximales d'un graphe

Dans le cas de la recherche des cliques d'un graphe symétrique la propriété $\mathfrak{T}(V)$ est vraie si et seulement si tous les sommets de V sont adjacents entre eux. Les cliques maximales coïncident alors avec les sous-ensembles $\mathfrak{T}$-majorants de X. L'algorithme 2 peut être utilisé sous la forme indiquée précédemment pour recenser toutes les cliques maximales. Pour exprimer $\mathfrak{T}$ on pourra représenter le graphe par sa matrice associée. L'algorithme global ainsi obtenu reste très simple et le programme correspondant très court. L'occupation mémoire des variables qui est égale à $N^2 + N + 10$ mots est du même ordre de grandeur que celle de l'algorithme qui apparaît dans [5] mais est très inférieure à l'occupation mémoire de l'algorithme de M. Roy [4]. Par contre ces deux derniers algorithmes comprennent à chaque itération une mémorisation de renseignements complémentaires qui évite le test $\mathfrak{T}$-majorant et qui les rend plus rapides à l'exécution que celui présenté ici. On notera que les cliques sont obtenues selon l'ordre lexicographique tout comme dans l'algorithme de M. Roy. Les deux algorithmes utilisent en effet la même procédure arborescente d'exploration qui dans les deux cas est décrite par l'intermédiaire d'une pile. Cependant si dans l'algorithme de M. Roy les éléments constitutifs de la pile sont des sous-ensembles de sommets, dans l'algorithme 2 ces éléments sont les sommets eux-mêmes. Cette situation explique que les deux algorithmes procèdent pour le détail des opérations d'une manière très différente.

1.4. Recensement des cliques θ-maximales d'un graphe

Étant donné que toute clique θ-maximale est majorante, le problème de la recherche des cliques θ-maximales se ramène d'abord à celui de la recherche des cliques majorantes. Il faut ensuite tester le nombre des sommets de chacune de ces dernières pour décider lesquelles sont θ-maximales. Les algorithmes 1 et 2 munis de ce test supplémentaire peuvent encore être utilisés pour résoudre ce problème. Cependant nous allons voir que la nature du test permet d'améliorer d'une façon non négligeable leurs performances.

Supposons en effet que la plus grande clique majorante obtenue en parvenant au stade 3 comprenne K sommets. Dès qu'il reste moins de $K - m$ sommets non encore examinés, où m désigne le nombre d'éléments contenus dans la pile, on peut passer directement au stade suivant car il ne sera jamais possible de trouver une clique supérieure ou égale à celle déjà rencontrée.

En ce qui concerne l'algorithme 2 la présence du même test rend inutile le test « $\mathfrak{T}$-majorant » du stade 5). En effet considérons la clique obtenue en parvenant au stade 5). Si cette clique est supérieure ou égale à la dernière clique θ-maximale trouvée, elle est forcément majorante.

Dans ces conditions, l'algorithme pour recenser toutes les cliques θ-maximales d'un graphe, est le suivant.

Algorithme 3 : Recensement de toutes les cliques θ-maximales d'un graphe.

1) Faire $I = 1$, $IS = 0$ et passer en 2).

2) Si I est supérieur à N, les sous-ensembles restant en mémoire sont les cliques θ-maximales du graphe; sinon créer une pile avec le sommet x_I, faire $y = x_I$ et passer en 3).

3) Placer sur la pile le premier sommet z venant après y dans l'ordre de numérotation, tel que z soit adjacent à tous les sommets contenus dans la pile. Itérer cette procédure en faisant $y = z$ tant que le nombre de sommets non encore examinés est supérieur ou égal à $IS - m$, m désignant le nombre de sommets actuellement contenus dans la pile. Passer en 4).

4) Soit m le nombre de sommets contenus dans la pile lorsqu'on parvient à ce stade. Si $m > IS$ passer directement en 6). Si $m = IS$ passer en 5). Si $m > IS$ supprimer tous les sous-ensembles de sommets précédemment mémorisés, faire $IS = m$ et passer en 5).

5) Mémoriser le sous-ensemble constitué par les sommets contenus dans la pile lorsqu'on parvient à ce stade. Passer en 6).

6) Si la pile contient au moins deux éléments passer en 7) sinon faire $I = I + 1$ et retourner en 2).

7) Soit t le sommet qui figure sur la pile lorsqu'on parvient à ce stade. Faire $y = t$, retirer ce sommet de la pile et retourner en 3).

A la fin de l'algorithme IS est égal à θ.

Cet algorithme qui ne présente plus l'inconvénient discuté précédemment est beaucoup plus rapide à l'exécution que l'algorithme 2. Cet avantage allié à une grande simplicité d'utilisation et à une occupation mémoire restreinte au mieux le rend particulièrement bien adapté à la résolution d'un tel problème.

1.5. Autres applications

Parmi les autres utilisations immédiates de ces algorithmes, citons :

— le recensement des sous-ensembles stables intérieurement en exprimant que $\mathfrak{I}(V)$ est vraie si et seulement si deux sommets quelconques de V ne sont pas adjacents. L'algorithme 2 permet alors le recensement des sous-ensembles stables intérieurement maximaux et l'algorithme 3 fournit tous les sous-ensembles stables intérieurement α-maximaux contenant α sommets, où α est le nombre de stabilité interne du graphe.

— Le recensement des noyaux qui constitue un cas particulier du problème précédent. En effet on obtient les noyaux d'un graphe en ne retenant parmi les sous-ensembles stables intérieurement maximaux que ceux qui sont aussi stables extérieurement.

— Le recensement des sous-ensembles maximaux qui, outre la propriété « être clique » ou être « stable intérieurement », doivent vérifier une propriété

additionnelle Π. Il suffit alors de remplacer dans les algorithmes précédents $\mathfrak{T}(V)$ par $\mathfrak{T}'(V) = \mathfrak{T}(V) \vee \Pi(V)$.

— Le recensement de certains sous-ensembles minimaux. Un sous-ensemble V vérifiant une certaine propriété $\mathfrak{T}$ est dit minimal si aucun sous-ensemble strict de V vérifie $\mathfrak{T}$. Par exemple en exprimant que $\mathfrak{T}(V)$ est vraie si et seulement si tout élément de V n'a pas comme suivant un élément de $X - V$, l'algorithme 2 donne tous les sous-ensembles E_i tels que les $X - E_i$ sont les sous-ensembles stables extérieurement minimaux.

Il apparaît ainsi qu'un même principe d'exploration peut être adapté au recensement de toute une classe de sous-ensembles maximaux ou minimaux au sens de différents critères. Nous avons souligné les caractéristiques des algorithmes correspondants relativement aux algorithmes les plus performants proposés par ailleurs pour les mêmes problèmes : plus grande simplicité, occupation mémoire en général inférieure, mais temps d'exécution plus long. Nous allons maintenant présenter un algorithme qui plus spécifiquement adapté au recensement des cliques maximales et θ-maximales, conduit à un temps d'exécution très intéressant lorsque le nombre de sommets par clique maximale est voisin du nombre N de sommets du graphe.

2. METHODE UTILISANT LA NOTION DE SOUS-GRAPHE PRESQUE COMPLET

2.1. Cliques dans un graphe presque complet

Définition 2 : Un graphe $G = (X, \Gamma)$ de N sommets est presque complet si, et seulement si chaque sommet a pour degré minimum $N - 2$, c'est-à-dire à-dire s'il est relié à au moins $N - 2$ autres sommets.

Théorème 2

Dans un graphe presque complet le nombre d'arcs manquants pour rendre le graphe complet est égal à la moitié du nombre de sommets de degré $N - 2$.

La démonstration est immédiate.

On notera (x_1, y_1); ...; (x_i, y_i); ...; (x_α, y_α) les α arcs manquants pour rendre le graphe complet, les x_i et y_i étant les sommets de degré $N - 2$ et $(e_1, e_2, ..., e_\beta)$ les β sommets de degré $N - 1$. On a bien sûr $\beta = N - 2\alpha$.

Théorème 3

Dans un graphe presque complet toute clique maximale est de la forme :

$$\{f_1, f_2, ..., f_i, ..., f_\alpha, e_1, e_2, ..., e_\beta\}$$

où f_i est soit le sommet x_i, soit le sommet y_i.

Démonstration ; Tout d'abord toute clique maximale comprend obligatoirement les sommets e_1, e_2, ..., e_β de degrés $N-1$ puisque ces sommets sont adjacents à tous les sommets du graphe. Par ailleurs toute clique C qui ne contient pas f_i n'est pas maximale puisque $C \cup \{f_i\}$ est une clique qui contient C. D'où le théorème.

Corollaire 1 : Dans un graphe presque complet G de N sommets une clique maximale possède $(N+\beta)/2$ sommets.

Corollaire 2 : Dans un graphe presque complet de N sommets le nombre de cliques maximales est 2^α.

Soit E l'ensemble des sommets de degré $N-1$ et F l'ensemble des arcs manquants pour rendre le graphe complet. D'après le théorème 3 les cliques d'un graphe presque complet peuvent être obtenues dès l'instant où l'on connaît les deux ensembles E et F. La détermination de ces derniers ne présente aucune difficulté.

2.2. Détermination d'un jeu de sous-graphes presque complets comprenant toutes les cliques maximales d'un graphe

D'après ce qui précède le problème de la détermination des cliques maximales d'un graphe G est immédiat dès l'instant où ce graphe est presque complet. Si G n'est pas presque complet on pourrait songer à recenser tous les sous-ensembles presque complets maximaux. Mais ce problème présente les mêmes difficultés que la détermination des cliques maximales.

Pour mieux exploiter la notion de sous-ensembles presque complet, les considérations suivantes peuvent être développées. Soit γ un sommet de G. Toute clique de G comprenant le sommet γ est aussi une clique du sous-graphe $H(\gamma)$ de G engendré par les sommets de X adjacents à γ. Les cliques maximales de G sont constituées d'une part des cliques maximales de $H(\gamma)$ et d'autre part des cliques maximales du sous-graphe obtenu en supprimant dans G le sommet γ. Si ces deux graphes sont presque complets le problème est encore immédiat; sinon on pourra répéter pour chacun d'eux la procédure de séparation appliquée au graphe initial et ainsi de suite. Il est clair qu'un tel processus permet de déterminer dans le graphe G un jeu de sous-graphes presque complets dont les cliques maximales coïncident avec les cliques maximales de G. Il sera commode de choisir pour sommet γ, le sommet de plus faible degré dans G puisque $H(\gamma)$ comprendra alors un nombre de sommets minimum.

Il serait possible de programmer ce processus sous une forme récursive, mais une telle méthode serait coûteuse en mots mémoires : à chaque nouvel appel il faudrait mémoriser la matrice associée du sous-graphe actuellement traité ou du moins certaines informations (au minimum la liste des sommets du sous-graphe en question) qui permettraient de la retrouver rapidement. Une forme non récursive nous paraît préférable car elle permet de mieux contrôler l'assignement mémoire.

Dans cette dernière optique, analysons plus en détail le déroulement du processus. Si le sous-graphe $H(\gamma)$ n'est pas presque complet il faut poursuivre la séparation en déterminant le sous-graphe $H_1(\gamma_1)$ de $H(\gamma)$ engendré par les sommets adjacents au sommet γ_1 de plus faible degré dans $H(\gamma)$. On itérera ainsi jusqu'à obtenir un sous-graphe $H_i(\gamma_i)$ presque complet. Il faut bien sûr mémoriser les différents sommets γ, γ_1, ..., γ_i. On pourra avantageusement les emmagasiner dans une pile MPA. Puis on supprimera provisoirement dans $H_{i-1}(\gamma_{i-1})$ le sommet γ_i et on reprendra le processus de séparation pour H_{i-1} en retirant au préalable le sommet γ_i de la pile MPA. La pile MPA va éventuellement progresser à nouveau et d'autres sommets seront supprimés provisoirement. Ces sommets seront emmagasinés dans une autre pile MPB. Un sommet γ_l supprimé provisoirement sera restitué lorsqu'un jeu de sous-graphes presque complets a été obtenu dans $H_{l-1}(\gamma_{l-1})$ c'est-à-dire lorsque la pile MPA est revenue au niveau où elle était lors de la suppression de γ_l. Une troisième pile MPC qui varie parallèlement à MPB retiendra les différents niveaux de la pile MPA.

L'algorithme complet est alors le suivant :

Algorithme 4 : Recherche d'un ensemble de sous-graphes presque complets contenant toutes les cliques maximales d'un graphe G.

Initialement les piles MPA, MPB, MPC sont vides.

1) Faire GAB = G. Passer en 2).

2) Si GAB est presque complet mémoriser ce graphe et passer en 4) sinon passer en 3).

3) Placer sur la pile MPA l'indice du sommet γ de plus faible degré dans GAB. Supprimer dans GAB tous les sommets qui ne sont pas adjacents à γ et retourner en 2).

4) Si la pile MPA est vide, alors un ensemble de sous-graphes presque complets contenant toutes les cliques maximales de G a été obtenu; sinon passer en 5).

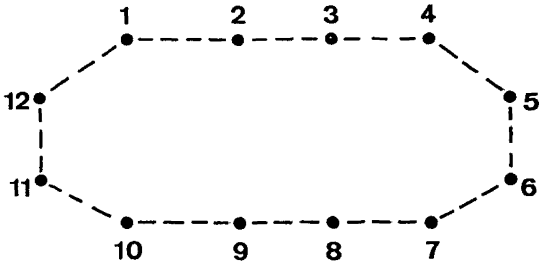
5) Si la pile MPA contient un seul sommet, supprimer ce sommet dans G, le retirer de la pile MPA et passer en 1) sinon passer en 6).

6) Tant que les piles MPB et MPC ne sont pas vides, retirer les éléments b et c figurant respectivement sur ces piles aussi longtemps que le niveau actuel de la pile MPA est inférieur ou égal à c. Passer alors en 7).

7) Soit t le sommet figurant sur la pile MPA. Placer ce sommet sur la pile MPB et le retirer de la pile MPA. Placer alors sur la pile MPC le nombre entier mesurant le niveau actuel de la pile MPA. Définir GAB comme le sous-graphe de G ne comprenant pas les sommets contenus dans la pile MPB et dont tous les sommets sont adjacents à ceux contenus dans la pile MPA. Retourner en 2).

2.3. Exemple d'application

Considérons le graphe de la figure ci-dessous à 12 sommets et auquel il manque les arêtes indiquées en pointillé pour qu'il soit complet.



On trouvera dans la table I les différents états des piles MPA et MPB lors du recensement des sous-graphes majorants presque complets contenant le sommet de numéro 12.

MPA	MPB	Liste des sommets du sous-graphe presque complet correspondant
12 10 8 6 4		12 10 8 6 4 2
12 10 8 6	4	12 10 8 6 3 2
12 10 8 4	6	12 10 8 5 4 2
12 10 8	6 4	12 10 8 5 3 2
12 10 6 4	8	12 10 7 6 4 2
12 10 6	8 4	12 10 7 6 3 2
12 10 4	8 6	12 10 7 5 4 2
12 10	8 6 4	12 10 7 5 3 2
12 8 6 4	10	12 9 8 6 4 2
12 8 6	10 4	12 9 8 6 3 2
12 8 4	10 6	12 9 8 5 4 2
12 8	10 6 4	12 9 8 5 3 2
12 6 4	10 8	12 9 7 6 4 2
12 6	10 8 4	12 9 7 6 3 2
12 4	10 8 6	12 9 7 5 4 2
12	10 8 6 4	12 9 7 5 3 2

2.4. Recensement des cliques maximales et θ-maximales d'un graphe

Pour recenser les cliques maximales il suffit dans l'algorithme précédent d'inclure au niveau du stade 2) le calcul des deux sous-ensembles E et F lorsque GAB est presque complet.

Les cliques θ-maximales sont obtenues de la même façon, mais en ne retenant que les sous-graphes presque complets qui conduisent à des cliques supérieures ou égales à celles précédemment trouvées. De plus si IS est le degré de la plus grande clique actuellement trouvée, tous les sommets de GAB de degré inférieur à IS ne participent pas à des cliques θ-maximales. On pourra donc les supprimer provisoirement avant de recommencer le stade 2, ce qui permet d'accélérer les opérations ultérieures.

Pour la programmation il est commode de représenter le graphe par sa matrice associée. Pour faciliter les opérations de traitement on a utilisé un vecteur NP contenant les numéros des sommets du sous-graphe GAB et un vecteur NS contenant leurs degrés respectifs. Les contenus de ces deux vecteurs doivent être préservés avant d'aborder la phase 2). Ils sont mis à jour par des sous-programmes spéciaux chaque fois qu'un sommet est supprimé et chaque fois qu'un nouveau sous-graphe GAB est analysé.

3. PERFORMANCES RESPECTIVES DES DEUX METHODES

Un problème non négligeable se pose lors de la mise en œuvre sur machine : c'est celui du stockage des cliques maximales ou θ-maximales. Ces cliques peuvent être en nombre très grand (voir corollaire 2) et il n'est généralement pas possible de les conserver en mémoire centrale. Par ailleurs le transfert sur mémoire auxiliaire d'une clique dès sa découverte monopoliserait beaucoup de temps pour les échanges. Il semble donc nécessaire d'adopter une attitude intermédiaire : stockage dans un bloc en mémoire centrale, puis transfert en mémoire auxiliaire lorsque le bloc est plein. L'importance du bloc dépendra en général de la taille de la mémoire centrale. Nous avons programmé les deux méthodes précédentes avec un bloc de 1 000 mots, taille très raisonnable pour la plupart des machines.

Pour effectuer des tests valables sur la durée d'exécution, il est nécessaire de choisir des graphes dans lesquels chaque sommet joue le même rôle, sinon la durée de la recherche dépendrait de l'ordre de numérotation des sommets. Pour de tels graphes les cliques maximales sont toutes θ-maximales.

On trouvera dans la table II quelques valeurs de la durée de la recherche des cliques θ-maximales pour différents graphes de 24 sommets obtenues avec l'algorithme 3 et l'algorithme 4 respectivement. Les essais ont été effectués sur un ordinateur CDC 6400. Le temps indiqué comprend également la durée des transferts entre la mémoire centrale et la mémoire auxiliaire.

Degré de chaque sommet	Nombre de cliques γ-maximales	Nombre de sommets par clique γ-maximale	Durée avec la 1ère méthode (en secondes)	Durée avec la 2ème méthode (en secondes)
23	1	24	0.037	0.066
22	4096	12	33.960	0.066
21	6561	8	17.249.	2.854
20	4096	6	6.117	5.909
18	1296	4	1.253	3.616
16	512	3	0.464	1.777
12	144	2	0.134	0.644

Lorsque le graphe est presque complet ou proche de la structure presque complète la deuxième méthode est très performante par rapport à la première. Par contre elle est légèrement plus désavantageuse lorsque les cliques comprennent peu de sommets. Du point de vue de l'occupation mémoire la deuxième méthode nécessite $N(N+8)+1\,000$ mots alors que la première n'en demande que $N(N+1)+1\,000$. La différence est faible et l'on aura presque toujours intérêt à employer la deuxième méthode.

REFERENCES

[1] BERGE C., *Graphes et Hypergraphes*, chapitre 13, Monographie de Mathématiques, Dunod, Paris, 1970.

[2] DEMOUCRON G., *Ensembles stables intérieurement d'un graphe*, Gestion, juillet 1968.

[3] HERZ J. C., *Quelques considérations sur les problèmes d'emploi du temps*, R.F.R.O., n° 38, pp. 85-91, 1966.

[4] ROY B., *Algèbre moderne et Théorie des Graphes*, chapitre 6, pages 14 et suivantes, Dunod, Paris, 1970.

[5] *Algorithmes des Graphes*, Contrat DGRST, pages 103 à 105, Institut de Programmation, Université de Paris VI, 1967.

L'auteur tient à remercier M. C. Picard, directeur de recherche au CNRS pour ses nombreuses suggestions.