N° d'ordre : 37/2008-M/MT

République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumédiène

Faculté des Mathématiques

Mémoire

Présenté pour l'obtention du diplôme de MAGISTER
En Mathématiques
Spécialité : *Recherche Opérationnelle (Génie Mathématique)*

Par : *Salima MERABET*

SUJET :

Sur quelques invariants de graphes et applications aux problèmes de Routage

Soutenu le 27/05/2008, devant le jury composé de :

M^r^ M. MOULAÏ, Maître de Conférence, USTHB	**Président**
M^r^ H. AIT HADDADENE, Professeur, USTHB	**Directeur de Mémoire**
M^lle^ I. BOUCHEMAKH, Professeur, USTHB	**Examinatrice**
M^r^ S. BOUROUBI, Maître de Conférence, USTHB	**Examinateur**

REMERCIEMENTS

A la lumière de ce modeste travail, je tiens à remercier, en premier lieu, ***ALLAH***
D'avoir me donner le courage, la volonté, la patience et la force pour accomplir ce travail jusqu'à son terme.

Mes remerciements vont :

A Monsieur ***AIT HADDADENE,*** *mon Directeur de mémoire, pour son aide à traiter ce Thème, pour ces orientations, ses précieuses remarques durant l'accomplissement de ce modeste travail et surtout pour son confiance et sa compréhension.;*

A Mademoiselle ***I. BOUCHEMAKH****, Professeur à USTHB, d'avoir accepter de présider le jury de ce mémoire.*

A Monsieur ***S.BOUROUBI,*** *chargé de cours à l'USTHB, pour avoir voulu examiner ce travail.*

A Monsieur ***M. MOULAI,*** *chargé de cours à l'USTHB, pour avoir voulu examiner ce travail.*

Mes chaleureux remerciements vont également à tous les enseignants du département Recherche Opérationnelle qui ont contribué à ma formation.

Je n'oublierais pas de remercier :

En fin mes chaleureux remerciement s'adressent à ma famille et tous mes amis (es) et toute personne qui à contribué de prés ou de loin dans la réalisation de ce modeste travail.

Introduction générale

Là où il y a une volonté, il y a un chemin.
Gaston Rebuffat

La théorie des graphes est un domaine très vaste issu de la recherche opérationnelle, elle est en évolution constante tant du point de vue des recherches fondamentales que celui des applications. Concernant la recherche fondamentale, il faut citer les travaux de P. Seymour et de ses collaborateurs qui ont démontré la Conjecture des Graphes Parfaits formulée par Claude Berge, amélioré la preuve du théorème dit des quatre couleurs et surtout, développé la théorie des mineurs de graphes.

D'autre part, les applications sont très nombreuses. Elles justifient une recherche importante en algorithmique. L'importance des réseaux de transport et de communication, qui sont d'ailleurs de plus en plus des réseaux évolutifs, par exemple faire fonctionner les connexions des utilisateurs à des serveurs de téléphonie mobile, ou à des réseaux pair-à-pair[1] et d'explique le foisonnement des problématiques.

Voici un des problèmes fondamentaux en théorie des graphes « Trouver une route d'une extrémité à une autre à travers une succession de relais interconnectés ». Ce genre de problème porte le nom de routage même les algorithmes et protocoles permettant de les résoudre sont également dits de routage.

La plupart du temps, on résout les problèmes de manière "brutale". Il suffit parfois d'appliquer une définition, où lorsque nous cherchons une solution particulière, nous pouvons énumérer toutes les solutions possibles. Une telle démarche est rarement satisfaisante du point de vue temps de calculs.

Pour résoudre un problème, une démarche usuelle consiste à s'intéresser à des cas particuliers plus simples, ensuite à ramener le problème général à ces cas particuliers. L'application de cette démarche dans les graphes, nous conduit à transformer des problèmes complexes en des problèmes sur des arbres. C'est ainsi qu'un certain nombre de décompositions des graphes en des structures arborescentes, ont vu le jour. Dans notre mémoire nous essayons d'étudier deux types de décompositions ; la décomposition arborescente et la décomposition en branches.

[1] Par exemple le réseau Internet.

Certains graphes possèdent naturellement une structure arborescente, c'est le cas des graphes triangulés[2]. Cette structure apparaît au cours de l'étude de la conjecture des graphes parfaits dans les travaux de Dirac en [Dir61] sous le nom de « Rigid Circuit Graphs ». Cependant, il est toujours possible de plonger un graphe *G* dans un graphe triangulé *H*. De cette façon, la structure arborescente de *H* se transpose à *G*. Cependant, lors d'un tel plongement, nous perdons l'information sur la structure de *G* ; il est donc nécessaire de trouver un plongement ayant de "bonnes" propriétés. C'est dans ce cadre que, dans les années 70, sont apparus les *k*-arbres et les *k*-arbres partiels dans les travaux de Rose [Ros74] et de Dirac.

En 1960, Kruskal [Kru60] a démontré que dans une famille infinie d'arbres, au moins l'un des arbres est mineur[3] d'un autre. Dans les années 80, Robertson et Seymour se sont intéressés à la généralisation de ce théorème à des familles infinies de graphes, ils l'ont nommé en suite ; « Conjecture de Wagner » [Wag37]. Pour démontrer ce théorème, ils ont essayé de se ramener au résultat de Kruskal et ont réintroduit [RS84, RS86] à cet effet, les décompositions arborescentes qui avaient déjà été définies par Halin [Hal76] sous un autre nom. Pour pouvoir appliquer le résultat de Kruskal, ayant besoin de mesurer à quel point un graphe ressemble à un arbre, ils ont utilisé pour cela, un paramètre associé aux décompositions arborescentes appelées, la largeur arborescente, notée *tw(G)*. Lors d'une série d'articles, ils ont obtenu de nombreux résultats importants, entre autres une généralisation du théorème de Kuratowski : pour toute classe de graphes ζ clos par minoration, il existe une famille finie $\Im_\zeta$ de graphes telle que ζ est exactement l'ensemble des graphes n'admettant aucun graphe de $\Im_\zeta$ comme mineur. La famille $\Im_\zeta$ est une famille d'obstructions pour la famille ζ.

Contrairement aux travaux de Halin passant inaperçus, ceux de Robertson et Seymour attirent l'attention. Arnborg, Corneil et Proskurowski [ACP87] ont montré que le problème de décision (satisfaisabilité) ; « le graphe *G* est de largeur arborescente *t* » est un problème NP-complet. Cependant, ils ont montré que si le paramètre *t* est fixé, ce problème devient polynomial et ils ont développé un algorithme de complexité $O(n^{t+2})$ pour le résoudre et ils ont construit, pour le cas échéant, une décomposition de largeur arborescente au plus *t*. Robertson et Seymour [RS95] ont amélioré le résultat de décision en utilisant le théorème de Kuratowski généralisé. Comme l'ensemble des graphes de largeur arborescente au plus *t* est clos par minoration, il s'exprime par une famille d'obstructions $\Im_t$. En construisant une décomposition approchée, ils ont arrivé à tester efficacement si le

[2] Un graphe est triangulé signifie que tout cycle de longueur supérieure à 3 possède une corde, les graphes triangulés contiennent les graphes complets.

[3] Un graphe est un mineur d'un graphe *G* s'il est obtenu en retirant des arêtes ou des sommets ou en contractant des arêtes de *G*.

graphe *G*, modélisant le problème, a pour mineur un des éléments de la famille $\Im_t$. La complexité de cet algorithme est $O(n^2)$ quoique celui-ci est non constructif. Bodlaender [Bod96] a amélioré ce résultat et il a donné un algorithme linéaire $O(n)$ pour ce problème de décision, ainsi que pour le problème de construction d'une décomposition arborescente correspondante, mais pour la classe des graphes de largeur arborescente au plus *t*, la constante cachée le *O* est d'ordre 2^t. On pourrait en conclure que, puisque ce qui nous intéresse est de savoir si un graphe possède une largeur arborescente « petit », l'algorithme de Boadlaender est suffisant mais le facteur que cache la notation *O* pour cet algorithme est exponentiel en *t*, la complexité donné par l'algorithme de Boadlaender est en fait de $O\left(2^{(2t)^t} n\right)$.

Parallèlement a cela, Courcelle [Cou89] puis d'autres comme Arnborg et col. [ALS91] ou Borie et col. [BPT91] ont montré que certains problèmes difficiles pouvant se résoudre efficacement pour les graphes de largeur arborescente bornée ; tout problème pouvant s'exprimer dans une logique monadique du second ordre[4] étendue se résout en temps linéaire; ces algorithmes cachent cependant eux aussi une constante exponentielle en la largeur arborescente dont il est le cas de problème du stable maximum et celui du cycle hamiltonien. Pour être plus précis, pour résoudre un problème comme le stable maximum, il faut un temps de $O\left(2^t n\right)$, lorsque le graphe en entrée a *n* sommets et la largeur arborescente au plus *t*. Ceci donne une idée de la borne supérieure que l'on peut accepter pour *t* de sorte à rester à des complexités « raisonnables ».

Ces résultats ont stimulé le travail sur le calcul de la largeur arborescente et sur les triangulations des graphes. En effet, comme nous l'avons déjà mentionné, "trouver une bonne décomposition arborescente revient à trouver une bonne triangulation". Le calcul de la largeur arborescente a été montré polynomial pour les graphes d'intervalles circulaires [SSP94], les graphes de permutation [BKK95] et les graphes de cordes [Klo96]. Plus généralement, en étudiant les triangulations minimales des graphes, Bouchitté et Todinca [BT01a, BT01b] ont montré que ce calcul se fait en temps polynomial pour toutes classes de graphes ayant un nombre polynomial de séparateurs minimaux.

Dans la suite de leur série d'articles sur les mineurs de graphes, Robertson et Seymour [RS91] ont présenté un nouveau type de décompositions de graphes proche des décompositions arborescentes c'est la décompositions en branches. Idem pour les décompositions arborescentes associant pour les décompositions en

[4] Par exemple, le problème de 3-coloration d'un graphe s'exprime par la formule :

$$\exists A, B, C \subseteq \left(\forall u \in V \left(u \in A \right) \vee \left(u \in B \right) \vee \left(u \in C \right) \right) \wedge$$
$$\left(\forall uv \in E \neg \left(u \in A \wedge v \in A \right) \vee \neg \left(u \in B \wedge v \in B \right) \vee \neg \left(u \in C \wedge v \in C \right) \right)$$

branches un paramètre ceci est appelé « La largeur de branches », notée *bw(G)*. Ils les ont introduit comme obstruction aux décompositions arborescentes et ont montré que tous les graphes l'inégalité $bw(G) \leq tw(G) + 1 \leq \lfloor 3bw(G)/2 \rfloor$ est vérifiée.

Les décompositions arborescentes et les décompositions en branches ont présenté de fortes similitudes. L'inégalité de Robertson et Seymour permet d'étendre les résultats du type de ceux de Courcelle aux graphes de largeur de branches au plus *t* ; des propriétés analogues à celles de la largeur arborescente sont ainsi établies pour la largeur de branches. Bodlaender et Thilikos [BT97] ont obtenu un résultat analogue à celui de Bodlaender [Bod96] en développant un algorithme linéaire pour le problème de satisfaisabilité suivant : « peut on décider si la largeur de branches est égale à *t* et construire le cas échéant une décomposition en branches correspondante? ».

Kloks et col. [KKM99] ont montré que le problème de satisfaisabilité associé à la largeur de branches est NP-complet. Tous ces résultats justifient l'idée que les décompositions arborescentes et les décompositions en branches possèdent les mêmes propriétés.

D'une part, Kloks et col. [KKM99] ont montré que le problème de satisfaisabilité associé à la largeur de branches est NP-complet pour les graphes triangulés alors que le problème correspondant pour la largeur arborescente est linéaire. D'autre part, Seymour et Thomas [ST94] ont développé un algorithme polynomial de calcul de la largeur de branches pour les graphes planaires alors que le problème correspondant pour la largeur arborescente reste toujours ouvert.

Nous avons structuré notre mémoire en cinq chapitres où :

Dans le premier chapitre, nous précisons la terminologie et les notations usuelles sur les graphes. Nous introduisons ensuite des notions spécifiques à notre champ d'étude. Nous commençons par celle de *séparateur minimal* avant de nous intéresser à celle, plus générale, de *séparation* et d'étendre à ces objets la relation usuelle de parallélisme sur les séparateurs minimaux. Nous présentons ensuite une classe de graphes étroitement liés aux séparateurs minimaux : *les graphes triangulés*. Approfondissement dans ces liens nous a amené à présenter certains théorèmes qui soulignent la structure arborescente de ces graphes.

Dans le second chapitre, on poursuit l'étude des séparateurs minimaux en vue de leur énumération. Nous donnons un résultat de structure sur l'ensemble des *a,b*-séparateurs minimaux dû aux travaux de Robertson et Seymour.

Par ailleurs, au troisième chapitre, nous choisissons de présenter les *décompositions arborescentes* et les *décompositions en branches* comme provenant de certains schémas de décompositions de graphes et quelques liens entre les deux décompositions.

À l'aide de ces outils, nous pouvons associer des décompositions en branches à des décompositions arborescentes et réciproquement, ainsi donner une preuve d'un théorème min/max de Robertson et Seymour reliant la largeur de branches et la largeur arborescente d'un graphe.

En étudiant certaines triangulations, dans le chapitre 4, plus précisément, ce chapitre présente un algorithme de calcul de triangulations dites *serrées* lequel est utilisé pour retrouver le résultat original de Bouchitté et Todinca d'une part et pour obtenir un algorithme de calcul de largeurs de branches d'autre part.

Nous présentons dans le chapitre 5 quelques résultats obtenus pour le calcul des invariants cités précédemment et nous proposons également un algorithme polynomial pour le calcul de la largeur arborescente de la classe des graphes *k*-cordaux et les graphes planaires de tel sort, pour la majorité de ces graphes nous avons pu déterminer soit la valeur exacte soit une borne inférieur. Ainsi déterminer la longueur arborescente des graphes k-cordaux. Pour la décomposition arborescente nous montrons la relation entre largeur arborescente et la frontière de la décomposition en banches. Ce chapitre est dédié aussi à l'étude de la décomposition de certaines classes de graphes.

En fin, nous clôturons notre travail par une conclusion générale qui nous permet de revenir sur les résultats de ce mémoire tout en proposant des perspectives de cet axe de recherche.

1. *Généralités et définitions*

Dans un réseau interconnecté les liens de communication sont bidirectionnels, c'est pourquoi de manière générale le réseau est modélisé par un graphe non orienté. Les sommets du graphe représentent les machines et les arêtes représentent les liens de communication.

Tout au long de la thèse nous considérerons des graphes simples $G=(V,E)$, finis et non orientés. De plus, nos graphes seront connexes, car tous les paramètres qui nous intéressent (longueur arborescente, largeur arborescente, largeur de branche, complétion minimale) se calculent pour les graphes non connexes en traitant séparément chacune des composantes connexes.

1.1 Préliminaires

Avant toute chose, il est bon de poser clairement les définitions que nous utilisons par la suite. Les notions que nous introduisons ici sont pour la plupart très classiques pour les graphes.

Définition 1.1 (Graphe, Sous-graphe)
Un graphe sera noté $G=(V,E)$, où V est l'ensemble de sommets et E est l'ensemble d'arêtes de G. Le nombre de sommets, respectivement d'arêtes sera noté n, respectivement m. Une arête est un lien entre deux sommets du graphe. Une arête (u,v) est une boucle si $u = v$. Tout au long de ce mémoire nous ne considérerons que des graphes simples, c'est-à-dire sans boucle ni arête multiple[1], voir par exemple la Figure 1.1 :

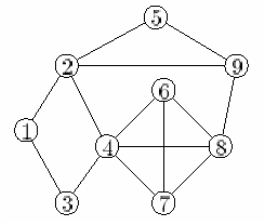


Figure 1.1 - Un graphe simple avec 9 sommets et 14 arêtes.

Un graphe $G'=(V',E')$ sous-graphe d'un graphe $G=(V,E)$ si $V' \subseteq V$ et $E' \subseteq E$, voir par exemple Figure 1.2.

[1] Une arête est dite multiple s'il existe au moins une autre arête avec les mêmes sommets, sa multiplicité étant le nombre total d'arêtes ayant ces sommets.

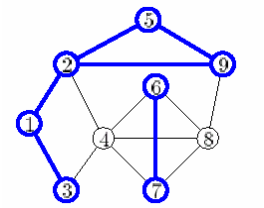


Figure 1.2 - Un sous graphes induits par les sommets 1, 2, 3, 5, 6, 7 et 9

Soit le graphe $G=(V,E)$, le sous-graphe $G'=(V',E')$ est un sous-graphe induit de G si, pour toute paire de sommets (x,y) de V', $uv \in E \Leftrightarrow uv \in E'$. Autrement dit, si deux sommets adjacents dans G sont présents dans G', alors ils sont aussi adjacents dans G'. G' est appelé le sous-graphe de G induit par V', il sera noté $G[V']$.

Un sous-graphe G' d'un graphe G est un sous-graphe couvrant de G, si G' contient tous les sommets de G : $V'=V$.

Par exemple, la Figure 1.2 montre un sous-graphe du graphe de la Figure 1.1 qui est induit par les sommets 1, 2, 3, 5, 6, 7 et 9. Il n'est pas couvrant car il manque les sommets 4 et 8. Il n'est pas non plus connexe car par exemple il ne possède pas de chaîne entre 6 et 3.

Définition 1.2 (Classes de graphes)

Nous n'épuiserons pas dans cette section la description des classes de graphes que nous aurons à traiter par la suite. Mais comme nous rencontrerons dans cet ouvrage plusieurs classes de graphes d'intersection, nous en donnons ici le principe.

On considère une famille $\mathcal{F}$ d'ensembles non vides. On lui associe le graphe d'intersection $G_{\mathcal{F}}$ de $\mathcal{F}$ obtenu comme suit : chaque ensemble de $\mathcal{F}$ est représenté par exactement un sommet de $G_{\mathcal{F}}$ et deux sommets de $G_{\mathcal{F}}$ sont adjacents si et seulement si les ensembles correspondants de $\mathcal{F}$ s'intersectent. On dira que la famille $\mathcal{F}$ est un modèle d'intersection de $G_{\mathcal{F}}$.

Quand la famille $\mathcal{F}$ est formée d'ensembles particuliers, on obtient des classes de graphes intéressantes.

Les graphes d'intervalles sont les graphes d'intersection des intervalles d'un ordre total (de la droite réelle, par exemple).

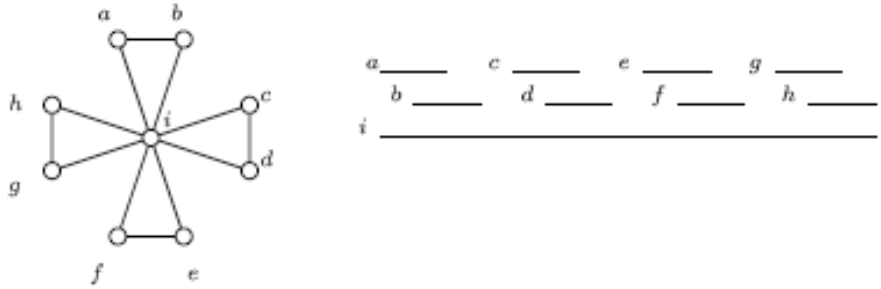


Figure 1.3- Graphe d'Intervalles

Les graphes de cordes sont obtenus comme intersections des cordes d'un cercle. Lorsque la famille $\mathcal{F}$ est formée d'arcs de cercle, on obtient les graphes d'intervalles circulaires. Ces deux dernières classes sont incluses dans la classe des graphes obtenus en intersectant des régions d'un cercle, où par région on entend la partie entre deux cordes qui ne se croisent pas.

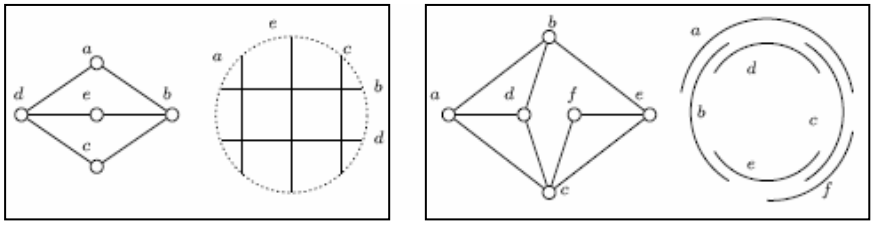


Figure 1.4- Graphes de cordes et d'intervalles circulaires

Définitions 1.3 (Adjacence, Voisinage, degré, Incidence)

Deux sommets *u* et *v* sont adjacents s'il existe une arête entre *u* et *v*. On dit aussi que *v* est dans le voisinage du sommet *u*. On notera *N(u)* l'ensemble des voisins de *u*. Le degré d'un sommet *u*, noté *deg(u)*, désigne le nombre de ses voisins.

Sur la Figure 1.1, le sommet 6 est de degré trois car *N(6) = {4, 7, 8}*.

Un sommet *u* est incident à une arête *e* s'il est situé à une des deux extrémités de cette arête. Inversement, une arête désigner le trait qui relie deux sommets du graphe.

Définition 1.4 (Voisinage clos d'un sommet)

Le voisinage clos d'un sommet *v* est défini comme $N[v] = N(u) \cup \{u\}$. Deux sommets v_1 et v_2 sont dits vrais jumeaux s'ils ont le même voisinage et sont adjacents, c'est-à-dire qu'ils ont le même voisinage clos $N[v_1] = N[v_2]$. Ainsi deux sommets v_1 et v_2 sont faux jumeaux s'ils ont le même voisinage mais ne sont pas adjacents, soit $N(v_1) = N(v_2)$. On définit aussi le non-voisinage de *v* comme étant $\overline{N}(v) = V \setminus N[v]$.

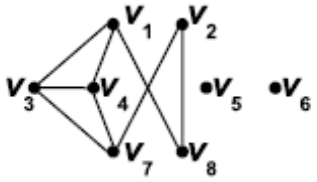


Figure 1.5- Le voisinage de v_1 est $N(v_1) = \{v_3,\ v_4,\ v_8\}$, son degré est donc 3.

Les sommets v_5 et v_6 sont isolés et les deux sont des faux jumeaux. Les sommets v_3 et v_4 sont des vrais jumeaux.

Définitions 1.5 (Chaîne, Cycle, Corde, Longueur)

Une chaîne dans un graphe *G* = (*V*,*E*), permettant d'aller d'un sommet u_1 à un sommet u_k, est une suite $(u_1, u_2, \ldots, u_k)$ de sommets distincts de *G* telle que pour

tout $i \in (1, \ldots, k-1)$, (u_i, u_{i+1}) est une arête. La longueur d'une chaîne est le nombre d'arêtes de la chaîne, ici *k-1*.

Un cycle est une chaîne dont le sommet de départ et le sommet d'arrivée sont identiques. Par convention on ne notera pas deux fois le sommet extrémité puisqu'un cycle n'a évidemment pas d'extrémité. Une corde dans un cycle est une arête reliant deux sommets non adjacents de ce cycle. Dans le graphe de la Figure 1.1, les sommets (1, 2, 5, 9, 8, 7, 4, 3) forment un cycle, les arêtes {2,9}, {2,4} et {4,8} en sont des cordes.

Définition 1.6 (Routage)

Dans un réseau de communication point à point ou dans les ordinateurs parallèles, une fonction de routage est utilisée pour envoyer des messages entre processeurs. Quand les réseaux grossissent, il devient important de réduire la quantité de mémoire contenue dans chaque noeud pour l'opération de routage. En même temps, il est essentiel d'acheminer les messages avec les chemins les plus courts possibles.

Définitions 1.7 (Distance, Diamètre)

La distance entre deux sommets est la longueur de la plus courte chaîne entre eux.

Le diamètre d'un graphe est la plus longue distance entre deux sommets de ce graphe.

Pour tout sous-graphe $G' = (V', E')$ de $G = (V, E)$, on définit le diamètre de G' dans *G* par $diam_G(G') = \max_{u,v \in V'} dist_G(u, v)$. Bien que la distance entre deux sommets d'un graphe non connexe ne soit pas définie, le diamètre d'un sous-graphe non connexe est défini.

Exemple Le diamètre du sous-graphe de la Figure 1.2 est égal à trois. Par extension, pour tout sous-ensemble V' de sommets de *V*, on définit le diamètre de V' dans *G* par $diam_G(V') = Max_{C_i \in G[V']}(diam_G(C_i))$ (avec C_i est une composante connexe de $G[V']$ $i := \overline{1, p}$ et *p* étant le nombre des composantes connexes de V').

Dans un graphe *G*, une clique est un sous-graphe complet, donc de diamètre 1. Une clique maximale est une clique de *G* qui est maximale par inclusion : si on ajoute n'importe quel autre sommet de *G*, on ne peut pas obtenir une clique. On note par $\omega(G)$ le nombre de sommet de la plus grosse clique de *G*, qui est aussi appelée la clique maximum de *G*. Dans le graphe de la Figure 1.1, le sous-graphe induit par les sommets 2, 5 et 9 est une clique maximale qui n'est pas maximum ; si on ajoute un sommet pour cette clique on n'obtient pas de clique, ce qui justifie que cette clique soit maximale, mais elle n'est pas maximum car le sous-graphe induit par les sommets 4, 6, 7 et 8 est une clique de taille quatre.

Définitions 1.8 (Connexité, composante connexe)

Un graphe connexe est un graphe dans lequel chaque paire de sommet est reliée par une chaîne. Un graphe qui n'est pas connexe est dit non connexe, et se décompose en composantes connexes.

Dans un graphe, une composante connexe est un sous-graphe induit maximal connexe, maximal signifie qu'il n'y a pas de sous-graphe induit connexe plus grand contenant les sommets de la composante.

Exemple Le graphe de la figure 1.4 a trois composantes connexes : $\{v_1, v_2, v_3, v_4, v_7, v_8\}$, $\{v_5\}$ et $\{v_6\}$.

Définitions 1.9 (Arbre, Arbre enraciné, Arbre couvrant)

Un arbre est un graphe connexe ne contenant aucun cycle.

Un arbre enraciné (appelé également arborescence) est un arbre où un nœud est distingué. Ce nœud est appelé racine de l'arbre. Les arbres enracinés seront, lorsque c'est possible, représentés « tête en haut » : la racine est le sommet situé le plus haut (voir un exemple avec la Figure 1.6).

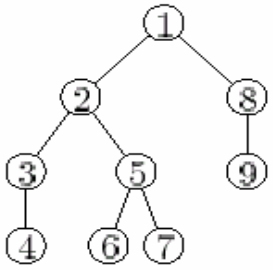


Figure 1.6 - Un arbre enraciné en 1

Un tel arbre induit une relation d'ordre partiel sur ses nœuds.

Un arbre couvrant est un sous-graphe maximum d'un graphe qui est aussi un arbre. On parle aussi d'arbre de recouvrement.

Un sous-graphe couvrant de G et un graphe G' avec :

- $V(G') = V(G)$ et
- $E(G') \subseteq E(G)$

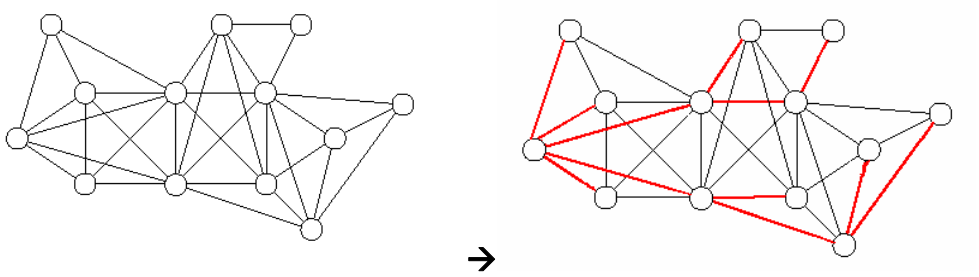


Figure 1.7 - Un arbre couvrant de plus court chemins.

Définitions 1.10 (Nœud, père, fils)

Nous appelons nœuds les sommets d'un arbre.

Le père (fils) est le plus petit antécédent (grands successeurs) d'un nœud *x*.

Exemple Pour la figure 1.6 le sommet 3 est le père du sommet 4 et le fils du sommet 2.

Définitions 1.11 (Graphe complet, Clique maximal)

Un graphe est dit complet si seulement si tous les sommets sont deux à deux adjacents. Si le nombre de sommet est égal à *n* sommets alors, on a $\frac{n(n-1)}{2}$ arêtes (*n*-clique[2]). Un graphe à *n* sommets est noté K_n (le *K* est en l'honneur de **K**uratowski, un pionnier de la théorie des graphes). Un graphe est donc complet si et seulement si son diamètre est 1. Voici le graphe de la Figure 1.8.

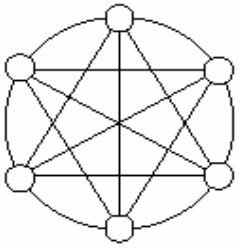


Figure 1.8 - Un graphe complet à 6 sommets noté K_6

Une clique maximale d'un graphe *G* est un sous-graphe induit $G[V']$, avec $V' \subseteq V$, tel que $G[V']$ est une clique et $\forall\ V''$, $V'' \subset V'$, $G[V'']$ n'est pas une clique. Par abus de notation on écrira aussi parfois que l'ensemble de sommets lui-même, V', est une clique maximale de *G* (est une clique de *G* maximale pour l'inclusion). Attention à ne pas confondre la clique maximale avec la clique maximum qui désigne usuellement une clique de taille maximale d'un graphe.

1.2 Séparateurs minimaux et frontières

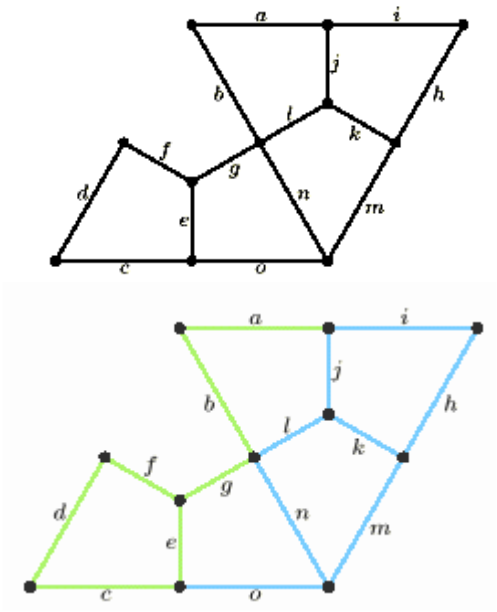
Comme nous l'avons déjà mentionné, l'objet de cette thèse est l'étude de certaines décompositions des graphes. Pour décomposer les graphes, nous allons utiliser les notions de séparateur minimal et de séparation que nous introduisons ci-dessous.

Définition 1.12 (Séparation)

Une **séparation** est une partition de *E* en deux ensembles $\{E_1, E_2\}$.

Soit le graphe $G = (V, E)$.

[2] Chaque sous-graphe de ce graphe est une clique



1.2.1 Séparateurs minimaux

Définition 1.13 (*a,b*-séparateur)
Soit $G=(V,E)$ un graphe quelconque. Un sous-ensemble $S \subseteq V$ est un séparateur de G si et seulement si le graphe $G[V \setminus S]$ n'est pas connexe. S est appelé un *a,b*-séparateur si et seulement si les deux sommets a et b sont dans deux composantes connexes différentes de $G[V \setminus S]$. S est un *a,b*-séparateur minimal si et seulement si S est un *a,b*-séparateur et aucun sous-ensemble de S n'est aussi un *a,b*-séparateur. S est un séparateur minimal si et seulement si, il existe a et b tels que S soit un *a,b*-séparateur minimal.

La Figure 1.9 présente par exemple un graphe G avec S un de ses séparateurs minimaux.

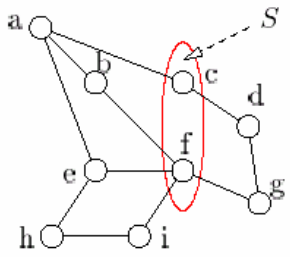


Figure 1.9 - Exemple de séparateur minimal.

Nous notons Δ_G l'ensemble des séparateurs minimaux de G. Si Ω est inclus dans $V(G)$, $\Delta_G(\Omega)$ désigne l'ensemble des séparateurs minimaux de G inclus dans Ω.

Définition 1.14 (Composante connexe pleine)
Etant donné S un séparateur de G, une composante connexe C de $G[V \setminus S]$ est appelée une composante connexe pleine si tout sommet de S a au moins un voisin dans C. ($N(C) \subset S$.)

Autrement dit : *C* est une composante connexe pleine si son voisinage contient *S*. Le voisinage d'un sous-ensemble de sommets V' étant défini par :

$$N(V') = \left(\bigcup_{u \in V'} N(u) \right) \setminus V' .$$

Dans le graphe de la Figure 1.9, le séparateur minimal *S* possède bien deux composantes connexes pleines : *{d, g}* et *{a, b, e, h, i}*.

Il faut remarquer qu'un *a,b*-séparateur *S* d'un graphe *G* peut être vu comme un ensemble de sommet par lequel toute chaîne entre *a* et *b* doit passer. De plus *S* est minimal si et seulement si pour tout sommet *u* de *S*, il existe une chaîne entre *a* et *b* qui intersecte *S* uniquement en *u*.

Nous notons $\zeta_G(S)$ l'ensemble des composantes connexes de $G \setminus S$. La composante connexe de $G \setminus S$ contenant un sommet *a* n'appartenant pas à *S* est notée $\zeta_G^a(S)$. Nous notons $\zeta_G^*(S)$ l'ensemble des composantes pleines par rapport à *S*.

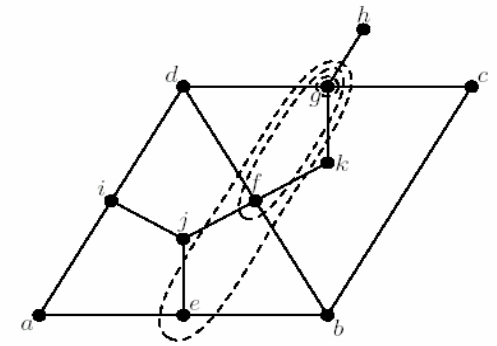


Figure 1.10 - Séparateurs minimaux

Dans la Figure 1.10, les composantes connexes de $G \setminus S$ sont *{a, d, i, j}*, *{b, c}*, *{h}* et *{k}*. Les composantes pleines sont *{a, d, i, j}* et *{b, c}*.

L'ensemble *{e, f, g}* est un *a,b*-séparateur minimal, *{f, g}* est un *j,k*-séparateur minimal et *{g}* est un *h,i*-séparateur minimal.

Théorème 1.15 ([GKKPP00]) Tout séparateur minimal d'un graphe de cordalité bornée par *k* est de diamètre au plus *k/2*.

Définitions 1.16 (Croisement, Parallélisme)
Soient *S* et *T* deux séparateurs de *G*. Nous dirons que *S* est parallèle à *T* si *S* est inclus dans un ensemble de la forme $T \cup C$ où *C* est une composante connexe de $G \setminus T$. Dans le cas contraire, c'est-à-dire si *S* rencontre au moins deux composantes connexes de $G \setminus T$, nous dirons que *S* croise *T*.

Lemme 1.17 ([Par96]) Les relations de parallélisme et de croisement sont symétriques entre séparateurs minimaux.

□ La démonstration est donnée dans [Maz04]

■

L'exemple de la Figure 1.14, montre que les séparateurs minimaux S_1 et S_2 sont parallèles, de même pour S_1 et S_3. Par contre S_2 et S_3 se croisent.

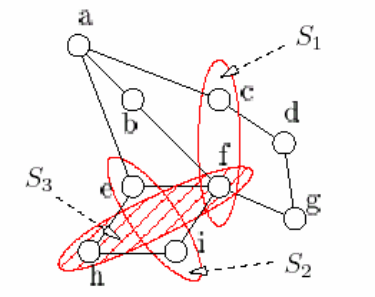


Figure 1.11 - Croisement et parallélisme entre séparateurs minimaux.

Propriété 1.18 Soient *S* et *T* deux séparateurs minimaux d'un graphe *G*. *S* et *T* sont parallèles si et seulement si il existe une composante connexe C_T de $G \setminus T$ telle que *S* soit inclus dans $C_T \bigcup N(C_T)$.

□ Soient *S* et *T* deux séparateurs minimaux d'un graphe *G*. Si *S* est inclus dans $C_T \bigcup N(C_T)$ où *C* est une composante connexe de $G \setminus T$, alors *S* croise au plus une composante connexe de $G \setminus T$: *S* est parallèle à *T*.

Réciproquement, supposons que *S* soit parallèle à *T*. Soient C_S et D_S deux composantes pleines dans $\zeta_G^*(S)$. Comme, d'après le lemme 1.17, la relation de parallélisme est symétrique, *T* est parallèle à *S* et ne peut pas rencontrer à la fois C_S et D_S, nous pouvons donc supposer que C_S et *T* sont disjoints. Puisque l'ensemble C_S induit un sous-graphe connexe de *G* et que C_S et *T* sont disjoints, C_S est inclus dans une composante connexe C_T de $G \setminus T$. Comme *S* est le voisinage de C_S, *S* est inclus dans $C_T \bigcup N(C_T)$.

■

Théorème 1.19 ([Dir61])
Un graphe est triangulé si et seulement si tous ses séparateurs minimaux induisent des cliques.

□ Soient *G* un graphe triangulé et *S* un séparateur minimal de *G*.

Montrons que *S* induit une clique de *G*. Pour cela, soient *x* et *y* deux sommets quelconques de *S*. La propriété 1.11 assure qu'il existe deux composantes connexes pleines *C* et *D* de $G \setminus D$ par rapport à *S*. Les sommets *x* et *y* possèdent donc des voisins v_x et v_y dans *C*. De plus, comme les sommets v_x et v_y appartiennent à une même composante connexe, il existe une chaîne $(x, c_1, ..., c_p, y)$ de *x* à *y* dont tous les sommets ci appartiennent à *C*. Si nous choisissons une telle chaîne de longueur minimale, la seule corde qu'elle puisse admettre relie *x* et *y*.

Choisissons de manière analogue une chaîne $(y,\ d_1,...,\ d_q,\ x)$ dont tous les sommets d_i soient dans *D*. Le cycle $\mu = (x,\ c_1,...,\ c_p,\ y,\ d_1,...,d_q,\ x)$ est de longueur au moins quatre. Le graphe *G* étant triangulé, μ possède une corde. Les sommets c_i et d_j n'appartenant pas à la même composante connexe de *G\S*, les sommets c_i ne sont pas adjacents aux sommets d_j. La seule corde que puisse admettre le cycle μ relie les sommets *x* et *y*. Les sommets *x* et *y* étant quelconques, *S* induit une clique de *G*.

Réciproquement, si *G* n'est pas triangulé, il possède un cycle μ égal à $(x_1,\ ...,\ x_p,\ x_1)$ de longueur au moins quatre sans corde. Nous allons construire un x_1, x_3 – séparateur minimal qui n'induit pas de clique de *G*. L'ensemble $N(x_1)$ est un x_1, x_3 – séparateur. Nous pouvons donc en extraire un x_1, x_3 – séparateur minimal *S*. Ce séparateur minimal contient x_2. De plus, comme aucun des sommets de x_4 à x_{p-1} n'est voisin de x_1, *S* contient aussi x_p. Les sommets x_2 et x_p ne sont pas voisins donc *S* n'induit pas de clique.

■

1.2.2 Frontières

Les séparateurs minimaux constituent un outil puissant pour l'étude des décompositions arborescentes. Cependant, ils ne sont pas suffisants pour celle des décompositions en branches. C'est pourquoi nous préférons souvent à cette notion celle, plus générale, de frontière.

Définition 1.20 (Frontière, séparation)
La frontière de $\{E_1, E_2\}$ est l'ensemble des sommets incidents à E_1 et à E_2.
Une séparation d'un graphe *G* est une partition de $E(G)$ en deux éléments $\{E_1, E_2\}$. La frontière d'une séparation est la frontière d'un de ses éléments. Une frontière est un ensemble de sommets qui est la frontière d'une séparation.

Propriété 1.21 ([Maz04]) Soit *S* un séparateur minimal d'un graphe *G*. Il existe une séparation de *G* dont la frontière est *S*.

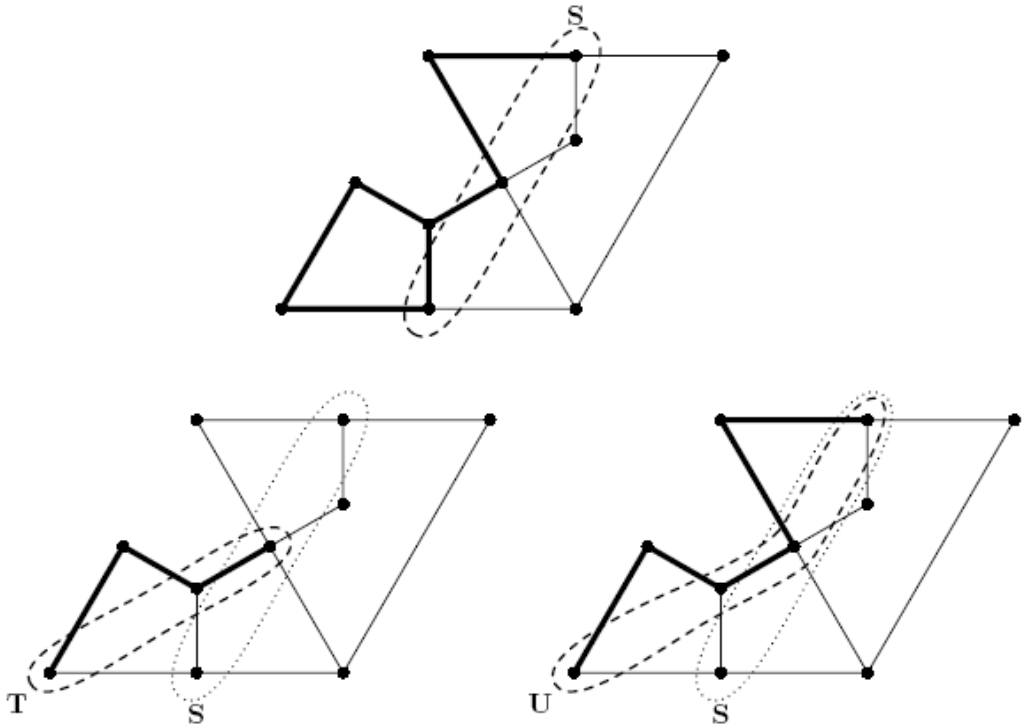
□ Soit *C* une composante pleine de $\zeta_G^*(S)$. Notons *A* l'ensemble des arêtes de *G* incidentes à au moins un sommet de *C*. La composante *C* étant pleine, chaque sommet de *S* est incident à au moins une arête de *A*. De plus, comme *S* est un séparateur minimal, il admet au moins une seconde composante pleine *D*. L'ensemble $E \setminus A$ contient toutes les arêtes incidentes à *D* donc tout sommet de *S* est incident à une arête de $E \setminus A$; la frontière de *A* est *S*.

■

La notion de frontières étend donc bien celle de séparateurs minimaux mais pour les utiliser, nous devons encore les munir d'une notion de parallélisme qui étende celle définie pour les séparateurs minimaux. Pour cela, nous pourrions

utiliser la définition 1.16 ou la propriété 1.18. Ces notions sont bien distinctes comme l'illustre la Figure 1.12.

Nous introduisons donc une notion de parallélisme intermédiaire qui étend la notion de parallélisme pour les séparateurs minimaux tout en palliant aux inconvénients mentionnés ci-dessus.



La frontière *T* est parallèle à la frontière *S* au sens de la définition 1.16 et de la propriété 1.18.
La frontière U est parallèle à la frontière *S* au sens de la définition 1.16 mais pas au sens de la propriété 1.18.
Figure 1.12 - Deux notions possibles de parallélisme de frontières

Définitions 1.22 (Parallélisme de frontières)
Soient *S* et *T* deux séparateurs de *G*. On dit que *S* est parallèle à *T* si *S* est inclus dans *T* ou si *S* est inclus dans $C \cup N(C)$ où *C* est une composante connexe de $G \setminus T$.

Remarque 1.23 La notion de parallélisme que nous venons de définir n'est pas symétrique pour les frontières.

Les frontières étant issues d'ensembles d'arêtes, nous définissons aussi une relation de chevauchement pour les ensembles d'arêtes.

Deux sous-ensembles *A* et *B* de *V* se chevauchent (dans *V*) si les ensembles $A \cap B$, $A \setminus B$, $B \setminus A$ et $V \setminus (A \cup B)$ sont tous non vides. S'ils ne se chevauchent pas, ils vérifient l'une des quatre conditions suivantes : $A \subseteq B$,

$B \subseteq A$, $A \cap B = \phi$ ou $A \cup B = V$. Quand il n'y a pas d'ambiguïté, nous disons seulement les ensembles « *A* et *B* se chevauchent » ou « *A* et *B* ne se chevauchent pas ».

Lemme 1.24 Si les ensembles *A* et *B* ne se chevauchent pas dans *V*, les ensembles *A* et $V \setminus B$ ne se chevauchent pas non plus.

□ Soient *A*, *B* inclus dans *V* ne se chevauchant pas.

– Si $A \subseteq B$, alors $A \cap (V \setminus B) = \phi$;
– Si $B \subseteq A$, alors $A \cup (V \setminus B) = V$;
– Si $A \cap B = \phi$, alors $A \subseteq (V \setminus B)$;
– Si $A \cup B = V$, alors $(V \setminus B) \subseteq A$.

Dans tous les cas, les ensembles *A* et $V \setminus B$ ne se chevauchent pas.

Puisque la relation de chevauchement est stable par passage au complémentaire, nous pouvons l'étendre aux séparations en posant que $\mathrm{E} = (E_1, E_2)$ chevauche $F = (F_1, F_2)$ si l'un des éléments de E chevauche l'un des éléments de la séparation *F*.

■

Définition 1.25 (Taille d'une frontière)
La taille d'une frontière est son cardinal.

On peut utiliser une séparation pour découper un graphe *G*.

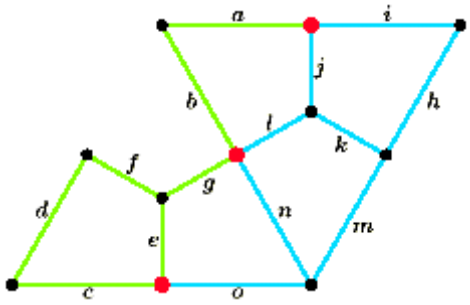


Figure 1.13 - Graphe simple avec 11 sommets et 14 arêtes.

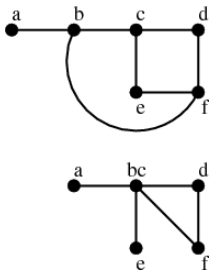
La Figure 1.13 montre pour :

E_1 : est l'ensemble des arêtes : a, b, c, d, e, f g ;

E_2 : est l'ensemble des arêtes : h, i, j, k, l, m, n et o.

On a la taille de sa frontière est égale à 3 (la cardinalité des sommets incidentes en même temps à E_1 et E_2).

Définition 1.26 (les mineurs d'un graphe)
Un graphe est mineur d'un graphe *G* s'il est obtenu en retirant des arêtes ou des sommets ou en contractants des arêtes de *G*.



Un exemple de deux graphes, dont l'un (celui du dessous) est un mineur de l'autre. Le mineur a été obtenu en supprimant l'arête [e,f] et en contractant l'arête [b,c]

1.3 Triangulation et triangulation minimal

Définition 1.27 (Graphes *k*-cordaux et graphes triangulés)
Un graphe est *k*-cordal s'il ne possède pas de cycle induit de longueur plus grande que *k*. Ceci signifie que tout cycle de longueur supérieure à *k* possède une corde. La cordalité d'un graphe *G* est le plus entier *k* tel que *G* soit *k*-cordal.
Cas particulier Un graphe est triangulé si et seulement si, il est *3*-cordal.

Les graphes triangulés (en anglais Chordal Graphs) ont une structure fortement arborescente. Ils sont souvent considérés comme des arbres généralisés.

Définition 1.28 (Arbre de Cliques)
Un arbre de cliques d'un graphe triangulé *G* est un arbre *T* dont chaque nœud est une clique maximale de *G* et satisfaisant la propriété d'intersection des cliques : ˝Soit *A* et *B* deux cliques maximales, l'ensemble *A\B* est contenue dans toutes les cliques sur la chaîne reliant *A* et *B* dans *T*˝.

Dans la suite de cette mémoire, nous noterons un arbre de cliques par *T*.

Théorème 1.29 (**[Gav74])** Un graphe est triangulé si et seulement si, il admet un arbre de cliques.

Dans la Figure 1.14 ci-dessous, nous présentons un graphe triangulé (tous ses cycles induits sont de longueur 3), son ensemble de cliques maximales et un arbre de clique correspondant.

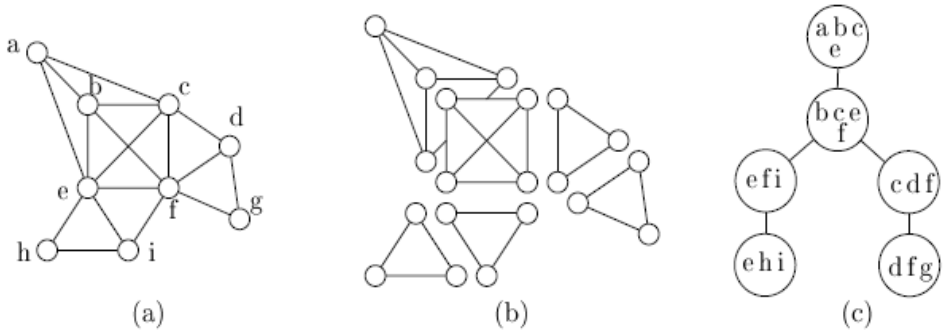


Figure 1.14 - Un graphe triangulé, son ensemble de cliques maximales et un arbre de cliques.

Il faut souligner que pour un graphe donné, on peut généralement associer plusieurs arbres de cliques non isomorphes[3]. Cette notion d'arbre de cliques des graphes triangulés nous sera très utile par la suite lorsque nous ferons du routage dans les graphes triangulés.

D'autant plus que, comme le montre le théorème suivant, ils sont calculables en temps linéaire :

Théorème 1.30 (**[BP94])** Pour tout graphe triangulé *G*, il est possible de calculer un arbre de cliques de diamètre minimum en temps linéaire.

Une autre caractérisation des graphes triangulés est donnée par les ordres d'élimination simplicielle.

Définition 1.31 (Sommet Simplicial)
Un sommet *u* est dit simplicial si son voisinage est une clique. Un ordre $x_1, x_2, ..., x_n$ sur les sommets de *G* tel que $\forall i \in \{1,...,n\}$, x_i est simplicial dans $G[\{x_i, ..., x_n\}]$ est appelé ordre d'élimination simplicielle de *G*.

Théorème 1.32 ([Gol80]) Un graphe est triangulé si et seulement si, il admet un ordre d'élimination simplicielle

La Figure 1.15 ci-dessous présente un ordre d'élimination simplicielle sur le graphe triangulé de la Figure 1.9 : x_1 étant le sommet 1, x_2 le sommet 2,…, x_9 le sommet 9.

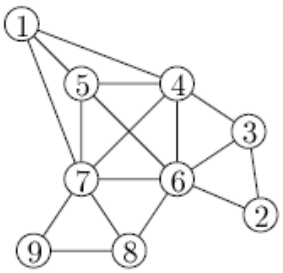


Figure 1.15 - Un ordre d'élimination simplicielle ($x_i = i$) sur un graphe triangulé.

Cette propriété est à la base des algorithmes tels que LexBFS (voir [RTL45] ou bien dans [CHR98]) qui est capable de déterminer, en temps linéaire, si un graphe est triangulé ou non.

Définition 1.33 (Triangulations et triangulations minimales)
Soit $G = (V, E)$ un graphe quelconque. Le graphe $G' = (V', E')$ est une triangulation de *G* si G' est un graphe triangulé, si $V = V'$ et si $E = E'$.

[3] Deux graphes sont isomorphe s'il existe une fonction bijective *f* de *A* vers *B*. Si *a*, *b* sont adjacents alors *f*(*a*) et *f*(*b*) sont adjacents. Le nombre de nœuds, d'arc et de degré doit être respectivement identiques. (Matrice d'adjacence).

Soit $G' = (V', E')$ une triangulation de *G*, G' est une triangulation minimale de *G* si et seulement si pour tout *F* tel que $F \subset E'$, le graphe $H = (V, F)$ n'est pas une triangulation de *G*.

Ainsi dans la Figure 1.16, les trois graphes de droites sont des triangulations minimales du cycle à 6 sommets. En effet ce sont des graphes triangulés, et si on enlève une des arêtes ajoutées, ils ne le sont plus.

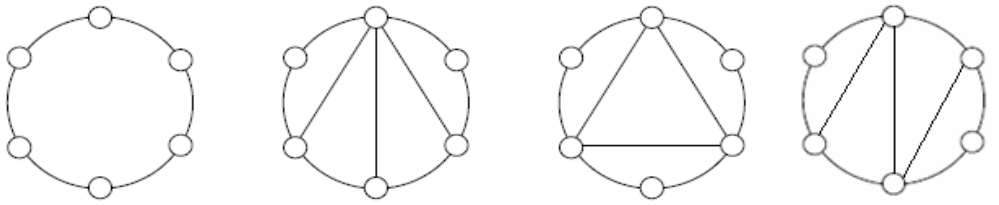


Figure 1.16- Le cycle à 6 sommets et trois triangulations minimales possibles.

Théorème 1.34 [RTL76] Pour tout graphe *G*, il est possible de construire en temps $O(nm)$ une triangulation minimale de *G*.

Définition 1.35 (Cliques maximales potentielles et treewidth)
Calculer le treewidth d'un graphe revient à trouver une triangulation (i.e. un sur-graphe triangulé) de largeur minimum, où la largeur est la taille de la clique maximum du sur-graphe, moins un. Le problème est NP-complet en général. Toutefois, des algorithmes polynomiaux existent pour certaines classes de graphes avec un nombre polynomial de séparateurs minimaux et il a été conjecturé que le calcul du treewidth est polynomial pour tous les graphes avec peu de séparateurs minimaux.

Les cliques maximales potentielles d'un graphe, qui sont en fait les cliques maximales des triangulations minimales du graphe respectif. Il est montré que si l'on sait énumérer en temps polynomial les cliques maximales potentielles pour certains graphes, le treewidth de ces graphes se calculent en temps polynomial. Bouchitté et Todinca [BT98] ont donné des algorithmes polynomiaux énumérant les cliques maximales potentielles pour toutes les classes de graphes dont on sait calculer le treewidth, et pour les graphes faiblement triangulés[4], pour lesquels le problème était ouvert. Néanmoins, nous ne savons toujours pas énumérer ces objets pour tous les graphes ayant peu de séparateurs.

Problème : Si *G* est un graphe, on note *r(G)* le nombre de ses séparateurs minimaux et *n* le nombre de sommets de *G* (un ensemble de sommets *S* est un séparateur minimal s'il existe deux composantes connexes distinctes *C* et *D* de *G\S* de sorte que tout sommet de *S* ait des voisins dans *C* et dans *D*). La question : Existe-t-il un polynôme *P* tel que pour tout graphe *G* on puisse trouver une suite

[4] Un graphe *G* est dit faiblement triangulé si, à la fois dans *G* et dans son complémentaire $\overline{G}$, tout cycle ayant au moins cinq sommets possède une corde (une arrête reliant deux sommets non consécutifs).

de graphes $G_0=G$, G_1, ..., G_k, où chaque G_{i+1} est obtenu de G_i par rajout d'une arête, G_k est une clique et pour tout i, $r(G_i) < P(n,r(G))$?.

2. Enumération des Séparateurs minimaux

Les séparateurs sont fondamentaux dans l'étude des graphes non-orientés, notamment comme outils de décomposition et de triangulation. Ils peuvent se voir comme un central par lequel passe toute communication entre différents groupes (sommets du graphe).
Nous rappelons dans ce chapitre les principales notions sur les séparateurs, classiquement axées sur les séparateurs de deux sommets, et aux séparateurs de *k* sommets.

Les algorithmes de calcul de la largeur arborescente et de la complétion minimale passent presque tous par la notion de séparateur minimal. Nous définissons ici ces objets et nous étudions en détail les séparateurs minimaux des graphes triangulés. Nous montrons ensuite comment énumérer tous les séparateurs minimaux d'un graphe.

2.1. Sous-ensembles d'articulation

Définition 2.1 (Point d'articulation) On appelle point d'articulation d'un graphe *G***,** un sommet dont le retrait déconnecte *G*.

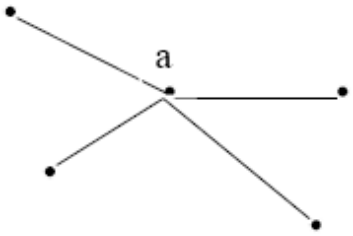


Figure 2.1- Exemple de sommet d'articulation

La Figure 2.1 montre que le sommet *a* est un point d'articulation pour le graphe.

Définition 2.2 (Sous-ensemble d'articulation) On appelle sous-ensemble d'articulation ou séparateur, un sous-ensemble S de X dont la suppression déconnecte G. (i.e. $G[X \setminus S]$ non connexe). Voir la Figure 2.2.

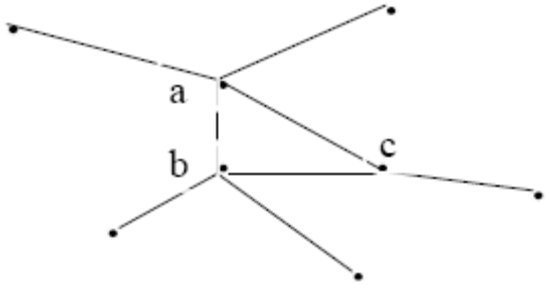


Figure 2.2- Exemple de sous-ensemble d'articulation

Dans la Figure 2.2 $\{a, b, c\}$ est un sous-ensemble d'articulation.

2.2 Séparateurs minimaux

2.2.1 Généralités sur les séparateurs minimaux

Remarquons qu'un séparateur minimal peut être strictement contenu dans un autre séparateur minimal, voir Figure 2.3 où le *h,g*-séparateur minimal $\{c\}$ est strictement inclus dans le *a,g*-séparateur minimal $\{c,f\}$.
Sur la Figure 2.3, les composantes connexes de $G\setminus T$ sont $\{a, b, e, d\}$, $\{g\}$ et $\{h\}$, et les deux premières sont pleines par rapport à T. Pour un sommet $a \in \{V \setminus S\}$de G, nous noterons $\zeta_G^a(S)$ la composante connexe de $G\setminus S$ contenant a.

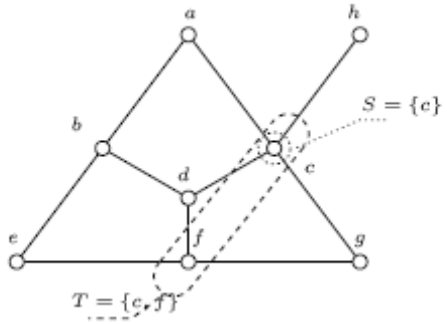


Figure 2.3- Séparateurs minimaux

Proposition 2.3 ([Tod99]) Soient $S \subset V$ un ensemble de sommets et C, D deux composantes connexes différentes de $G \setminus S$ noté $\zeta_G(S)$. Les propositions suivantes sont équivalentes :

1. C et D sont des composantes connexes pleines associées à S ;
2. Il existe $a \in D$ et $b \in D$ tels que S soit un *a,b*-séparateur minimal ;
3. Pour tout $a \in D$ et $b \in D$, S est un *a,b*-séparateur minimal.

Corollaire 2.4 ([Tod99]) S est un séparateur minimal de G si et seulement si S a au moins deux composantes pleines.

La propriété qui suit donne une explication sur l'existence des séparateurs minimaux qui ne sont pas minimaux par inclusion. Ceux-ci sont exactement les séparateurs minimaux induisant aussi des composantes non-pleines :

Proposition 2.5 ([Tod99]) Soient *G* un graphe, *S* un séparateur minimal de *G* et *C* une composante connexe de $G \setminus S$, non pleine par rapport à *S*. Alors l'ensemble de sommets $S' \subseteq S$ ayant au moins un voisin dans *C* est un séparateur minimal de *G*.

Comme *S* est un séparateur minimal, il induit une composante connexe pleine *D* dans *G*. De la même façon que dans la preuve de la proposition précédente, *C* est une composante pleine associée à S'. D'un autre côté, *D* est inclus dans une autre composante connexe, toujours pleine, associée à S'.
Par le corollaire précédent, S' est un séparateur minimal de *G*.

Remarquons que l'on peut regarder un *a,b*-séparateur *S* comme un ensemble de sommets tel que toute chaîne de *a* à *b* passe par *S*. De plus *S* est un *a,b*-séparateur minimal si et seulement si pour tout $x \in S$, il existe une chaîne de *a* à *b* qui intersecte *S* uniquement en *x*.

2.2.2 Séparateurs minimaux des graphes triangulés

Dirac a donné en 1961 une première caractérisation des graphes triangulés à l'aide des séparateurs minimaux [Dir61] :

Théorème 2.6 ([Dir61]) Un graphe *G* est triangulé si et seulement si tous ses séparateurs minimaux sont des cliques.

En fait, le x_1, x_3 – séparateur minimal contenu dans le voisinage de x_1 est unique, comme le montre la proposition suivante.

Proposition 2.7 ([Tod99]) Pour tout couple (*a,b*) de sommets non adjacents il existe un unique *a,b*-séparateur minimal contenu dans *N*(*a*).

Propriété 2.8 Pour toute paire (a,b) fixée de sommets de *V* non reliés, il y a au moins un *a,b*-séparateur minimal. (En général, il y en a plusieurs.)

□ Le voisinage $N(a)$ de a est séparateur du graphe, puisqu'il n'y a pas d'arc entre *a* et *b*. Il est *a,b*-séparateur, et contient donc un *a,b*-séparateur minimal. (En fait, il en contient un unique.)

■

Remarque 2.9 Nous pouvons dire que le séparateur minimal *S* est le plus proche de *a* parmi les *a,b*-séparateurs.

Le théorème de Dirac est expliqué de manière beaucoup plus structurelle par le très important théorème qui suit, qui se trouve dans les travaux de Ho et Lee [HL89] et Lundquist [Lun90] :

Théorème 2.10 ([Dir61]) Soit *G* un graphe triangulé et soit *T* un arbre de cliques quelconque de *G*. Un ensemble de sommets *S* est un séparateur minimal de *G* si et

seulement si *S* s'écrit sous la forme $\Omega \bigcap \Omega'$, où Ω et Ω' sont des cliques maximales de *G*, adjacentes dans l'arbre de cliques *T*.

Avant d'aborder la preuve, donnons un lemme simple sur la structure d'un arbre de cliques. Si *T* est un arbre de cliques d'un graphe triangulé *G* et Ω et Ω' sont deux cliques adjacentes dans *T*, nous allons noter T_Ω et $T_{\Omega'}$ les sous-arbres de *T* obtenus en enlevant l'arête $\Omega\Omega'$ (L'arête qui relie les deux cliques), où $\Omega \in T_\Omega$ et $\Omega' \in T_{\Omega'}$.

Posons *V* et V' les ensembles de sommets de *G* qui appartiennent à au moins une étiquette[1] de T_Ω, respectivement $T'_{\Omega'}$:

Lemme 2.11 ([BT01b]) $S = \Omega \bigcap \Omega'$ sépare dans *G* tout $a \in V_\Omega \setminus S$ de tout sommet $b \in V_{\Omega'} \setminus S$.

2.3 Le treillis des *a,b*-séparateurs minimaux

Définitions 2.12 (Les treillis)

Un ensemble est appelé treillis s'il est ordonné et que tout couple d'éléments possède une borne supérieure et une borne inférieure.

Soit un graphe simple *G*, et deux sommets distincts *x* et *y* dans *G*, l'ensemble des parties de G contenant x mais pas y est un treillis pour la relation d'inclusion : l'inf de deux ensembles est leur intersection et leur sup est leur union. Pour les *a,b*-séparateurs minimaux, la situation est comparable.

Remarques 2.13

- La notion d'*a,b*-séparateur minimal est d'essence non-orientée: un *a,b*-séparateur minimal sera aussi *b,a*-séparateur minimal.
- Sous-ensemble d'articulation minimal $\Rightarrow$ *a,b*-séparateurs minimal.

i.e.:

Un *a,b*-séparateur minimal n'est pas forcément sous-ensemble d'articulation minimal: on peut avoir *S* et *T* deux séparateurs minimaux tel que $S \not\subset T$. *S* et *T* ne seront pas alors séparateurs minimaux pour une même paire de sommets : $\exists$ *a*, *b* tel que *T* est *a,b*-séparateur minimal, mais *T* ne sera pas 'sous-ensemble d'articulation minimal'.

Par contre, si *S* est sous-ensemble d'articulation minimal, il sera *a,b*-séparateur minimal pour toute paire *(a,b)* tel que *a* et *b* appartiennent à deux composantes connexes distinctes de $\zeta_G(S)$.

[1] Les étiquettes (labels) sont simplement des noms donnés aux sommets et aux arêtes de façon à pouvoir les différencier. L'étiquetage des sommets est généralement arbitraire

Exemple 2.14

S est sous-ensemble d'articulation minimal :

Le retrait de *S* déconnecte le graphe, mais ne sépare pas a de *c*.

T est *a,c*-séparateur minimal.

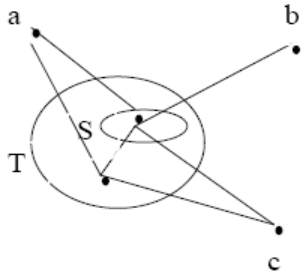
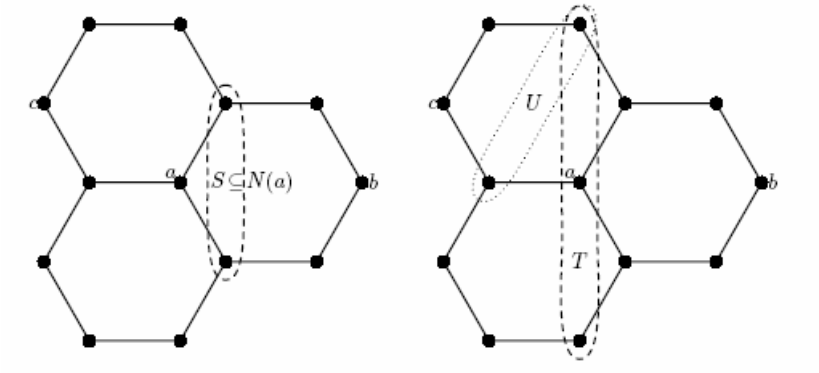


Figure 2.3-Exemple de séparateur minimal

Définition 2.15 (Séparateur proche)

Soient S et T deux *a,b*-séparateurs minimaux d'un graphe $G=(V,E)$. Le séparateur S est plus proche de a que T (noté $S \preccurlyeq_a T$) si $C_G^a(S)$est inclus dans $C_G^a(T)$.



Le séparateur $S = N_G\left(C_G^a(N(a))\right)$ est proche de a.

Le séparateur $U = N_G\left(C_G^b(S \cup N(a))\right)$ est proche de T.

Exemple - Séparateurs proches d'un sommet ou d'un autre séparateur

Proposition 2.16 ([KK98]) L'ensemble des *a,b*-séparateurs minimaux d'un graphe G forme un treillis pour la relation d'ordre $\preccurlyeq_a$.

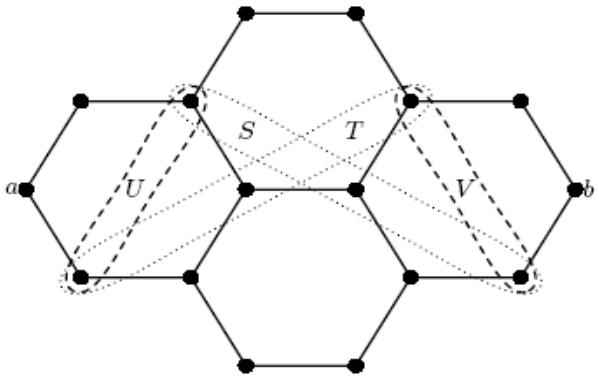


Figure 2.4- L'inf et le sup de deux séparateurs minimaux

Le séparateur $U = N_G\left(C_G^a(S \cup T)\right)$ est l'inf de *S* et *T*.
De même, $V = N_G\left(C_G^b(S \cup T)\right)$ est le sup de *S* et *T*.

2.4 Généralisation de la notion de séparateur minimal

2.4.1 L'ensemble des Séparateurs Minimaux de deux sommets fixés

Exemple 2.17
{*c,f*} , {*c,e*} , {*d,f*} , {*d,e*} sont tous *a,b*-séparateurs minimaux .

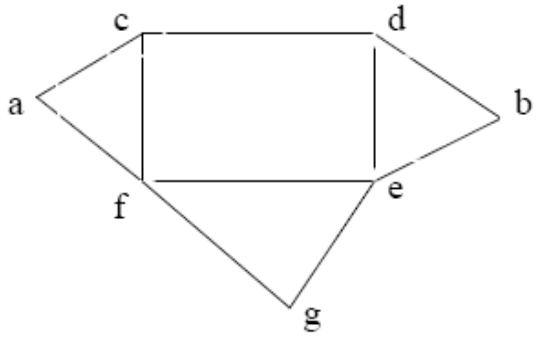


Figure 2.4-Exemple des séparateurs minimaux

Si l'on privilégie l'un des sommets (*a* par exemple), on peut établir sur ces séparateurs la relation d'ordre suivante :
$S_1 \leq S_2$ si et seulement si $\zeta_G^a(S_1) \subseteq \zeta_G^a(S_2)$.

i.e. :
'S_1 a une plus petite composante connexe contenant *a* que S_2'.

Cela définit un treillis (voir [Pau94]), avec des propriétés particulières pour certaines classes de graphes. (Ce treillis forme une chaîne pour un graphe triangulé ou de permutation.)

Sur l'exemple 2.17

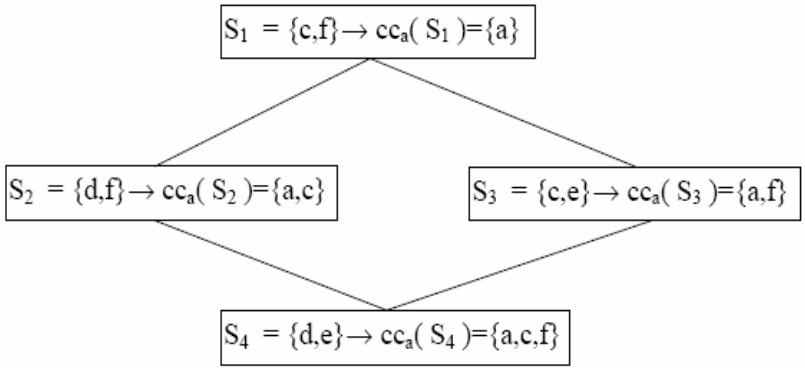


Figure 2.5-Sur Exemple 2.17

On remarquera que *{c,e}* et *{d,f}* sont incomparables: le graphe n'est pas triangulé, et c'est justement en *cdef* qu'il apparaît un cycle sans corde de longueur 4.

2.4.2 Séparateur minimal de *k* sommets

On va maintenant proposer une nouvelle définition, plus générale, des séparateurs minimaux :

On considérera les séparateurs qui sont minimaux pour déconnecter *k* sommets les uns des autres (au lieu de deux sommets classiquement).

***k*-séparateur minimal**

Définition 2.18 (*k*-séparateur minimal)
On appellera $x_1, x_2, ..., x_k$-séparateur minimal un séparateur *S* dont la suppression laisse les sommets $x_1, x_2, ..., x_k$ dans *k* composantes connexes différentes de $G[V \setminus S]$, et qui est minimal (au sens de $\subseteq$) pour cette propriété.

Exemple 2.19
$\{s_1\}$ est *a,b*-séparateur minimal et *b,c*-séparateur minimal.
Par contre, son retrait ne déconnecte pas *a* de *c*.
S est *a,b,c*-séparateur minimal ;
(et il est aussi *a,c*-séparateur minimal).

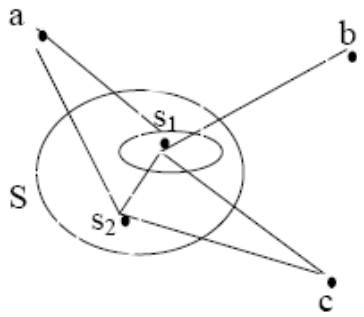


Figure 2.6-Exemple de *k*-séparateur minimal

Remarque 2.20
On pourrait définir pour k=1 le '*1*-séparateur minimal' dont le retrait 'déconnecte' en une seule composante connexe: ce '*1*-séparateur minimal' serait unique et égal à ϕ.

Remarque 2.21
Un séparateur peut être en même temps *k*-séparateur minimal et *j*-séparateur minimal, $j \neq k$.
(Sur l'exemple précédent, *S* est à la fois *2*-séparateur minimal et *3*-Séparateur minimal).

Propriété 2.22 (*Les arêtes sortant d'un k-séparateur Minimal*)
Tout sommet d'un *k*-séparateur minimal *S* possède au moins deux arêtes allant vers deux composantes connexes différentes de $CC(S)$.

2.5 Énumération des séparateurs minimaux d'un graphe

Tout le travail de cette thèse est lié à la compréhension de la structure des séparateurs minimaux des graphes. On notera Δ_G l'ensemble des séparateurs minimaux du graphe *G*. Comme nous serons amenés à utiliser plus tard tous les séparateurs minimaux d'un graphe, nous esquissons ici l'algorithme de Kloks et Kratsch [KK94] qui les énumère en un temps polynomial en la taille du graphe et du nombre de ses séparateurs minimaux.

Cet algorithme procède en énumérant les *a,b*-séparateurs minimaux pour tout couple de sommets *a* et *b*. Sachant que nous ne disposons d'aucun algorithme d'énumération qui travaille directement sur la totalité des séparateurs minimaux d'un graphe. Ceci vient probablement du fait que nous ne savons pas munir Δ_G d'une bonne structure, par exemple d'une structure d'ordre. La situation est complètement différente si l'on considère seulement les *a,b*-séparateurs minimaux de *G*, pour un couple de sommets *a* et *b* fixés. Cet ensemble, noté $\Delta_G(a,b)$, peut être muni de façon très naturelle d'une structure de treillis.

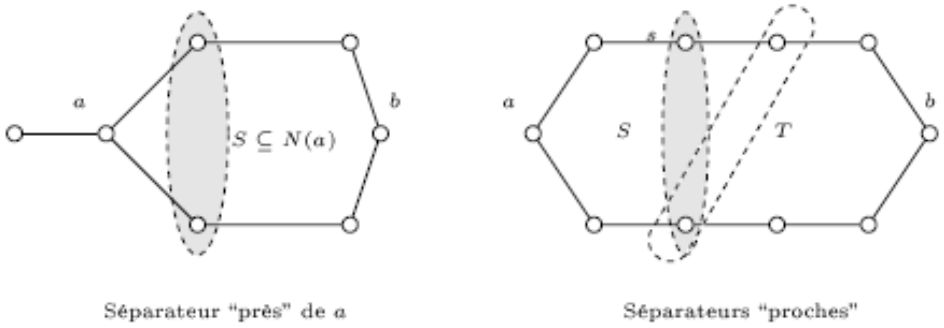


Figure 2.7-Structure des *a,b*-séparateurs minimaux

Comme nous l'avons fait remarquer (proposition 2.7 et remarque 2.8), il existe un unique *a,b*-séparateur minimal qui est le plus proche de *a*, c'est-à-dire qui est contenu dans $N_G(a)$. Il s'agit du séparateur formé par les sommets de $N_G(a)$ ayant au moins un voisin dans la composante connexe de $G \setminus N_G(a)$contenant *b*. Considérons maintenant un *a,b*-séparateur minimal *S*, qui n'est pas le plus proche de *b*. Il existe alors $s \in S$ non adjacent à *b*. Soit T' le *s,b*-séparateur minimal le plus proche de *s* dans le graphe $G' = G\left[C_G^b(S) \cup \{s\}\right]$et posons S' l'ensemble de sommets de *S* n'ayant pas de voisin dans $G_G^b(T')$. Kloks et Kratsch prouvent dans [KK94] que l'ensemble de sommets $T = S \cup T' \setminus S'$est un *a,b*-séparateur minimal de *G*. En quelque sorte, *T* est un *a,b*-séparateur minimal «proche de S ». De cette manière, en partant du *a,b*-séparateur minimal le plus proche de a et en allant de proche en proche, on obtient tous les *a,b*-séparateurs minimaux de *G*. Comme touts ces résultats sont constructifs, on trouve :

Théorème 2.23 ([KK94]) L'ensemble Δ_G des séparateurs minimaux de *G* peut être calculé en temps $O\left(n^5|\Delta_G|\right)$.

Cet algorithme, ainsi que des versions ultérieures [KK98, SL97], calculent tous les séparateurs minimaux d'un graphe en énumérant les *a*,*b*-séparateurs minimaux pour tout les couples de sommets *a* et *b*. La proposition 2.3 suggère qu'un séparateur minimal peut être *a*,*b*-séparateur minimal pour un nombre important de couples *a*, *b*. On pourrait donc espérer un algorithme beaucoup plus performant d'énumération de tous les séparateurs minimaux d'un graphe si on arrivait à les considérer globalement, et non pas comme l'union des *a*,*b*-séparateurs minimaux pour toutes les paires *a*, *b*. Mais, comme nous l'avons dit, nous ne sommes pas arrivés à munir les séparateurs minimaux d'un graphe d'une relation dont on saurait maîtriser la structure, alors que l'on peut prouver qu'une version de la relation de $a,b-$ séparateurs ‘‘ proches’’ décrite plus haut induit une structure de treillis sur $\Delta_G(a,b)$.

Une tentative intéressante d'ordonner les séparateurs minimaux d'un graphe est due à Berry [Ber95]. Elle utilise une autre notion, de ‘‘séparateur deux-coloriable’’, dont les séparateurs minimaux sont un cas particulier. On obtient un treillis sur ces objets, qui est en fait le treillis de Galois de la relation « *x* et *y* sont des sommets distincts, non adjacents, de *G*». Le bon côté est que chaque treillis des $a,b-$ séparateurs minimaux est un sous-ordre du treillis des séparateurs deux-coloriables. Le mauvais côté est que le nombre de séparateurs deux-coloriables peut être exponentiellement plus grand que celui des deux-séparateurs et surtout qu'un séparateur minimal peut être représenté par plusieurs séparateurs deux-coloriables, et c'est cette duplication de l'information qui engendre artificiellement la ‘‘bonne structuration’’.

La caractérisation des séparateurs minimaux à l’aide de composantes pleines (proposition 2.3) permet de construire certains *a*,*b*-séparateurs minimal à partir de *a*,*b*-séparateurs.

Lemme 2.24 ([Tod99]) Soit *R* un *a*,*b*-séparateur d’un graphe *G* tel que la composante connexe $C_G^a(R)$ soit pleine par rapport à *R*. L’ensemble $N\left(C_G^b(R)\right)$ est un *a*,*b*-séparateur minimal *S*.

En utilisant ce lemme, nous pouvons construire facilement certains séparateurs minimaux. Pour cela, il faut partir d’un *a*,*b*-séparateur dont la composante contenant *a* est pleine : c’est le cas du voisinage de *a* convient. Nous obtenons ainsi la première classe de séparateurs minimaux de Kloks et Kratsch.

Corollaire et définition 2.25 (Séparateurs proches d’un sommet) [Maz04]
Soient *G* un graphe et *x* un sommet de *G*. Les voisinages dans *G* des composantes connexes de $G\setminus\left(N(x)\cup\{x\}\right)$ sont des séparateurs minimaux.
Nous disons qu’un tel séparateur est proche de *x*.

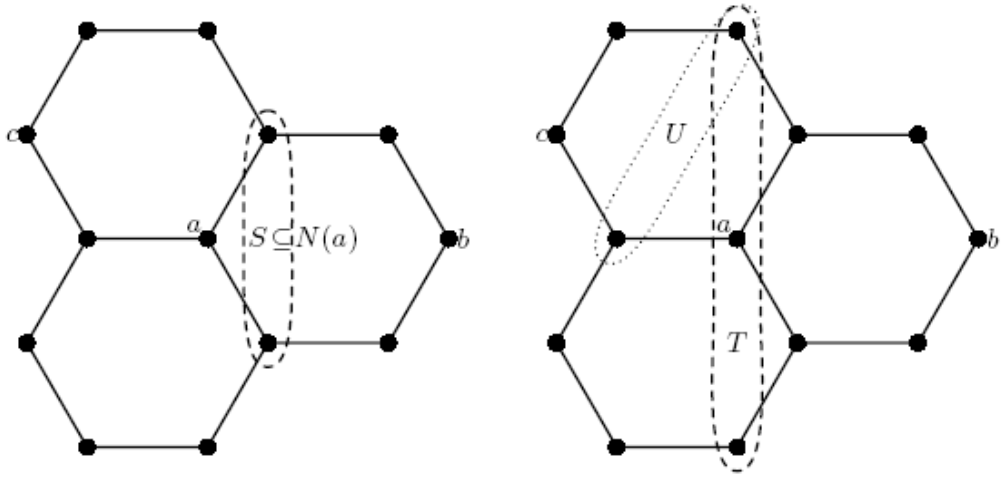
Lemme 2.26 ([Tod99]) Soient *a* et *b* deux sommets non adjacents d'un graphe *G*. Le voisinage de la composante connexe de $G \setminus N(a)$ contenant *b* est le plus petit a,b-séparateur minimal pour la relation $\preccurlyeq_a$.

Nous pouvons aussi utiliser le lemme 2.26 pour obtenir de nouveaux séparateurs minimaux à partir d'un séparateur minimal *S* donné. Pour cela, nous allons «pousser» un sommet de *S* en dehors du séparateur et ainsi obtenir le second type de séparateurs minimaux de Kloks et Kratsh.

Corollaire et définition 2.27 (Séparateur proche d'un autre [Maz04])
Soient *S* un *a*,*b*-séparateur minimal d'un graphe *G* et *x* un sommet de *S* non adjacent à *a*. Le voisinage $S_{a:x\rightarrow}$ dans *G* de la composante connexe $C_G^a(S \cup N(x))$est un séparateur minimal. Le séparateur $S_{a:x\rightarrow}$ est obtenu en poussant *x* loin de *a*. Nous dirons qu'il est proche de *S*.

□ Par construction, $S_{a:x\rightarrow}$ est inclus dans $S \cup N(x)$, $C_G^a(S_{a:x\rightarrow})$ est donc inclus dans. Ceci implique $C_G^b(S)$que est inclus dans $C_G^b(S_{a:x\rightarrow})$.
Comme de plus, *x* ne fait pas partie de $S_{a:x\rightarrow}$ et qu'il est voisin d'un sommet de $C_G^b(S)$, le sommet *x* appartient à $C_G^b(S_{a:x\rightarrow})$. Pour résumer :

- tous les sommets de S sont adjacents à un sommet de $C_G^b(S)$donc ils sont adjacents à un sommet de $C_G^b(S_{a:x\rightarrow})$;
- tous les sommets de N(x) sont adjacents à x donc ils sont adjacents à un sommet de $C_G^b(S_{a:x\rightarrow})$.



Le séparateur $S = N_G\left(C_G^b(N(a))\right)$ est proche de *a.*
Le séparateur $U = N_G\left(C_G^c(S \cup N(a))\right)$ est proche de *T*.

Figure 2.8- Séparateurs proches d'un sommet ou d'un autre séparateur

La composante connexe $C_G^b(S_{a:x\rightarrow})$ est pleine. Comme, par construction, $C_G^b(S_{a:x\rightarrow})$ l'est aussi et qu'elles sont distinctes, $S_{a:x\rightarrow}$ est bien un séparateur minimal.

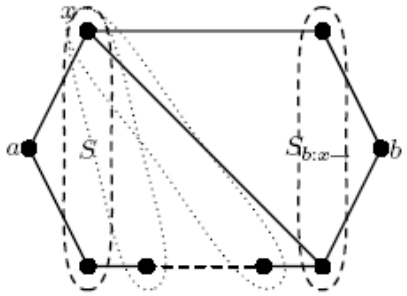


Figure 2.9 – Si proche et pourtant si loin.

Le séparateur $S_{b:x\rightarrow}$ est obtenu en poussant *x* loin de *b*. Il est donc proche de *S*. Pourtant, dans le treillis des *a*,*b*-séparateurs minimaux, tous les séparateurs représentés en pointillés sont entre S et $S_{b:x\rightarrow}$. Le séparateur $S_{b:x\rightarrow}$ peut donc être arbitrairement loin de *S*.

Lemme 2.28 ([KK98]) Soient *S* un a,b-séparateur minimal d'un graphe *G* et *x* dans $S \setminus N(b)$. Le séparateur $S_{b:x\rightarrow}$ obtenu en poussant *x* loin de *b* est le plus petit a,b-séparateur minimal plus grand que *S* et ne contenant pas x pour la relation $\preccurlyeq_a$.

2.3 L'algorithme d'énumération de Berry et col.

Si nous cherchons à énumérer les séparateurs minimaux d'un graphe et que nous disposons d'un séparateur minimal *S*, nous pouvons essayer de pousser un sommet *x* de *S* dans toutes les directions pour obtenir d'autres séparateurs minimaux. Les séparateurs ainsi obtenus sont les voisinages des composantes connexes de $G \setminus (S \cup N(x))$. En systématisant ce procédé à tous les sommets de *S* et à chaque nouveau séparateur obtenu, nous obtenons l'algorithme 2.1 [BBC00] dû à Berry, Bordat et Cogis. Les corollaires 2.25 et 2.27 certifient que cet algorithme calcule bien des séparateurs minimaux. La propriété 2.29 prouve la correction de cet algorithme.

Propriété 2.29 ([Ber98]) L'algorithme **enumeration** calcule tous les séparateurs minimaux d'un graphe donné.

□ Supposons par l'absurde que l'algorithme **enumeration** ne produit pas un a,b-séparateur minimal *S*. Dans la phase d'initialisation, **enumeration** calcule tous les séparateurs minimaux inclus dans les voisinages des sommets de graphe. En particulier, il calcule le a,b-séparateur minimal inclus dans le voisinage de *a*. L'algorithme calcule donc au moins un a,b-séparateur minimal. De plus, d'après le lemme 3.26, celui-ci est plus proche de a que *S*.

Parmi les a,b-séparateurs minimaux plus proches de a que S produits par l'algorithme, considérons S_c le a,b-séparateur minimal le plus loin de a possible.

Si S_c et S sont distincts, il existe un sommet x de $S_c \setminus S$ appartenant à la composante connexe $C_G^a(S)$. Si nous poussons le sommet x de b vers a, nous obtenons un nouveau séparateur minimal plus grand que S_c. Or d'après le lemme 2.28, celui-ci est plus petit que S ce qui contredit la maximalité de S_c. L'algorithme a donc bien calculé le séparateur minimal S.

■

Algorithme 2.1 enumeration

entrée :
$G = (V, E)$ // *un graphe connexe*
sortie :
S : // *l'ensemble des séparateurs minimaux de G*
début
$S \leftarrow \phi$;
$L \leftarrow \phi$; // *liste des séparateurs non traités*
pour chaque $x \in V$ et $C \in \zeta_G(x \cup N(x))$ **faire**
si $N(C) \notin S$ **alors**
$L \leftarrow L \cup N(C)$
$S \leftarrow S \cup N(C)$
tant que $L \neq \phi$ **faire**
soit $S \in L$
$L \leftarrow L \setminus \{S\}$
pour chaque $x \in S$ **et** $C \in \zeta_G(S \cup N(x))$ **faire**
si $N(C) \notin S$ **alors**
$L \leftarrow L \cup N(C)$
$S \leftarrow S \cup N(C)$
rendre S
fin

Propriété 2.30 ([BBC00]) L'algorithme **enumeration** peut être implémenté de façon à avoir une complexité en temps $O(nm)$ par séparateur.

Définition 2.31 (Graphe planaire)
Un graphe est **planaire** si on peut le dessiner sur la sphère sans que les arêtes ne se croisent.

Remarque 2.32 Dans le cas des graphes planaires, l'algorithme **enumeration** fonctionne en $O(n^2)$ par séparateur. En effet, comme le degré moyen d'un sommet d'un graphe planaire est strictement inférieur à six, le nombre d'arêtes d'un tel graphe est inférieur à trois fois le nombre de sommets[2].

[2] Plus précisément, le degré moyen est au plus égal à $6 - \frac{12}{n}$ et le nombre maximal d'arêtes est $3n - 6$. Pour une démonstration de ces résultats, nous renvoyons le lecteur à [Die00].

3. Décomposition de graphes

Diviser pour régner
Machiavel, 1532

Le principe d'un schéma de décomposition est de diviser un problème de façon récursive en sous-problèmes, de résoudre chacun des sous problèmes puis de fusionner les résultats partiels pour obtenir une solution générale. Un exemple d'une telle technique est le « tri fusion » : pour trier une liste *l*, la découper en deux listes l_1 et l_2 de tailles équivalentes ; trier chacune des deux listes puis fusionner les deux sous-listes triées.

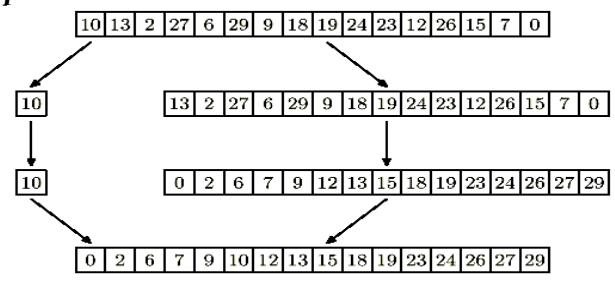
Pour qu'une telle technique soit efficace, il faut s'arranger pour que les sous-problèmes se résolvent bien de façon inductive et pour que l'opération de fusion se fasse bien. Ainsi, la façon dont le problème principal est découpé est très importante. Dans le cas du tri fusion, si la liste est découpée de façon totalement déséquilibrée en formant une liste de taille 1 et une seconde de taille *n-1*, nous obtenons le tri par insertion dont la complexité en moyenne est $O(n^2)$ alors qu'elle est $O(n\ ln(n))$ pour le tri fusion.

□ **Exemple 3.1 (Les tris)**

Pour trier une liste, on peut :

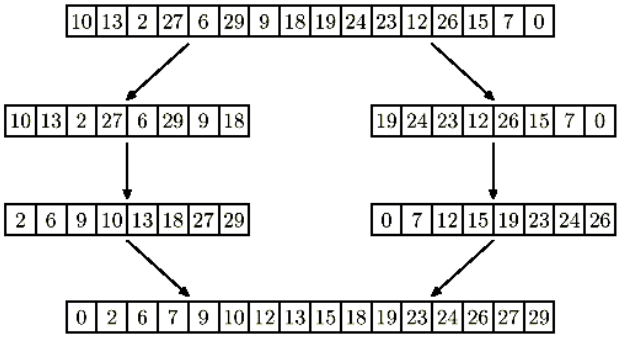
- Diviser la liste en deux sous-listes ;
- Trier récursivement les deux sous-listes ;
- Fusionner les sous-listes triées.

Le tri par insertion



Complexité : $O(n^2)$.

Le tri fusion



Complexité : *O(n log n).*

■

Il y a plusieurs types de décompositions de graphes : la décomposition linéaire, la décomposition en branche, la décomposition arborescente,…etc. Mais nous nous somme intéressés à étudier deux types de décomposition ; la décomposition arborescente et la décomposition en branches.

3.1 Décomposition arborescente

La notion de décomposition arborescente a été introduite par Robertson et Seymour en 1986 lors de leurs travaux sur les mineurs de graphes [RS86].

Notations

V_T : Ensemble de sommets de *T*.

$\bigcup_{u \in V_T}$: Union de tous les sommets de l'arbre *T*.

Définition 3.2 (Décomposition arborescente)

Une décomposition arborescente d'un graphe *G = (V,E)* est un arbre *T* dont les nœuds, appelés sacs (en anglais bags) sont des sous-ensembles de sommets de *G*. Cet arbre respecte les trois règles suivantes :

1. Pour tout sommet *u* de *G*, il existe au moins un sac contenant *u* : $\bigcup_{u \in V_T} = V$,
2. Pour toute arête (u,v) de *E*, il existe au moins un sac contenant *u* et *v* : il existe $X \in V_T$ tel que $u, v \in X$,
3. Pour tout sommet *u* de *G*, l'ensemble des sacs contenant *u* induit un sous-arbre de *T* : $X, Y, Z \in V_T$ si Y est sur le chemin de *X* à *Z*, alors $X \cap Z \subseteq Y$.

$3 \Leftrightarrow$ Pour tout sommet *u* de G , l'ensemble des sacs contenant *u* induit un sous-arbre de *T*.

Exemple 3.3 (décomposition arborescente)

La Figure 3.1 présente un graphe *G* et une décomposition arborescente *T* de *G*. Il est en effet facile de voir que *T* vérifie les trois règles de la définition 3.1. Cette figure présente aussi la triangulation de *G* pour laquelle *T* est un arbre de cliques.

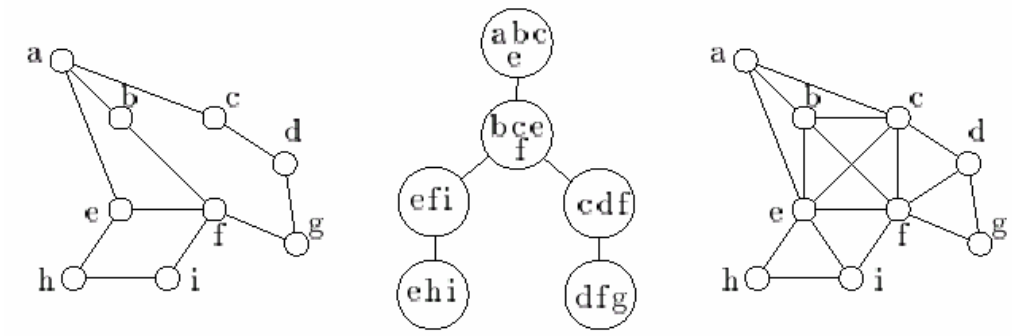


Figure 3.1 - Un graphe *G*, une décomposition arborescente *T* de *G* et la triangulation correspondante.

Remarque 3.4

Pour décomposer le graphe *G* :

- On complète un séparateur *S* en une clique ;
- On considère les composantes connexes C_i de $G\backslash S$ pour construire les graphes G_i ;
- On décompose les graphes de G_i.

Algorithme 3.1 decomposition_arborescente

entrée :

G = (V,E) //un graphe connexe.

sortie :

T : //une décomposition arborescente de *G*

début

si *G* est une clique **alors**

T est l'arbre réduit à un nœud contenant *V*

sinon

$S \leftarrow$ un *a,b*-séparateur minimal de *G*

Compléter *G[S]* en une clique

$T \leftarrow$ l'arbre réduit à un nœud *x* contenant *S*

pour chaque composante connexe C_i de $G\backslash S$ **faire**

$T_i \leftarrow$ decomposition_arborescente $(G(C_i \cup N(C_i))$

$y_i \leftarrow$ un nœud de T_i dont l'étiquette contient $N(C_i)$

attacher T_i à *T* en reliant *x* et y_i

rendre *(T)*

fin

3.1.1 Les invariants connus liés à la décomposition arborescente

3.1.1.1 La largeur arborescente

Autour de la largeur arborescente se sont développées une théorie et une littérature très volumineuses. En effet de nombreux problèmes de

l'algorithmique des graphes connus pour être NP-difficiles en général devient polynomiaux ou linéaires dès lors que le graphe considéré est de largeur arborescente bornée. Des travaux ont montré par exemple que tout problème exprimable par des formules de la logique monadique étendue du second ordre peut être résolu en temps linéaire dans des familles de graphes de largeur arborescente bornée.

Malheureusement, trouver une décomposition arborescente de largeur minimum est un problème NP-difficile, et ce même dans des classes de graphes restreintes comme les graphes de degré borné, les graphes bipartis ou les graphes de co-comparabilité.

Il existe tout de même certaines familles de graphes pour lesquelles il est possible de calculer la largeur arborescente par exemple le graphe complet est de largeur arborescente *n-1*, c'est le pire des cas. Les arbres eux, sont exactement les graphes de largeur arborescente *1*, ceci est censé justifier le *« 1 »* dans la définition de la largeur arborescente.

La largeur arborescente d'un graphe peut se caractériser à l'aide des triangulations de la manière suivante :

Théorème 3.5 ([Gol80]) La largeur arborescente de *G* est égale au minimum, parmi toutes les triangulations possibles *G'*, de $\omega(G') - 1$.

Le calcul de la largeur arborescente d'un graphe *G* peut donc se faire en cherchant des triangulations de *G* qui minimisent la taille de la clique maximum. On peut même se limiter à chercher parmi les triangulations minimales de *G*. C'est pourquoi celles-ci sont très largement étudiées dans la littérature.

Définition 3.6 (Largeur arborescente) La largeur d'une décomposition arborescente est le nombre maximum de sommets contenus dans un de ses sacs moins 1 : *largeur(T)*= $\max_{B \in V(T)} \{|B| - 1\}$.

La largeur arborescente d'un graphe *G* (en anglais tree-width) est la plus petite des largeurs de toutes les décompositions arborescentes possibles de G :
$tw(G) = \min_T \{largeur(T)\}$.

Objectif La largeur arborescente est un paramètre de graphes qui quantifie à quel point un graphe est proche d'un arbre ou pas.

Remarque 3.7
Dans la Figure 3.1, la décomposition arborescente proposée est de largeur *3*, car son plus gros sac contient *4* sommets de *G*. Notons que c'est d'ailleurs le mieux que l'on puisse faire. En effet il est possible de montrer qu'il n'existe pas de

décomposition arborescente de ce graphe qui soit de largeur *1* ou *2* (ce graphe est de largeur arborescente *3*).

3.1.1.2 La longueur arborescente

Cette notion de longueur arborescente vient compléter celle de largeur arborescente qui est abondamment étudiée depuis des années.
Il apparaît donc une nouvelle famille de graphes : celle des graphes qui admettent une décomposition arborescente dont les sacs sont de diamètre borné par une constante δ . Nous appelons cette famille, la famille des graphes de longueur arborescente δ .

La longueur arborescente est définie de la façon suivante :

Définition 3.8 (Longueur arborescente) Le diamètre d'un sac *B* est la distance dans le graphe entre les deux sommets les plus éloignés du sac :

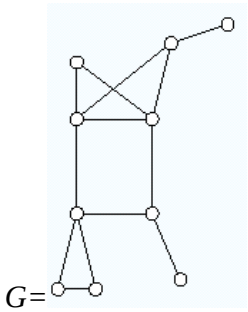
$$diam_G(B) = \max_{u,v \in B} \{dist_G(u, v)\}.$$

La longueur d'une décomposition arborescente *T* est le plus grand diamètre de ses sacs : $Longueur(T) = \max_{B \in V(T)} \{diam_G(B)\}$.

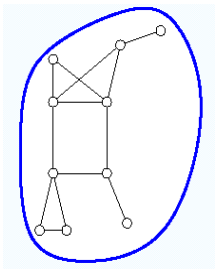
La longueur arborescente d'un graphe *G* (en anglais tree-length), notée *tl(G)*, est la plus petite des longueurs de toutes les décompositions arborescentes possibles de *G* : $tl(G) = \min_T \{longueur(T)\}$.

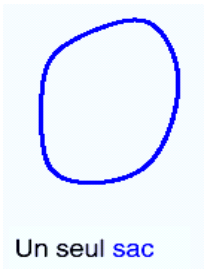
Question Comment couvrir un graphe avec des sous-graphes de faible diamètre ?.

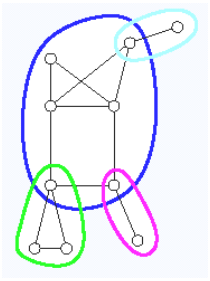
Exemple 3.9 Soit *G* le graphe suivant :



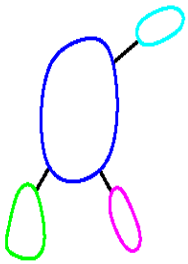
Plusieurs arbres sont possibles...

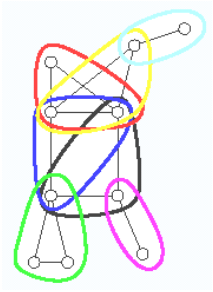






Décomposition en composantes bi-connexes





$|X| \leq 3,\ diam_G(X) \leq 2$

3.1.2 Recherche de sous-graphes couvrants approximant les distances

Un sous-graphe H couvrant un graphe G est *(a, b)*-approximant (en anglais *(a,b)*-spanner).

Définition 3.10 (Spanner)

Un sous-graphe G' couvrant G est un (a, b)-spanner si pour tout couple de sommets (u, v) on a :

$$dist_{G'}(u,v) \leq a.dist_G(u,v) + b$$

- Si a=1 alors G' est un $b-$ spanner additif.
- Si b=0 alors G' est un $a-$ spanner multiplicatif.

Objectif Minimiser le nombre d'arêtes de G'.

Théorème 3.11 ([Dou03])

Pour tout k il existe un graphe triangulé n'admettant pas d'arbre couvrant qui soit un $k-$ spanner additif ($a=1$ *et* $b=k$).

Exemple 3.12

Soit le graphe G de la figure 3.3.

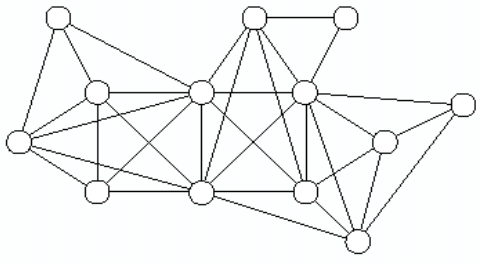


Figure 3.3 - *G* un graphe triangulé avec 13 sommets.

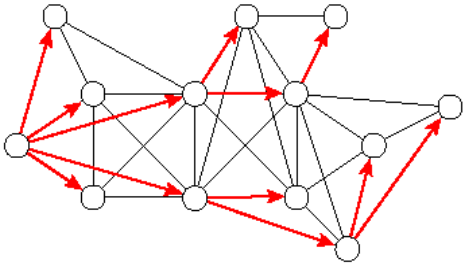


Figure 3.4 - Arbre couvrante du graphe *G*.

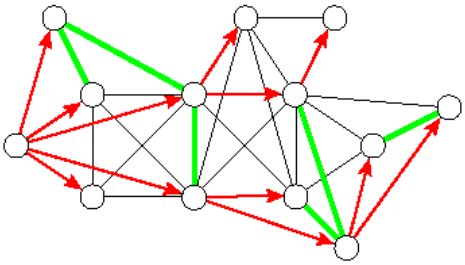


Figure 3.5 - Un sous-graphe couvrant *(1, 1)*-approximant (*1*-spanner additif).

Théorème 3.13 ([Dou03])

Tout graphe triangulé admet un *4*-spanner additif avec au plus *2n-2* arêtes.

Soit *T* une décomposition arborescente d'un graphes *G*

- La largeur de *T* est le plus petit entier *k* tel que :

$\forall X \in V(T), |X| \leq k+1$

- La **largeur** arborescente (tree-width) de *G*, notée $tw(G)$, est le minimum des largeur parmi toutes les décompositions arborescentes possibles de *G*.

- La longueur de *T* est le plus petit entier δ tel que :

$\forall X \in V(T), diam_G(X) \leq \delta$

- La **longueur** arborescente (tree-length) de *G*, notée $tl(G)$, est le minimum des longueur parmi toutes les décompositions arborescentes possibles de *G*.

Exemple 3.14

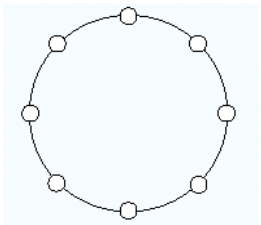


Figure 3.6 - Si *G* est un cycle alors :

$$tw(G) = 2$$

$$tl(G) = \lceil n/3 \rceil = \Omega(n)$$

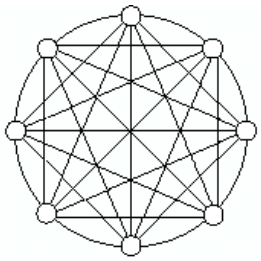


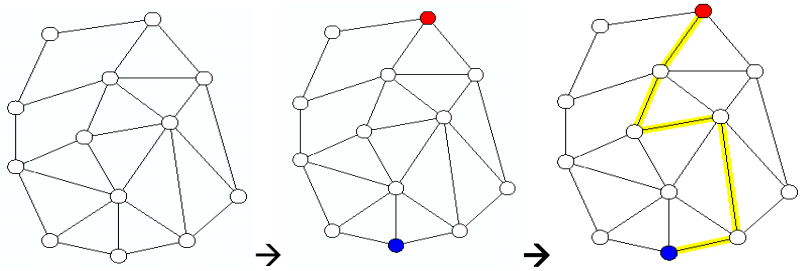
Figure 3.7 - Si G est triangulé (ici un graphe complet) alors :

$$tw(G) = \omega(G) - 1 = \Omega(n)$$

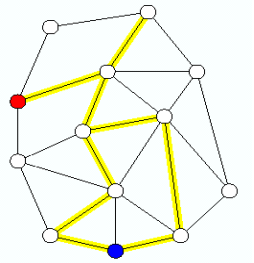
$$tl(G) = 1$$

Exemple 3.15 (Routage)

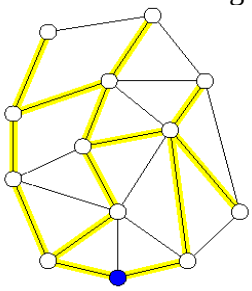
Etant donné un réseau et deux sommets. Il faut construire une route entre ces deux sommets.



Quelque soit le couple de sommets il faut déterminer la route ou bien le chemin[1] entre eux.



Une fois la **diffusion**[2] réalisée on obtient un sous-graphe couvrant.



Une fois l'échange total réalisé on obtient l'ensemble de toutes les arêtes utilisées par le schéma de routage.

[1] Dans les réseaux de télécommunication la notion de chaîne et de chemin sont confondu
[2] Exemple : Envoie des messages dans le un réseau de télécommunication

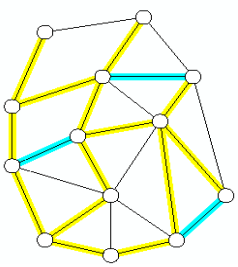


Figure 3.8 - Un sous-graphe couvrant *(1,2)*-spanner (*2*-spanner additif)

3.2.3 Schéma de routage

Performances d'un schéma de routage

- Taille des informations nécessaires à chaque sommet.
- Longueur des routes produites entre toute paire de sommets.

Théorème 3.16 ([Dou03])

Tout graphe de longueur arborescente δ admet un schéma de routage tel que la longueur de la route entre toute paire de sommet ne dépasse jamais leur distance plus 2δ.

Théorème 3.17 ([Dou03])

Tout graphe de longueur arborescente δ admet un $2\delta-$spanner additif avec $O(\delta.n\log n)$ arêtes.

□ Voir thèse de Yon Dourisboure [Dou03].

■

Définition 3.18 (Graphe *k*-chordal)

Un graphe est *k*-chordal si tous ses cycles induits sont de longueur au plus *k.*

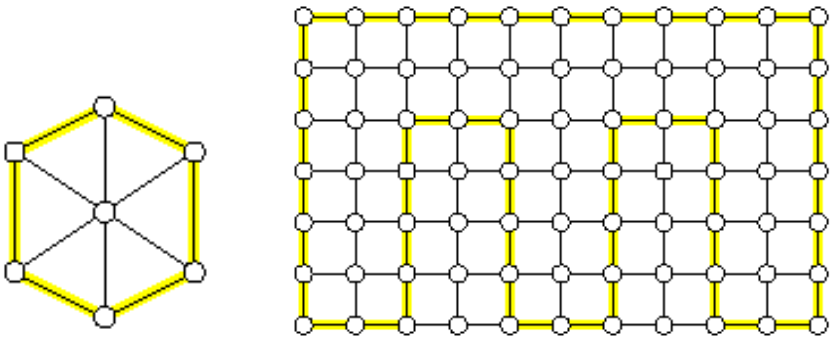
La chordalité d'un graphe *G* est le plus petit entier *k* tel que *G* est *k*-chordal.

Théorème 3.19 ([Dou03])

Tout graphe de chordalité *k* admet un *(k+1)*-spanner additif avec *2n-2* arêtes.

Théorème 3.20 ([Ber98])

Tout graphe de chordalité *k* est de longueur arborescente au plus $\left\lfloor \frac{k}{2} \right\rfloor$.



k=6 *k*=48

Figure 3.9-Exemple de chordalité

Cependant la chordalité peut être largement supérieure à la longueur arborescente.

Question
Est-il possible, pour tout graphe *G*, d'obtenir un $O(tl(G))$-spanner additif avec un nombre linéaire d'arêtes ?.

Lemme 3.21 ([Dou03])
Soit *u* et *v* deux sommets d'une même partition[3] on a : $dist_G(u, v) \leq 3.tl(G)$

Théorème 3.22 ([Dou03])
Tout graphe *G* à *n* sommets et de longueur arborescente δ admet un 6δ-spanner additif avec $O(n.\delta)$ arêtes.

Conclusion
Tout graphe de longueur arborescente δ admet :

- Un $2\delta-$spanner additif avec $O(n.n\log n)$ arêtes.
- Un 6δ-spanner additif avec $O(\delta.n)$ arêtes.

3.2 Décomposition en branches

Dans une optique « diviser pour régner », seules les opérations de découpe et de fusion devraient être coûteuses. Obtenir une solution sur un cas de base devrait se faire rapidement sinon ce cas de base devrait pouvoir se re découper. De ce point de vue, les décompositions arborescentes ne sont pas optimales. En effet, la taille du plus grand séparateur minimal utilisé pour diviser le graphe dans une telle décomposition peut être arbitrairement plus petite que la largeur de cette décomposition. De ce point de vue, les décompositions en branches sont plus satisfaisantes.

Robertson et Seymour ont introduit un nouveau paramètre dans le but de construire une obstruction à la largeur arborescente. Ce paramètre est appelé « largeur de branche » noté $bw(G)$.

Ils ont montré que pour tout graphe *G*, $bw(G) \leq tw(G) \leq 3/\ 2bw(G)$. Seymour et Thomas ont montré que le calcul de la largeur de branche d'un graphe planaire est polynomial alors que le problème de la largeur arborescente d'un graphe planaire est encore ouvert. Malgré ce résultat, la largeur de branche n'a presque pas été étudiée.

Un graphe de corde est le modèle d'intersection des cordes d'un disque. Kloks a montré comment calculer la largeur arborescente des graphes de cordes.

Objectif Certains problèmes NP-complets sont linéaires pour une classe de graphes de largeur de branches bornée.

[3] Le même sac d'une décomposition.

Définition 3.23 (Décomposition en branche)

Soit *G* un graphe. Une décomposition en branche T_r de *G* est un arbre *T* muni d'un fonction *r* d'étiquetage des feuilles de *T* de telle sorte que :

1. Les nœuds internes de *T* soient de degré trois
2. La fonction *r* soit une bijection de l'ensemble des feuilles de *T* vers les arêtes de *G*.

A chaque arête *e* de *T* correspond une séparation $\mathcal{E}=\{E_1,E_2\}$ où E_1 et E_2 sont les ensembles d'arêtes étiquetant les feuilles des deux composantes connexes de $T\backslash\{e\}$. La frontière de l'arête *e* est celle de la séparation $\mathcal{E}$.

3.2.1 L'invariant lié à la décomposition en branches

3.2.1.1 La Largeur de branche

Définition 3.24 (Largeur de branche)

La largeur de la décomposition en branche notée *bw(T)* est la plus grosse taille d'une frontière de ses arêtes et la largeur de branches de *G* notée *bw(G)*[4] est la plus petite largeur d'une décomposition de *G*.

La Figure 3.10 donne un graphe et une de ses décompositions en branches. Pour « découper » le graphe *G*, les décompositions en branches utilisent une partition $\{E_1, E_2\}$ de l'ensemble de ses arêtes. Nous obtenons alors deux graphes $G(E_1)$ et $G(E_2)$. Mais, comme ce mode de « découpage » permet de diviser des cliques, nous ne pouvons plus compléter la frontière entre ces deux graphes en une clique pour respecter la structure du graphe initial. Nous sommes donc amenés à considérer les deux sous-graphes contractés $G_{/E_2}$ et $G_{/E_1}$.

Définition 3.25 (graphe contracté)

Soient *G = (V,E)* un graphe et *E'* un ensemble d'arêtes de *G*. Le graphe *G[E\E']* auquel nous avons rajouté l'arête $\partial(E')$ est noté $G_{/E'}$. C'est le graphe *G* dans lequel les arêtes de *E'* sont contractées. Autant les décompositions arborescentes peuvent être présentées en se restreignant aux graphes, autant le bon cadre pour les décompositions en branches est celui des graphes.

L'algorithme decomposition_branches (algo. 3.2) construit un tel schéma de décomposition. Il est aisé de se convaincre que ces schémas sont bien des décompositions en branches et que toutes les décompositions en branches d'un graphe peuvent être obtenues ainsi. La Figure 3.10 illustre une de ses exécutions. Les arêtes de la décomposition correspondent aux séparations choisies pour décomposer *G*.

[4] Le terme anglo-saxon est *branchwidth* ce qui explique la notation *bw(G)*.

Algorithme 3.2 decomposition_branches

entrée :

$G = (V,E)$ //un graphe connexe

sortie :

T : //une décomposition en branches

début

si le graphe à une seul arête **alors**

T l'arbre réduit à un nœud contenant l'arête de G.

si le graphe à deux arêtes **alors**

T l'arbre réduit à une arête dont les feuilles contiennent les arêtes de G.

si le graphe à trois arêtes **alors**

T l'étoile à trois branches dont les feuilles contiennent les arêtes de G.

si le graphe à plus de trois arêtes **alors**

Soit $\{E_1, E_2\}$ une séparation de G avec $|E_1| \geq 2$ et $|E_2| \geq 2$

$S \leftarrow \partial(E_1)$.

$V_1 \leftarrow \{v \in V \| \ v \text{ est incident à au moins une arête de } E_1\}$.

$T_1 \leftarrow$ decomposition_branches $(G_1 = (V_1, E_1 \cup \{S\}))$.

$V_2 \leftarrow \{v \in V \| \ v \text{ est incident à au moins une arête de } E_2\}$.

$T_2 \leftarrow$ decomposition_branches $(G_2 = (V_2, E_2 \cup \{S\}))$.

$T \leftarrow T_1$ *et* T_2 recollés sur la feuille contenant S.

rendre *(T)*

fin

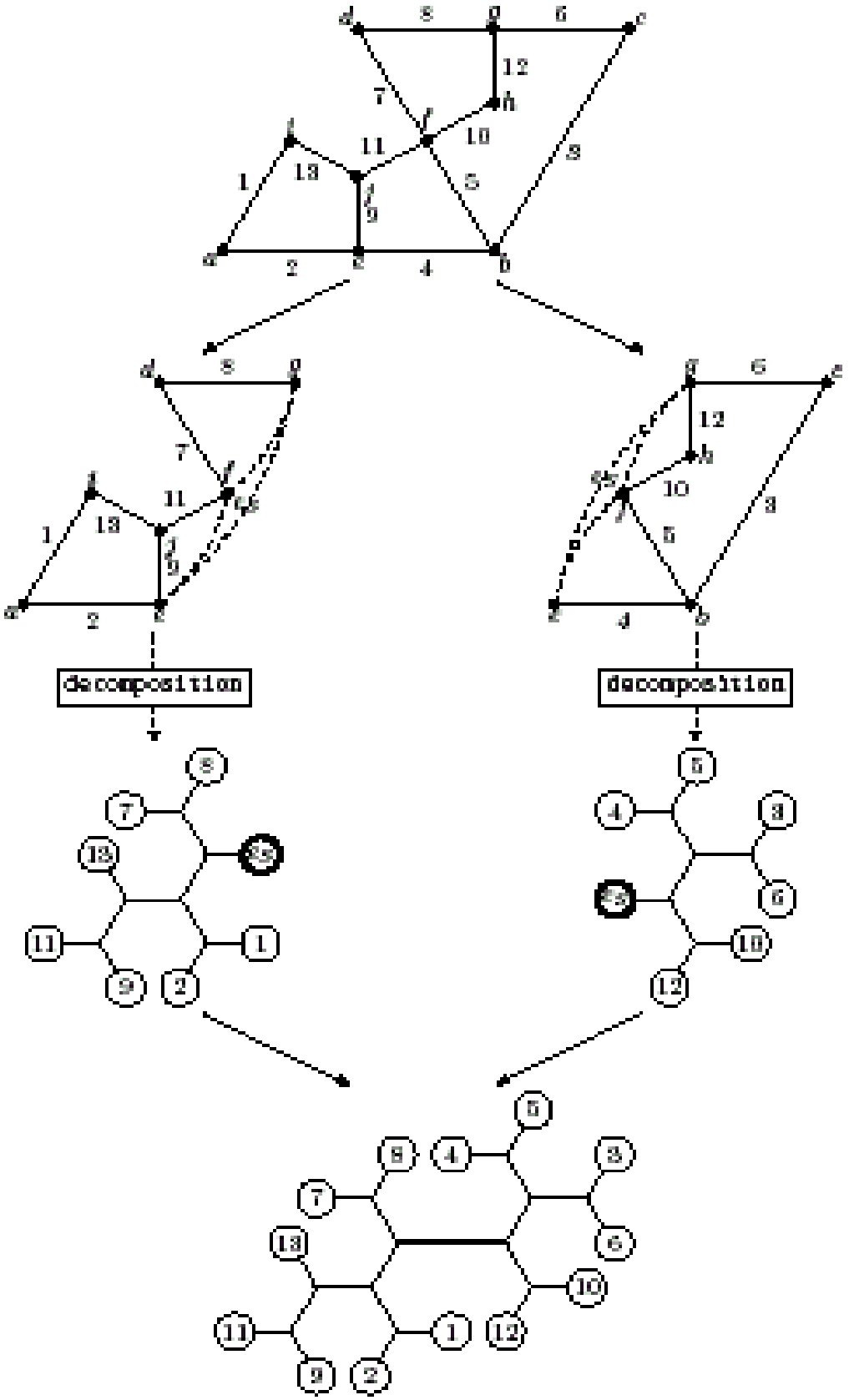


Figure 3.10 - Une exécution de l'algorithme decomposition_branches

Certaines décompositions en branches peuvent donc être vues comme l'application d'une technique « diviser pour régner » sur des graphes. L'efficacité d'un tel schéma dépend de la taille d'une plus grosse frontière utilisée lors de la décomposition. La largeur de branches permet bien de juger de la qualité d'une décomposition en branches (voir Figure 3.11).

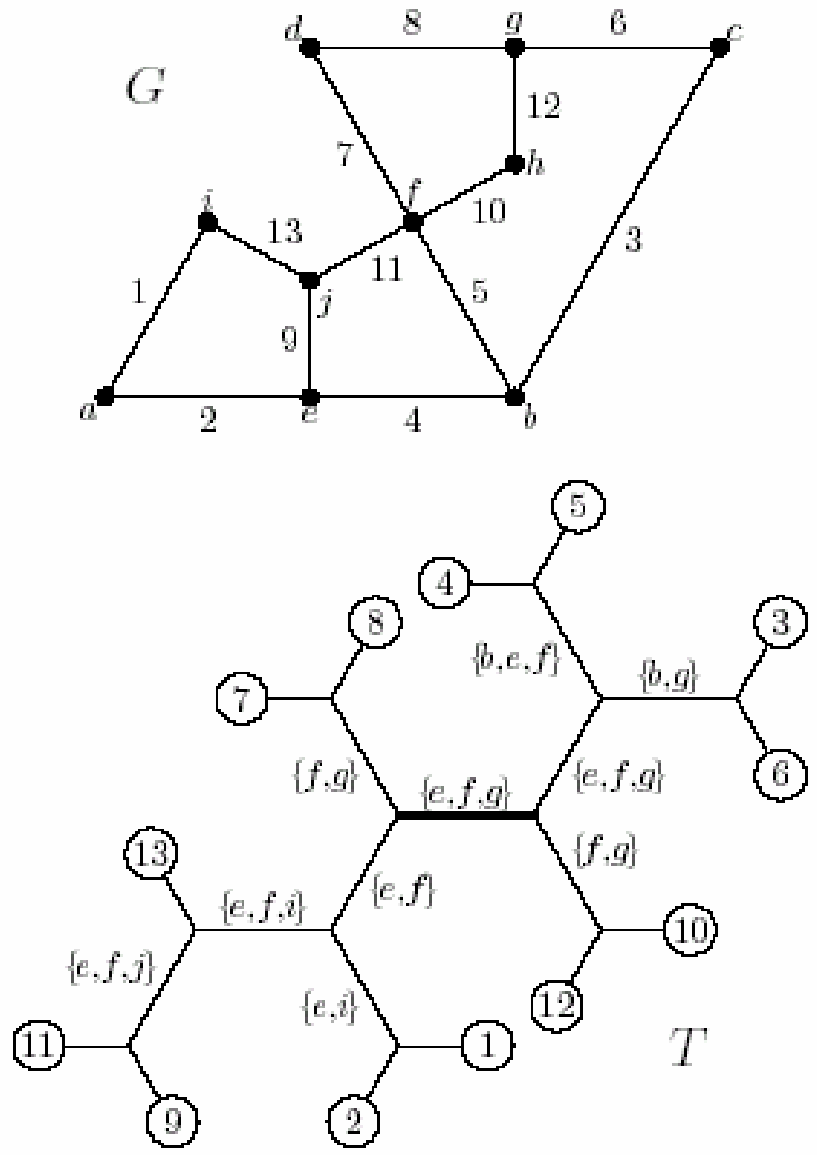


Figure 3.11 –Décomposition en Branches

L'arbre *T* est une décomposition en branches du graphe *G*.

L'arête de *T* représentée en gras correspond à la séparation $\{E_1, E_2\}$ = *({1, 2, 7, 8, 9, 11, 13}, {3, 4, 5, 6, 10, 12}).*

Sa frontière est *{e, f, g}* et sa taille est donc de trois. Les autres ensembles donnés correspondent aux frontières des arêtes à côté des quelles ils sont représentés.

La largeur de *T* est trois (*bw(T)=3*).

3.4 Quelques liens entre les deux décompositions

Nous avons introduit les décompositions arborescentes et les décompositions en branches en suivant une démarche similaire. Nous montrons maintenant certaines propriétés qui justifient que cette similarité ne s'arrête pas là. Tout d'abord, ces deux paramètres sont compatibles avec la relation de minoration.

Propriété 3.26 [RS84] Soient H un sur-graphe[5] d'un graphe G. Nous avons : $tw(G) \leq tw(H)$.

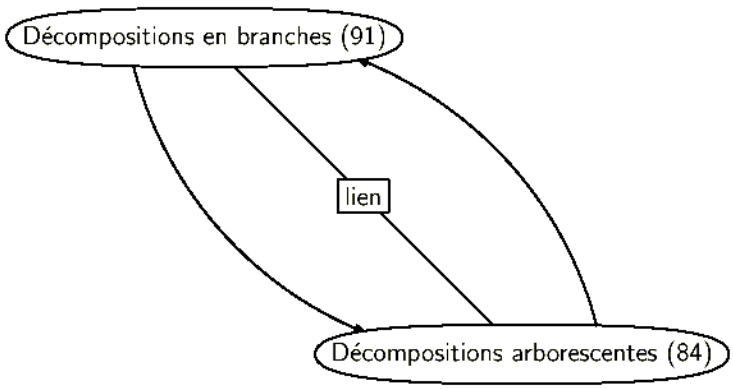
□ Soit T_{θ_H} une décomposition arborescente de H dont la largeur est $tw(H)$. Si nous ne gardons dans les étiquettes de T_{θ_H} que les sommets de G, nous obtenons une décomposition arborescente T_{θ_G} dont la largeur est inférieure ou égale à celle de H. Par conséquent,

$$tw(G) \leq tw(T_{\theta_G}) \leq bw(T_{\theta_H}) = bw(H).$$

■

Propriété 3.27 [RS91] Soit H un sur-graphe d'un graphe G. Nous avons : $bw(G) \leq bw(H)$.

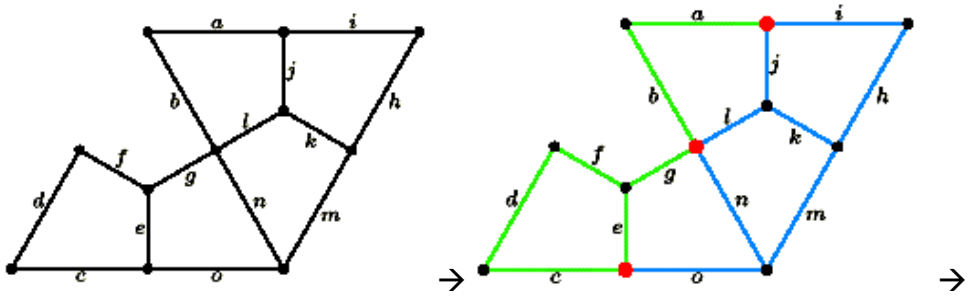
Décompositions algorithmiques des graphes



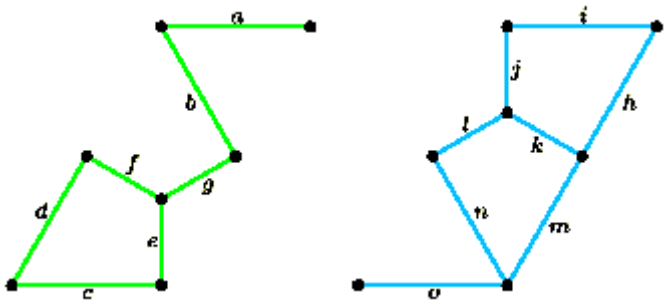
Arbres de matriochkas

a. Première méthode : Décomposition en branche

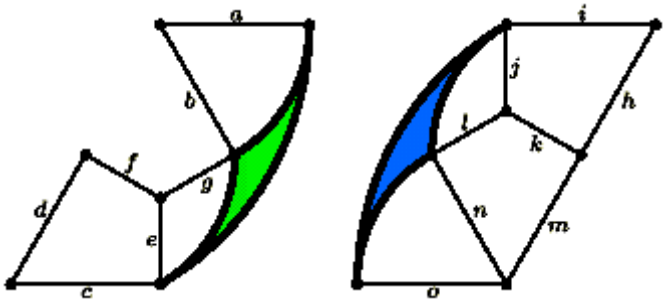
Soit $G = (V, E)$ un graphe.



[5] $H(V', E')$ est un sur-graphe d'un graphe $G(V, E)$ est un graphe dont il vérifie les deux conditions : $V' = V$ *et* $E' \supset E$.

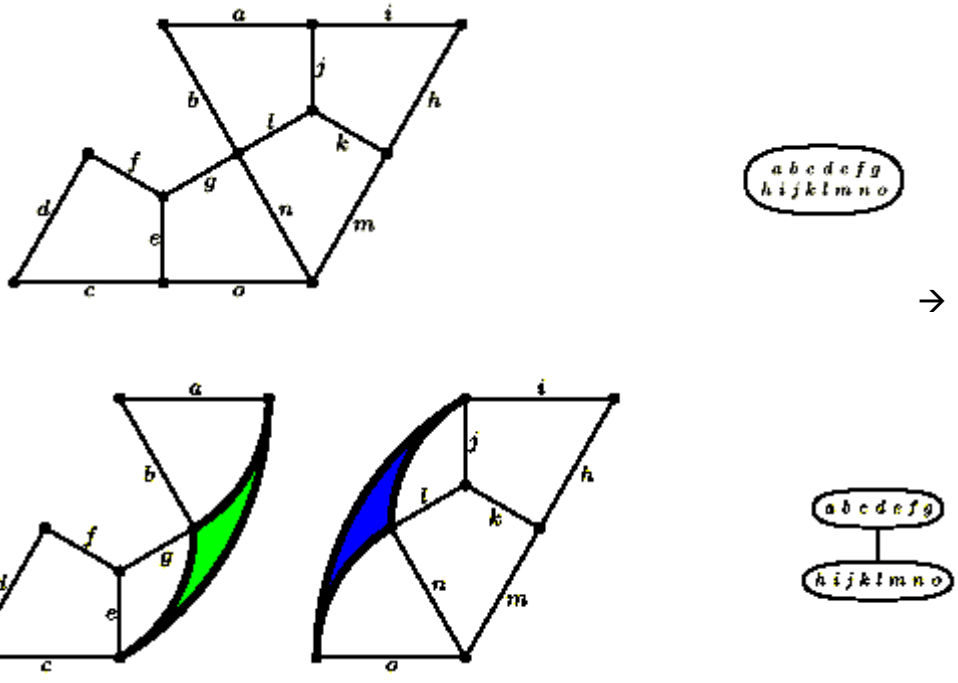


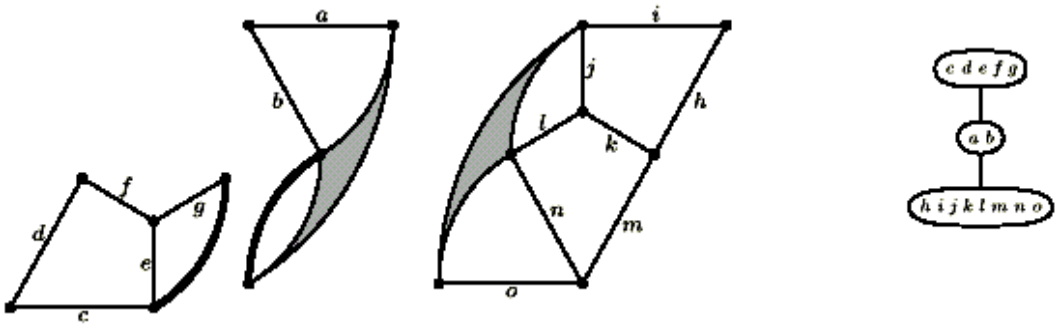
Pour tenir compte du « morceau retiré », on ajoute des arêtes.



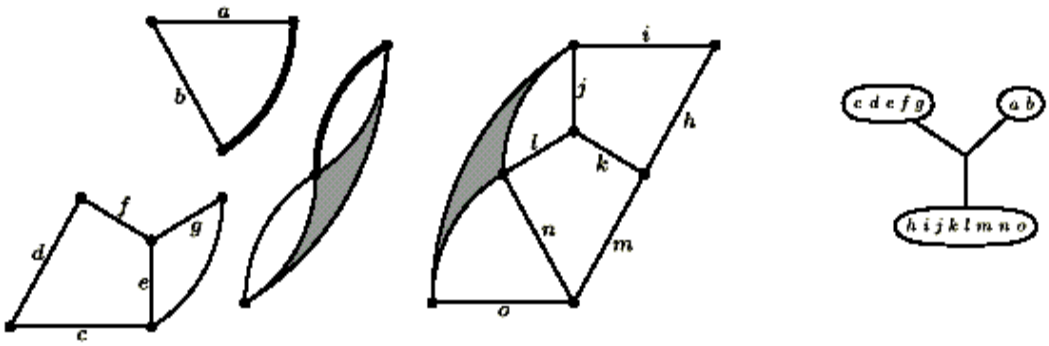
Les arêtes représentent la frontière de $\mathcal{E}$.

On peut représenter les découpages successifs par un arbre *T* qu'on affine.





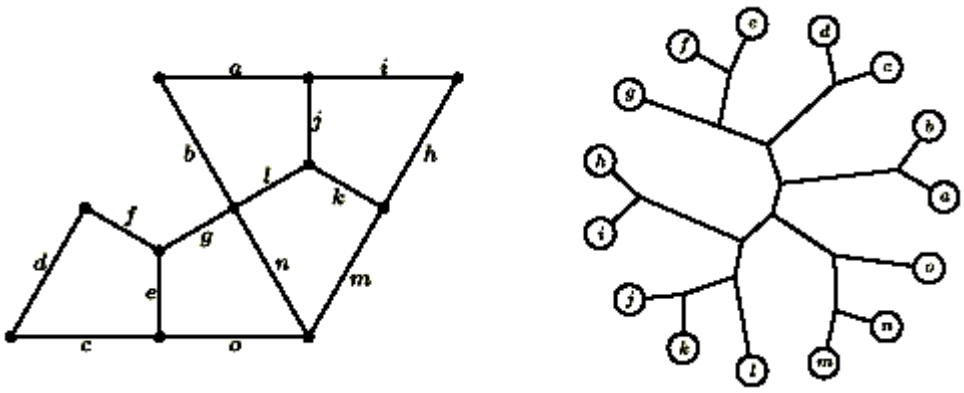
→



Chaque nœud de *T* correspond à un « morceau » du graphe.
Les arbres étiquetés obtenus sont des **arbres de matriochka**.

Définition 3.28 (Décomposition en branches)
Une **décomposition en branches** d'un graphe *G* est un arbre de matriochka *T* qu'on ne peut pas affiner.

- Ses feuilles sont étiquetées par les arêtes de *G*.
- Ses nœuds internes sont de degré trois.



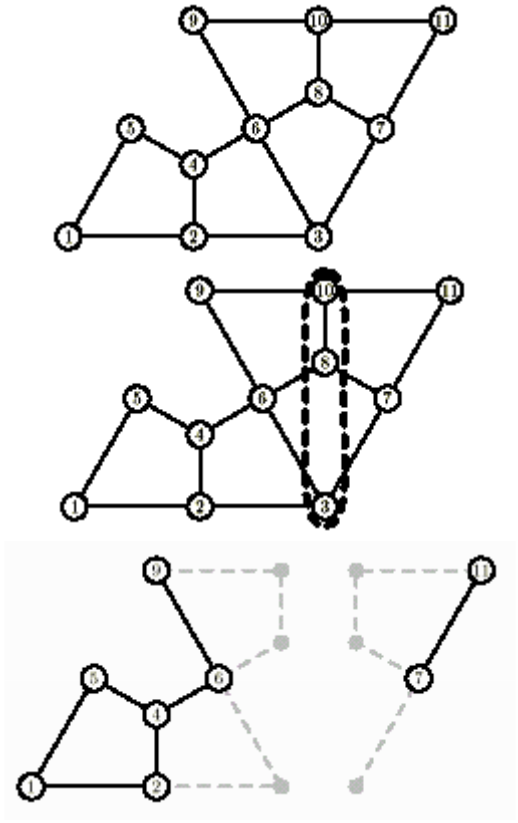
Remarque
La taille des frontières utilisées conditionne l'efficacité de ce schéma diviser pour régner.

Définition 3.29 (Largeur de branches)
La **largeur** d'une décomposition en branches est la plus grande des tailles de ses frontières.

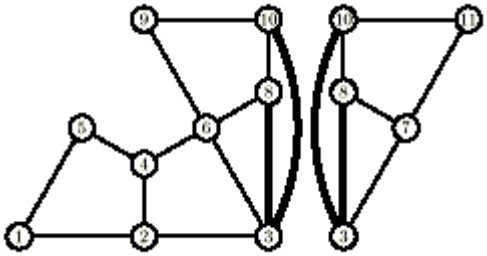
La **largeur de branches** d'un graphe est la plus petite largeur d'une de ses décompositions en branches.

b. Seconde méthode : Décompositions arborescentes :

On peut utiliser un séparateur pour découper un graphe *G*.

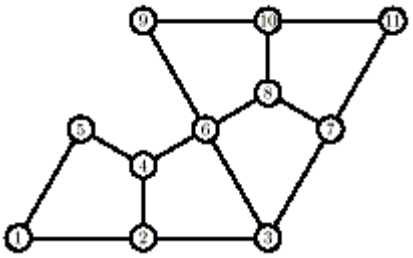


On rajoute le séparateur aux « morceaux ».

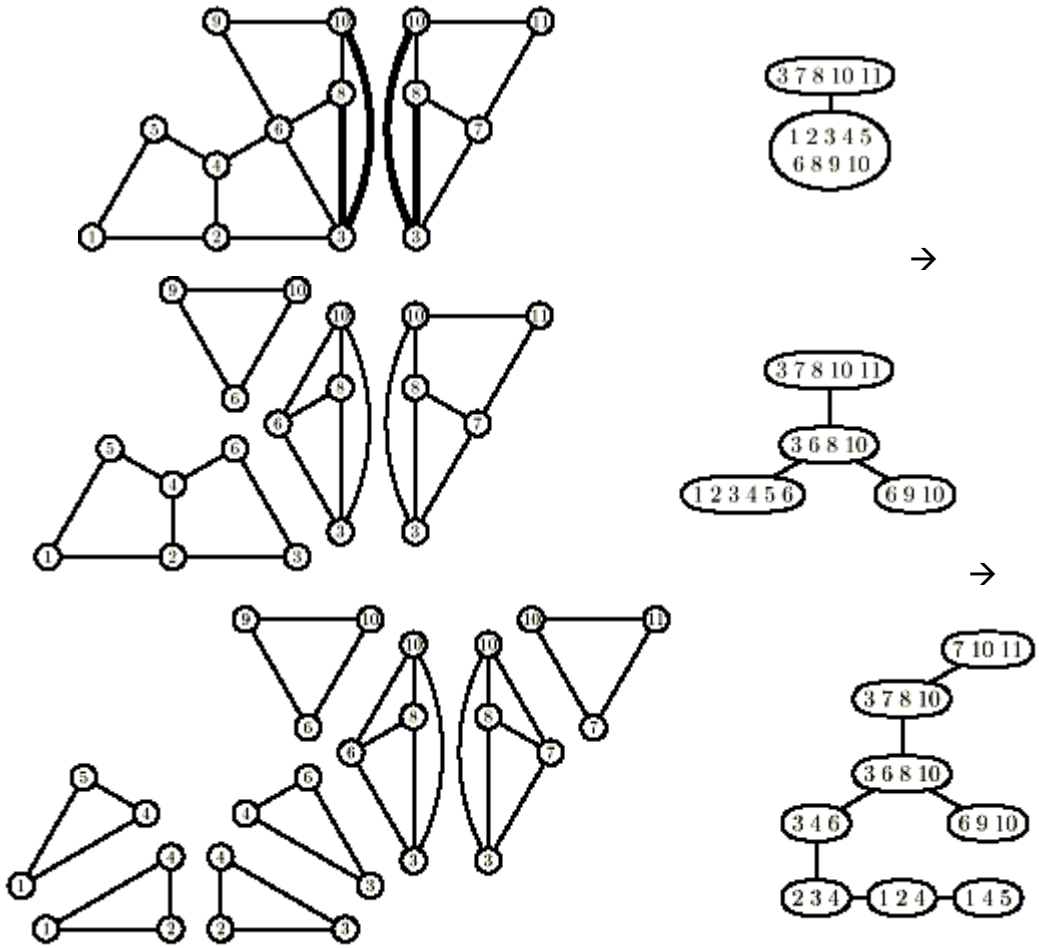


De plus, on le complète en une clique.

On peut représenter les découpages par un arbre *T*.



→



Chaque nœud de *T* correspond à un « morceau » du graphe.

Définition 3.30 (Largeur arborescente)

Une **décomposition arborescente** d'un graphe $G = (V, E)$ est un arbre T dont chaque nœud u est étiqueté par un ensemble Vu de sommets de G de telle sorte que :

- $\forall x \in V$, les nœuds dont l'étiquette contient x conduisent un sous-arbre non vide T_x de T ;
- $\forall (x, y) \in E, T_x \cap T_y \neq \phi$.

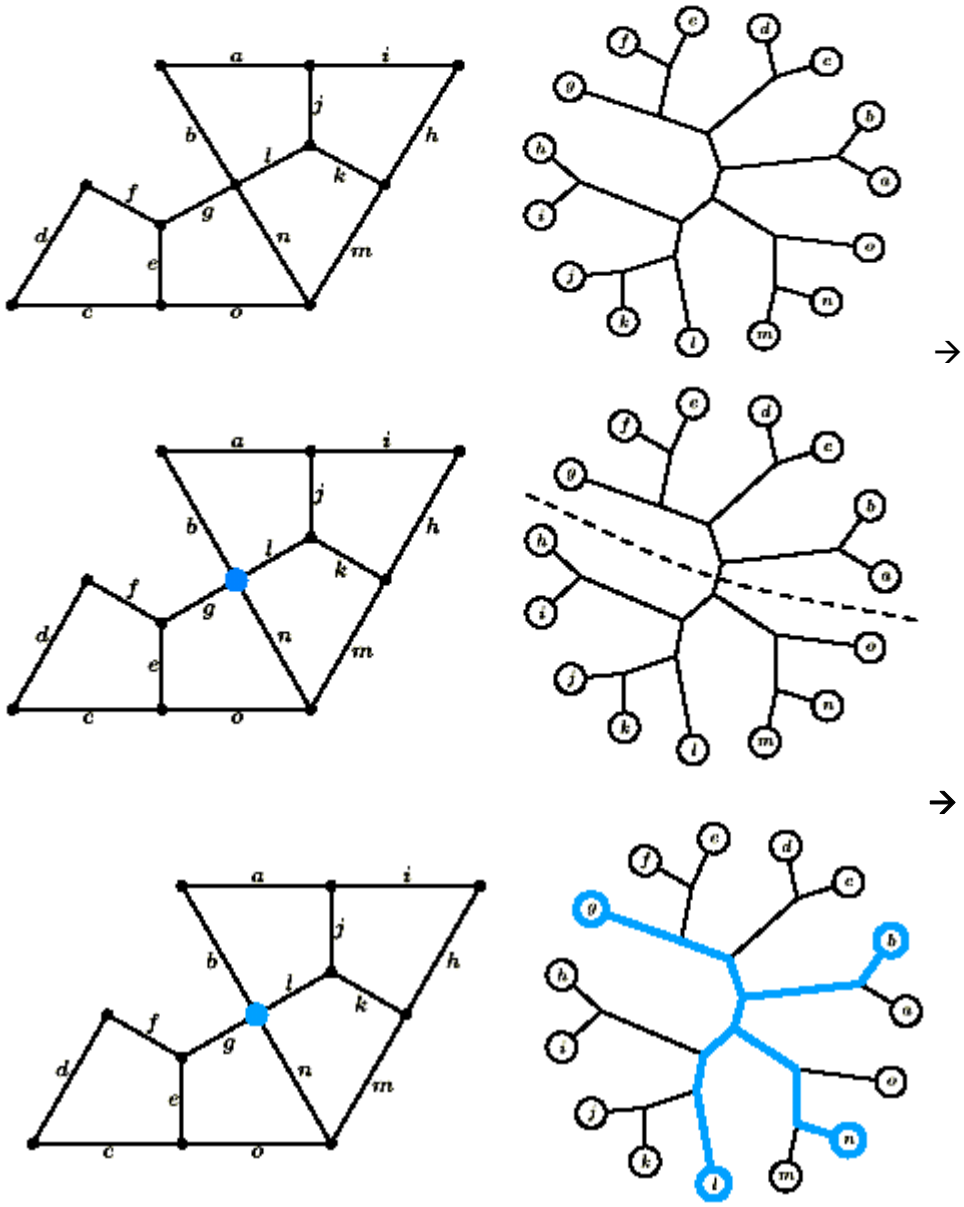
La **largeur** d'une décomposition arborescente est $\max\left(|V_u| - 1\right)$.

La **largeur arborescente** d'un graphe est la plus petite largeur d'une de ses décompositions arborescentes.

c. Lien entre ces deux types de décompositions

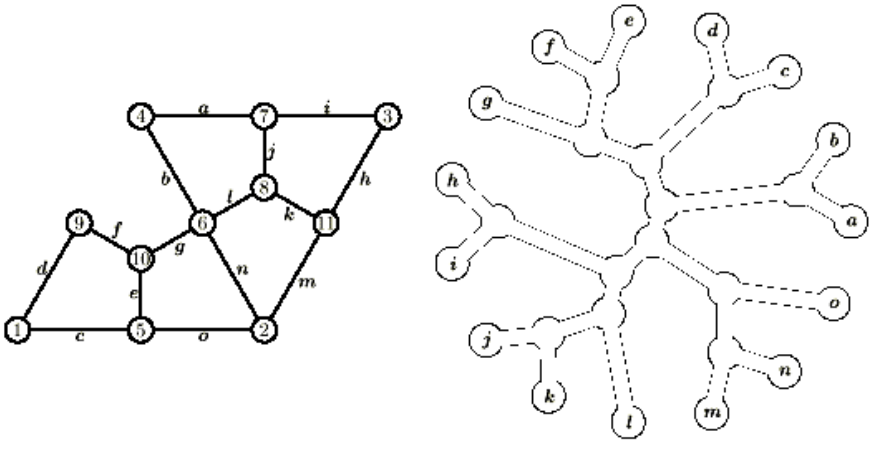
Question

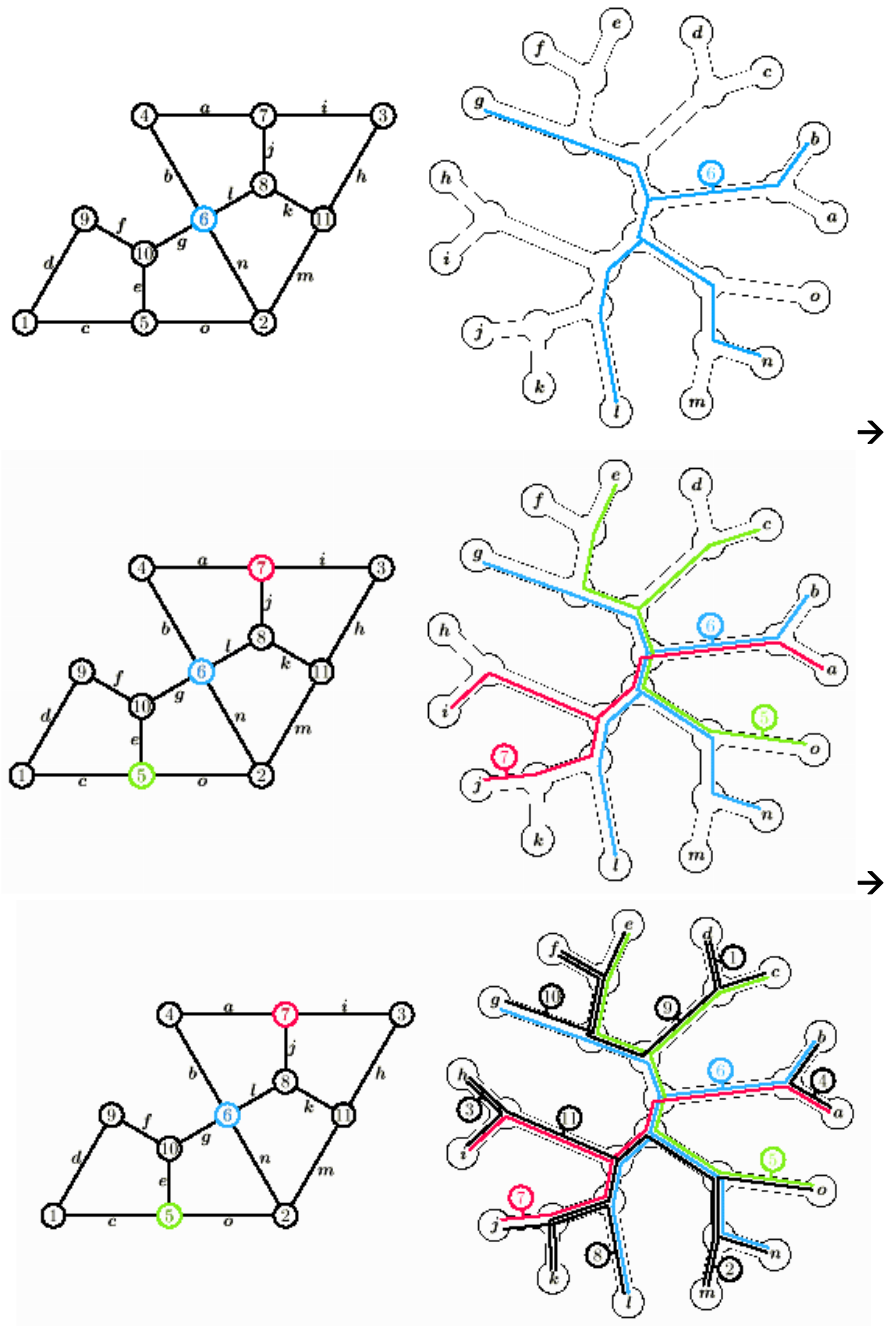
Comment reconnaître les frontières contenant un sommet donné ?



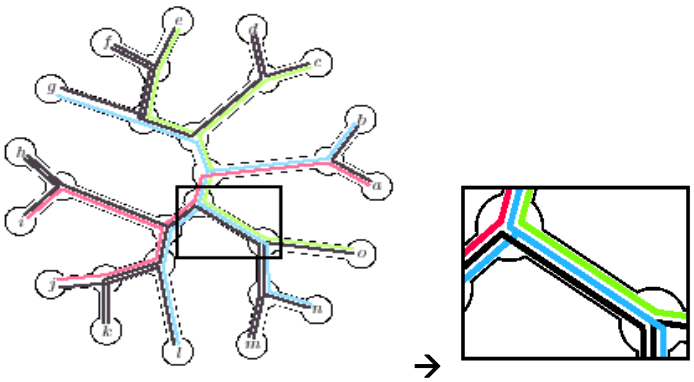
Elles induisent le sous-arbre dont les feuilles sont étiquetées par les arêtes incidentes au sommet donné.

On obtient ainsi une famille de sous-arbres d'un arbre.

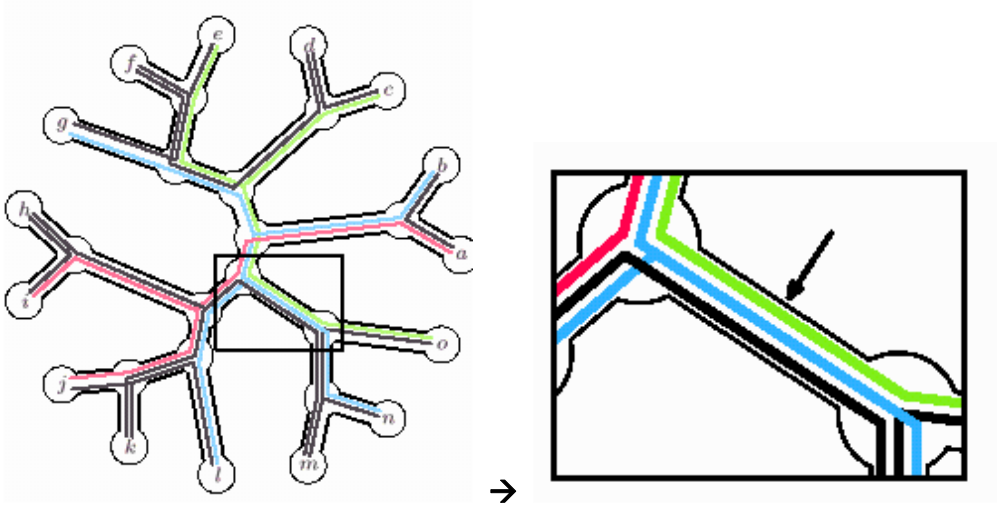




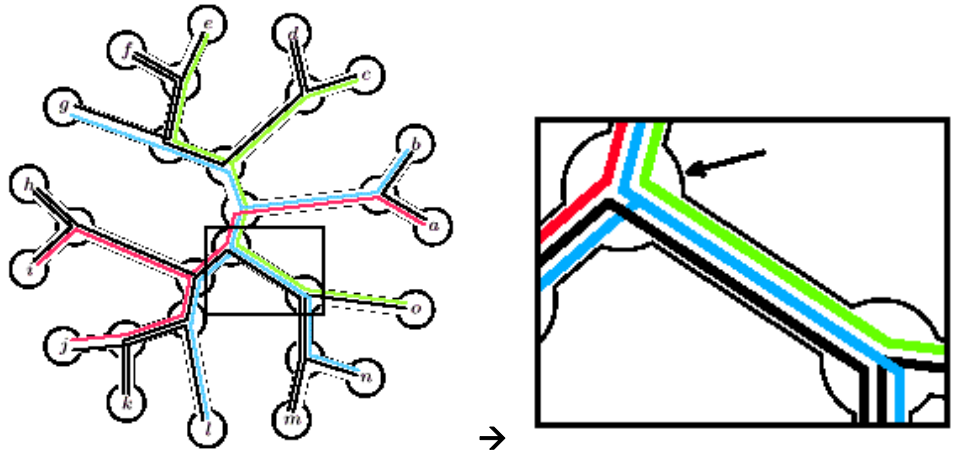
On obtient ainsi une famille de sous-arbres d'un arbre et donc une décomposition arborescente.



Si on minimise la taille des arêtes, on obtient la largeur de branches.

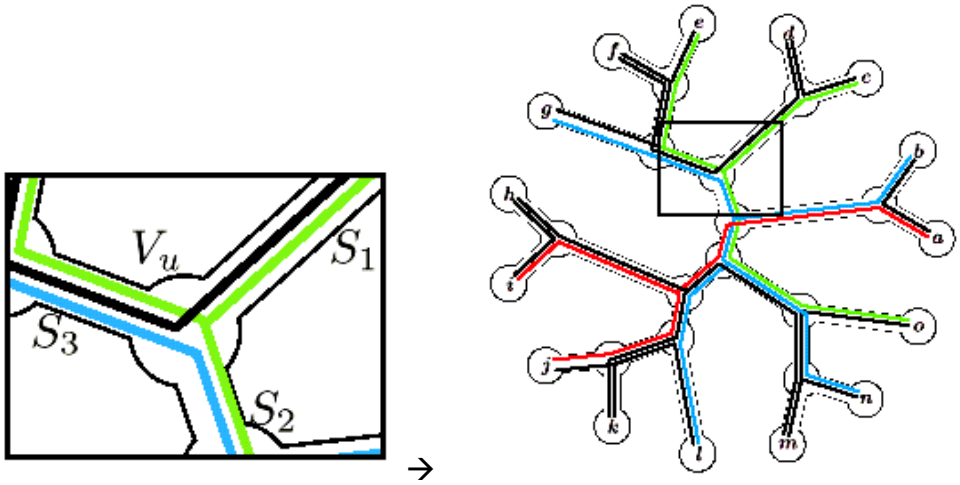


Si on minimise la taille des nœuds, on obtient la largeur arborescente.



On a $|S_1| + |S_2| + |S_3| \geq 2|V_n|$

Si $bw(T) = bw(G)$, $3bw(G) \geq 2|V_u|$.



Le théorème suivant est dû à Robertson et Seymour en 1991 :

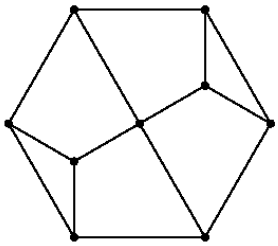
Théorème 3.31 ([RS91])

$$bw(G) \leq tw(G) + 1 \leq \frac{3}{2} bw(G)$$

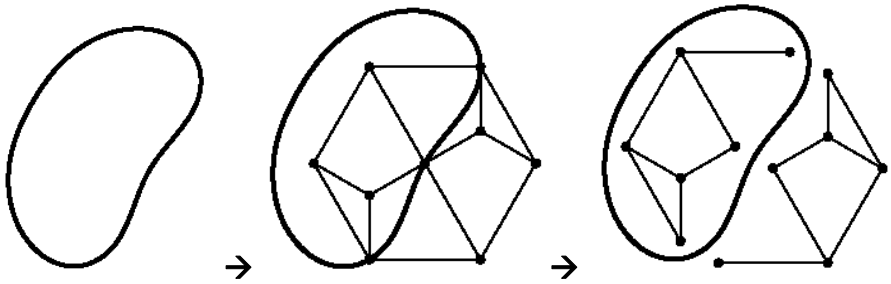
Applications aux graphes planaires

Définition 3.32 (3-connexe)

Un graphe est **3-connexe** s'il reste connexe quand on lui retire un ou deux sommets.

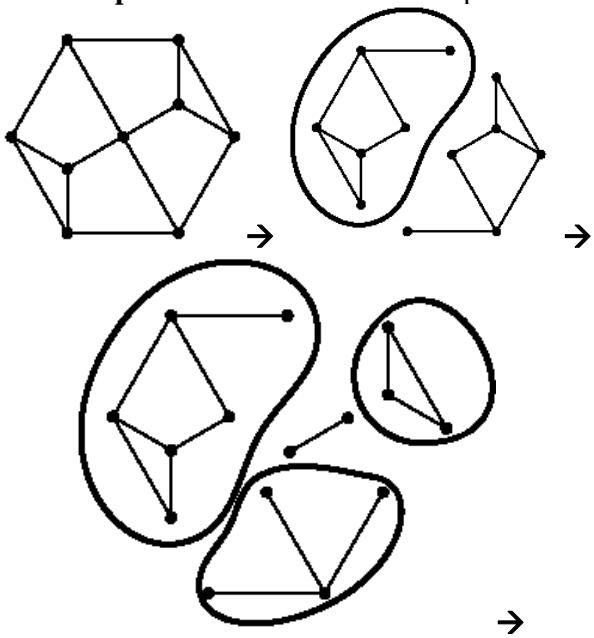


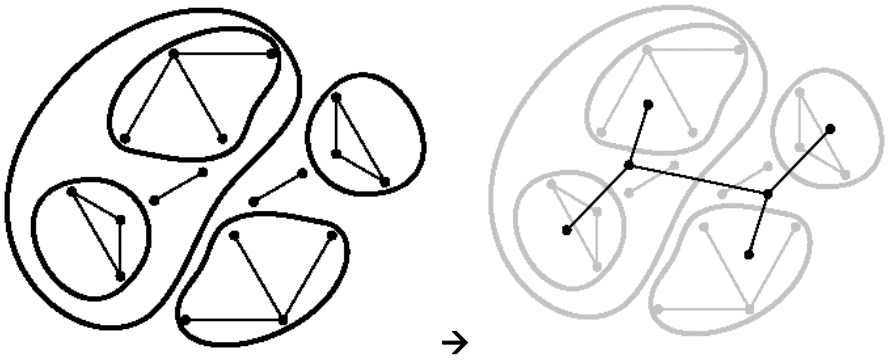
Décompositions et courbes de Jordan



Propriété 3.33

Une courbe de Jordan **compatible** avec *G* induit une séparation de *G*.



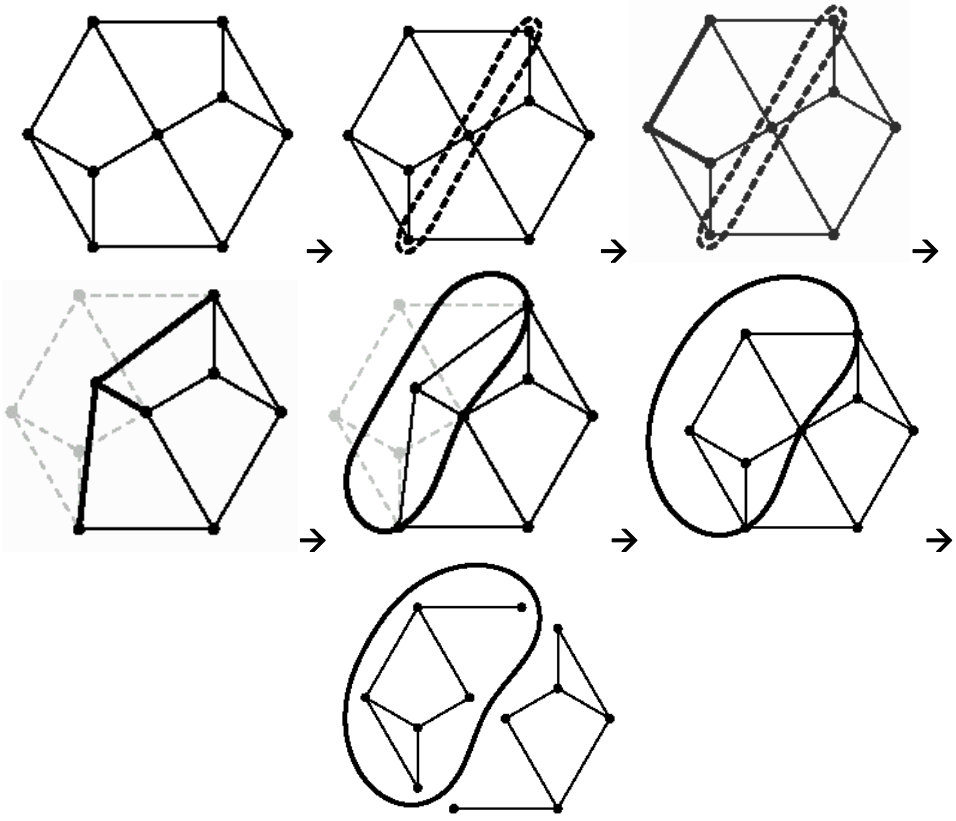


Théorème 3.34 ([Maz04])

Une famille de courbes de Jordan parallèles de *G* définit un arbre de matriochka. Un tel arbre est planaire.

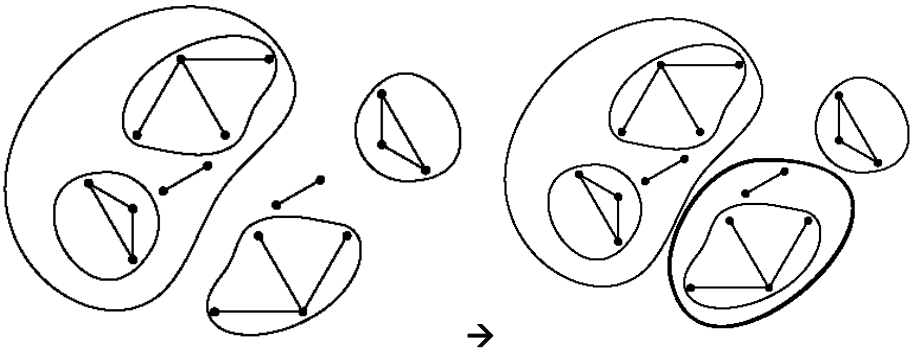
Propriété 3.35 ([Maz04])

Tout séparateur minimal est réalisé par une courbe de Jordan.



Théorème 3.36

Il existe un arbre de matriochka planaire de largeur arborescente minimale.

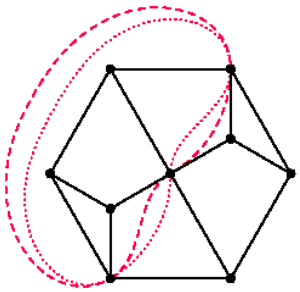


Théorème 3.37

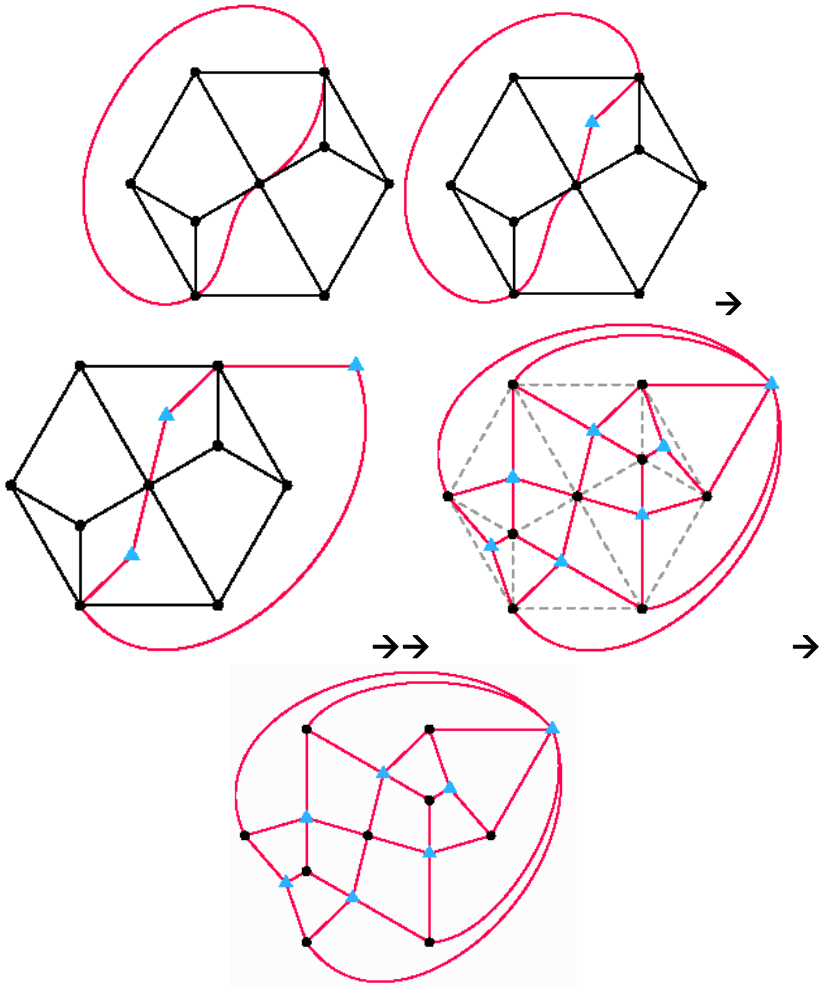
Si on affine un arbre de matriochka, sa largeur arborescente n'augmente pas.

Graphe intermédiaire

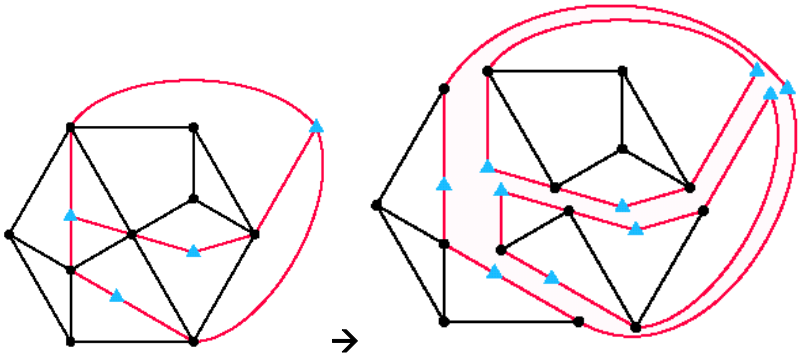
Deux courbes de Jordan distinctes peuvent définir la même séparation.



Pour éviter cette ambiguïté, on fait des choix.



On définit ainsi le **graphe intermédiaire** G_1 de G.



Propriété 3.38 [Maz04]
Une « région » délimitée par une famille maximale de courbes de Jordan de G_1 deux-à-deux parallèles est :

- Soit une face de G_1 ;
- Soit une $\Theta-$ structure.

Corollaire 3.39 [Maz04]
Un arbre de matriochka induit par une famille maximale de courbes de Jordan de G_1 deux-à-deux parallèles est une décomposition en branches.

Théorème 3.40 [ST94]
Il existe une décomposition en branches planaire de largeur arborescente minimale.

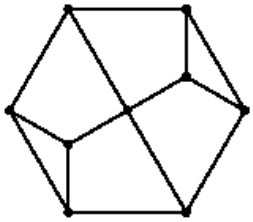
Le théorème suivant est dû à Seymour et Thomas en 1994 :

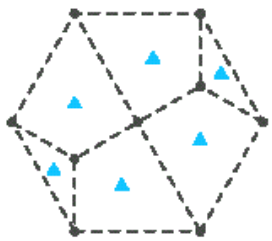
Théorème 3.41 [ST94]
Il existe une décomposition en branches planaire de largeur de branches minimale.

Théorème de dualité

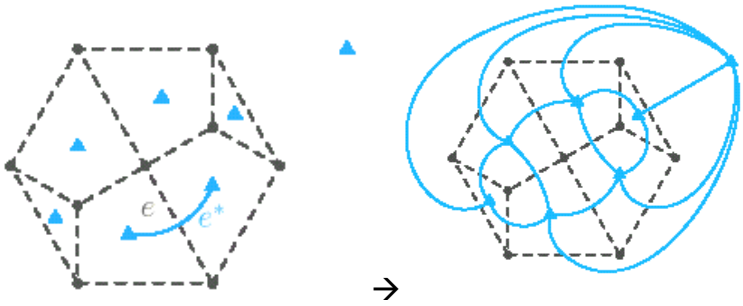
Définition 3.42 (Graphe dual)

Le **graphe dual** G^* d'un graphe planaire G est obtenu en plaçant un sommet par face de G.
Soit le graphe Suivant :

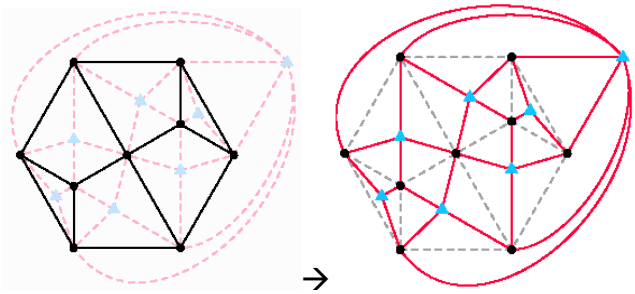




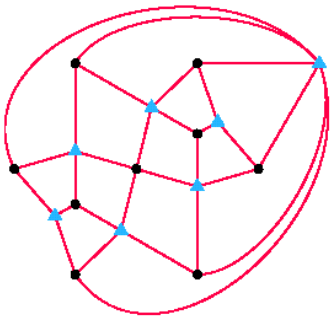
- Une arête e^* pour chaque arête e de G reliant les faces incidentes à e.



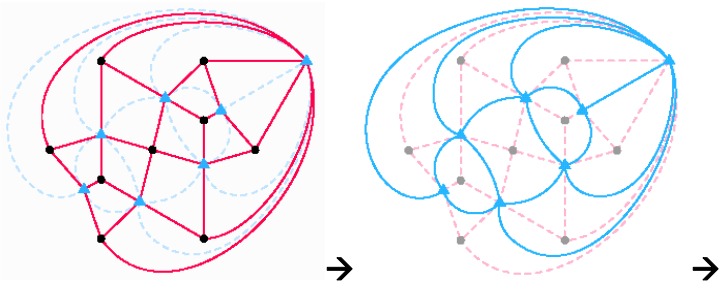
Chaque face de G_1 correspond `a une arête de G.

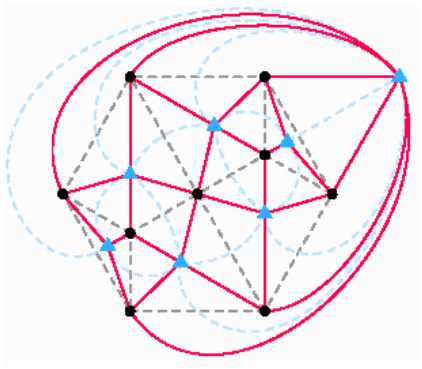


On peut reconstruire G à partir de G_1.



On peut aussi reconstruire G^* :

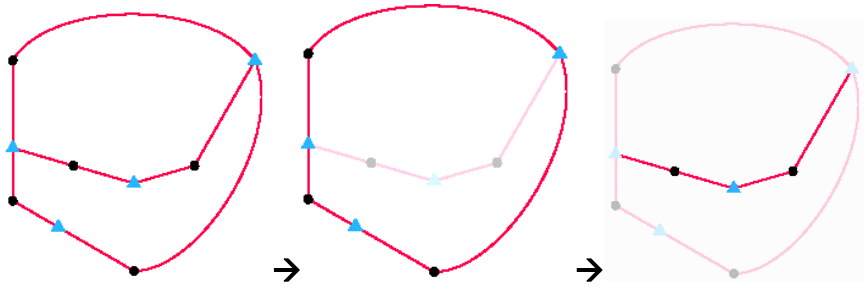




Théorème 3.43 [Maz04]

Un graphe planaire et son dual ont le même graphe intermédiaire.

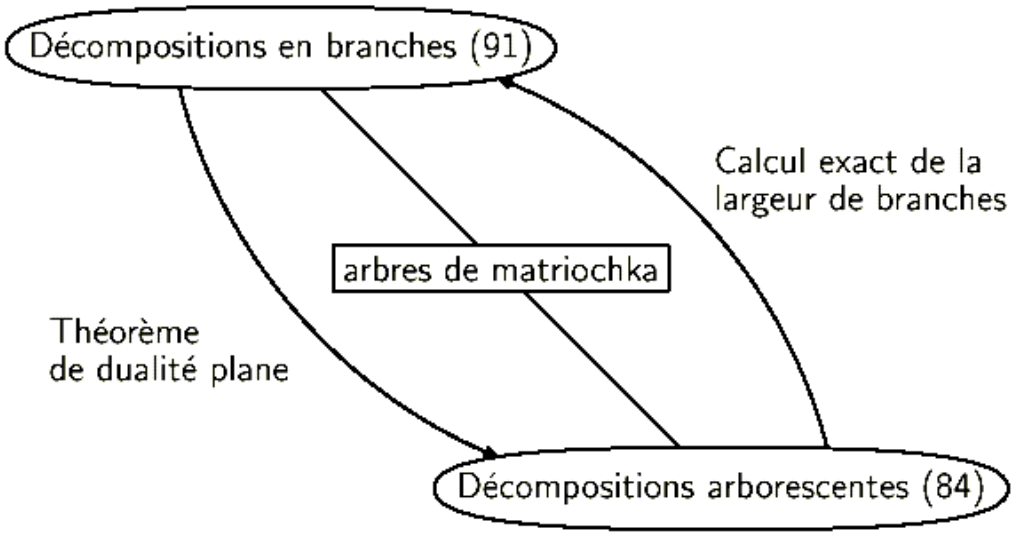
On peut donc transférer une décomposition en branches planaire de G à G^*.



Théorème 3.44 [Maz04]

Pour tout graphe planaire, $tw(G) - 1 \leq tw(G^*) \leq tw(G) + 1$.

3.4 Conclusion la relation entre la décomposition en branche et la décomposition arborescente est donnée par le schéma suivant :



4. Triangulations minimales

La notion de décomposition arborescente est très liée à celle de triangulation. Les définitions 3.6 et 3.24 énoncent que la largeur de branches et la largeur arborescente d'un graphe sont égales aux plus petites largeurs correspondantes d'une triangulation de graphe. Les graphes triangulés étant structurellement simples, nous pouvons penser que le calcul de leur largeur de branches est plus aisé que pour un graphe quelconque ; c'est en particulier le cas pour la largeur arborescente. Il peut donc être intéressant de chercher à calculer la largeur arborescente ou la largeur de branches de triangulations de G pour en déduire celle de G. Cependant, en utilisant cette approche, nous perdons d'un côté ce que nous gagnons de l'autre : au lieu de calculer un paramètre pour un seul graphe, nous devons le calculer pour un grand nombre de triangulations.

Nous cherchons donc à restreindre la classe des triangulations à considérer.

Dans ce chapitre nous donnerons une nouvelle caractérisation des triangulations minimales d'un graphe, d'un point de vue plus local. Nous introduirons notamment les notions de clique maximale potentielle et de famille maximale de séparateurs voisins. Ces objets caractérisent toutes les cliques maximales de toutes les triangulations minimales d'un graphe G.

Les notions de clique maximale potentielle et de famille maximale de séparateurs voisins sont complètement nouvelles. Elles étaient introduites par Bouchitté et Todinca [BT98a, BT98b] et qui a permit d'unifier les algorithmes déjà existants de calcul de la largeur arborescente et de la complétion minimales pour certaines classes de graphes avec un nombre polynomial de séparateurs. Plus exactement, Todinca [Tod99] a montré que tous ces algorithmes calculent de manière implicite toutes les

cliques maximales potentielles du graphe en entrée. De plus, ces notions on permit pour résoudre ces problèmes pour une autre classe, à savoir les graphes faiblement triangulés. Tout ceci donne de bonnes raisons de penser que le calcul de ces paramètres de triangulation pour tous les graphes avec ‘‘peu’’ de séparateurs minimaux passe par la connaissance des cliques maximales potentielles. En tout cas nous montrerons dans les chapitres suivants que ces objets sont suffisants pour calculer la largeur arborescente et la complétion minimale d'un graphe.

Nous ne donnerons dans ce chapitre que les définitions et les propriétés structurelles fondamentales des cliques maximales potentielles et des familles maximales de séparateurs voisins.

Pour avoir un aperçu sur les cliques maximales potentielles, reprenons l'exemple du cycle. Toute triangulation est obtenue en complétant une famille de séparateurs deux à deux parallèles, comme sur la Figure 4.1. Nous obtenons des ‘‘régions’’ triangulaires entre ces séparateurs et les cliques maximales des triangulations sont toutes des triplets de sommets $\{i, j, k\}$. Cela nous suggère que toutes les triangulations du cycle sont déterminées par ces ‘‘cliques maximales potentielles’’ formées par des triplets de sommets.

Nous nous donnerons ici les moyens de reconnaître une clique maximale potentielle : nous répondrons à la question suivante : *étant donné un ensemble de sommets K d'un graphe G, existe-t-il une triangulation minimale H de G telle que K soit une clique maximale de H ?*.

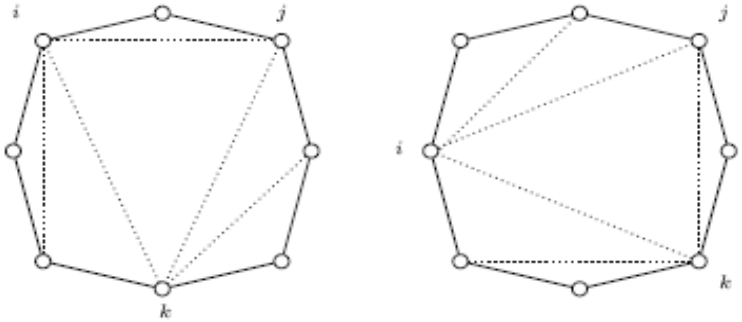


Figure 4.1- Triangulations du cycle

4.1 Cliques maximales des graphes triangulés

Nous commençons par quelques observations sur la formation des cliques maximales des graphes triangulés en fonction des séparateurs minimaux du graphe, dans le but de les étendre aux cliques des triangulations minimales des graphes quelconques. Introduisons d'abord une nouvelle notion. En considérant une famille de séparateurs deux à deux parallèles et telle qu'aucun d'entre eux ne sépare deux autres -les

séparateurs S_1, S_2, S_3 de la Figure 4.2 illustrent cette situation - on voit qu'ils déterminent une zone du graphe qui se trouve ‘‘entre’’ eux. Plus formellement :

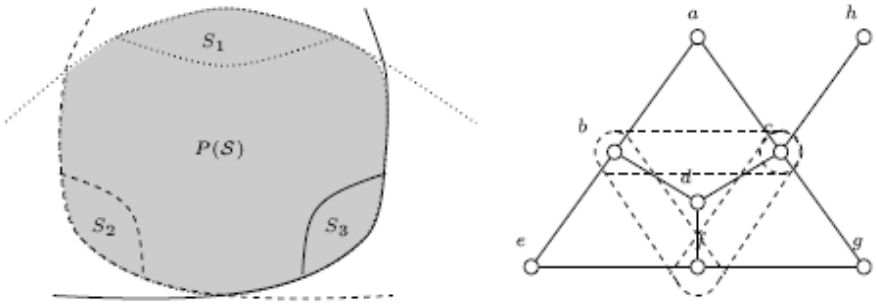


Figure 4.2-Partie entre une famille de séparateurs

Définition 4.1 (Partie entre une famille de séparateurs) Soit G un graphe quelconque et $\zeta \subseteq G$ une famille de séparateurs minimaux telle que pour tout $S \in \zeta$, il existe un bloc $(S, C_G(S))$ contenant tous les séparateurs de ζ. Si ζ n'a pas d'élément maximum pour l'inclusion, on définit la partie entre les séparateurs de ζ par :

$$P(\zeta) = \bigcap_{S \in \zeta} (S, C_G(S))$$

Sur l'exemple de la Figure 4.2, si l'on prend les séparateurs entourés en pointillés $\zeta = \{\{b,c\},\{c\},\{b,f\},\{c,f\}\}$ on obtient $P(\zeta) = \{b,c,d,f\}$.

Si la partie entre une famille ζ de séparateurs minimaux existe, les séparateurs de ζ doivent être deux à deux parallèles. En effet, si $S,T \in \zeta$, puisque $T \subseteq (S, C_G(S))$ on a $S \| T$. D'un autre côté, si l'on se donne une famille ζ' de séparateurs minimaux deux à deux parallèles, sans élément maximum par inclusion, la partie entre ces séparateurs existe si et seulement si aucun $S \in \zeta'$ ne sépare deux éléments $T,U \in \zeta'$. Notons aussi que, d'après la définition, si $P(\zeta)$ existe, alors elle contient tous les séparateurs de S.

Dans la suite nous nous intéresserons aux cliques maximales d'un graphe triangulé et à leur position par rapport aux séparateurs qu'elles contiennent. Si K est un ensemble de sommets de G, nous noterons par $\Delta_G(K)$ l'ensemble des séparateurs minimaux de G contenus dans K.

Notation .||. : Parallèle.

Lemme 4.2 ([Tod99]) Soient G un graphe, S un séparateur minimal et une clique de G. Alors Ω est incluse dans un bloc de S. Si T est un séparateur minimal et $G[T]$ est une clique, alors $S \| T$. En particulier, les séparateurs minimaux d'un graphe triangulé sont deux à deux parallèles.

Lemme 4.3 ([KKM95]) Soient *H* un graphe triangulé et une de ses cliques maximales. Alors soit $\Delta_H(\Omega)$ a un élément maximum par inclusion, soit $P(\Delta_H(\Omega))$ existe et contient Ω.

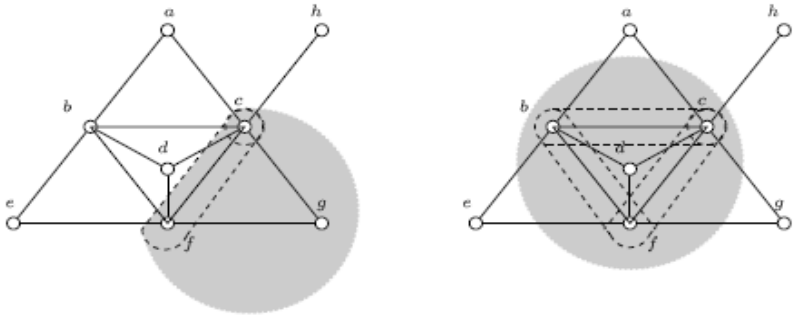


Figure 4.3- Cliques maximales

Théorème 4.4 Soit *H* un graphe triangulé et une clique maximale de *H*. Si tous les éléments de $\Delta_H(\Omega)$ sont contenus dans un séparateur minimal $S \in \Delta_H(\Omega)$, alors est un bloc (S,C) de *S*. Si $\Delta_G(\Omega)$ n'a pas d'élément maximum par inclusion, alors $\Omega = P(\Delta_H(\Omega))$.

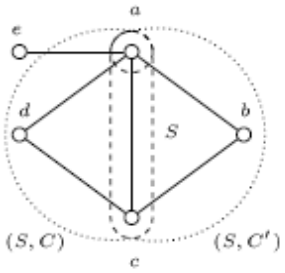


Figure 4.4-Cliques maximales - cas dégénéré

4.2 Cliques maximales potentielles

Une clique maximale potentielle d'un graphe est un ensemble de sommets qui induit une clique maximale dans au moins une triangulation minimale de ce graphe. Il a été prouvé que si ces objets peuvent être énumérés en temps polynomial pour une classe de graphes, la largeur arborescente et la complétion minimale sont calculables en temps polynomial pour ces graphes. Nous montrons ici que les cliques maximales potentielles d'un graphe peuvent être générées en temps polynomial par rapport au nombre de ses séparateurs minimaux.

En conséquence, la largeur arborescente et la complétion minimale sont calculables en temps polynomial pour tous les graphes ayant un nombre polynomial de séparateurs minimaux.

Signalons que la terminologie de clique maximale potentielle est reprise d'après les travaux indépendants de Kloks, Kratsch et Müller [KKM98].

Cette définition est de nature combinatoire, au sens où pour identifier une clique maximale potentielle de *G* il nous faut passer par les triangulations minimales du graphe. L'objectif de cette section est de montrer que les cliques maximales potentielles peuvent se définir de façon parfaitement locale - algorithmique, si l'on veut-en restant dans le graphe *G* et sans utiliser les triangulations minimales.

La clique maximale potentielle est fortement liée aux séparateurs qu'elle contient, tout comme une clique maximale d'un graphe triangulé.

Définition 4.5 (Famille maximal de séparateurs) Une famille *S* de séparateurs minimaux d'un graphe *G* est une famille maximale de séparateurs voisins s'il existe une clique maximale potentielle Ω de *G* telle que $S = \Delta_G(\Omega)$ (l'ensemble des séparateurs minimaux de *G* contenus dans Ω).

Une clique maximale potentielle d'un graphe *G* est un ensemble de sommets de *G* qui induit une clique d'une triangulation minimale de *G*.

Nous justifierons plus loin l'appellation de ''séparateurs voisins''.

Avant de donner une généralisation du théorème 4.4, qui nous permettra de reconnaître les cliques maximales potentielles et les familles maximales de séparateurs voisins, nous avons besoin de quelques résultats.

Propriété 4.6 ([Dir61]) Les séparateurs minimaux d'un graphe triangulé sont deux à deux parallèles.

Théorème 4.7 ([Tod99])

Soient *H* un graphe triangulé et une clique maximale de *H*. Si tous les éléments de $\Gamma_H(\Omega)$ sont inclus dans un séparateur minimal *S* de $\Gamma_H(\Omega)$, alors Ω est un bloc $C \cup S$ où *C* est une composante connexe de $C_H^*(S)$. Dans le cas contraire, $\partial(\Gamma_H(\Omega))$[1] existe et est égal à G_{Δ_H}.

Soit *H* une triangulation minimale d'un graphe *G*. L'ensemble Δ_H est une famille de séparateurs minimaux deux à deux parallèles de *G* maximale pour l'inclusion et *H* est égale à G_{Δ_H} (le graphe G_{Δ_H} correspondant au sous-graphe $G\left[C_{\Delta_H} \cup N(C_{\Delta_H})\right]$ dans lequel nous avons complété $N(C_{\Delta_H})$ en une clique).

Théorème 4.8 [Tod99] Soit *G* un graphe et Ω un ensemble de sommets de *G*. Supposons que $\Delta_G(\Omega)$ ait un élément maximum *S*. i.e, chaque $T \in \Delta_{G(\Omega)}$ est contenu dans *S*. Alors Ω est une clique maximale potentielle si et seulement si les deux conditions sont satisfaites :

[1] La partie entre la famille $\Gamma_H(\Omega)$ de séparateurs voisins.

1. est un bloc $(S,\ C)$ de *S* dans *G*.
2. $G_{\Delta_G(\Omega)}[\Omega]$ est une clique.

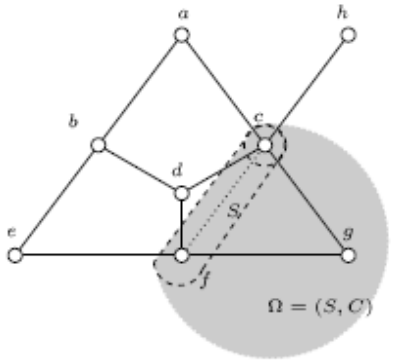


Figure 4.5- Clique maximale potentielle : premier cas.

Théorème 4.9 Soit *G* un graphe et un ensemble de sommets de *G*. Supposons que $\Delta_G(\Omega)$ n'ait pas d'élément maximum pour l'inclusion. Alors Ω est une clique maximale potentielle si et seulement si les deux conditions sont satisfaites :

1. est la partie entre les éléments de $\Delta_G(\Omega)$.
2. $G_{\Delta_G(\Omega)}[\Omega]$ est une clique.

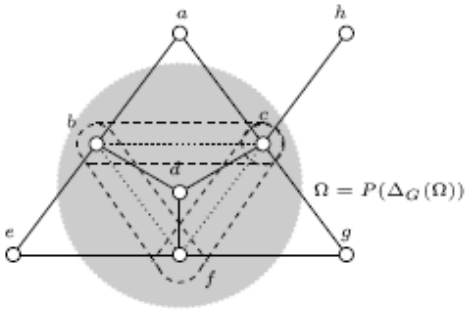


Figure 4.6- Clique maximale potentielle : deuxième cas

Les théorèmes 4.8 et 4.9 fournissent un moyen de reconnaître une clique maximale potentielle. La chose la plus importante est que si l'on a un graphe *G* et l'ensemble de ses séparateurs Δ_G, ces deux théorèmes nous donnent un algorithme pour savoir si un ensemble de sommets forme une clique maximale potentielle. Ceci a permit pour Bouchitté et Todinca dans [BT98] de calculer toutes les cliques maximales potentielles pour certaines classes de graphes. A ce jour nous n'avons pas d'algorithme énumérant toutes les cliques maximales potentielles d'un graphe quelconque en un temps polynomial en la taille du graphe et le nombre de ces cliques potentielles, comme on sait le faire pour les séparateurs minimaux.

Notons aussi que l'on peut reconnaître directement si une famille de séparateurs ξ est une famille maximale de séparateurs voisins. Si la famille possède un élément *S* maximum par inclusion, il suffit de vérifier si pour un des blocs $(S,\ C)$ de *S* dans *G*, l'ensemble de sommets $(S,\ C)$ est une clique maximale potentielle et si $\zeta = \Delta_G(S \cup C)$. Si ξ n'a pas de maximum, il faut que $P(\xi)$ existe, que ce soit une clique maximale potentielle de G et enfin nous devons avoir $S = \Delta_G(P(\xi))$.

Si nous avons utilisé pour ces familles le terme de ‘’séparateurs voisins’’, c'est parce que aucun autre séparateur minimal ne peut s'intercaler entre eux sans les croiser. Ils sont d'une certaine façon compacts, proches les uns des autres. Si l'on regarde le graphe de la figure 4.7 (les arêtes du graphe sont marquées en continu), les séparateurs minimaux $\{a,b\}$, $\{a,c\}$, $\{a,d\}$, $\{b,c\}$, $\{b,d\}$et $\{c,d\}$ forment une famille maximale de séparateurs voisins et bordent la clique potentielle $\{a,b,c,d\}$. Ceci est quelque part en contradiction avec la notion habituelle de proximité que l'on a dans un graphe, on imagine mal en quoi les séparateurs $\{a,d\}$et $\{b,c\}$ sont voisins. C'est peut-être ce qui explique en partie la difficulté que nous avons à trouver un algorithme d'énumération des cliques maximales potentielles et des familles de séparateurs voisins. Nous avons vraiment besoin de tous les séparateurs de la famille pour comprendre s'ils sont voisins (dans notre sens) ou pas. Comme dans le cas des graphes triangulés, une famille maximale de séparateurs voisins ayant un plus grand élément peut border plusieurs cliques maximales.

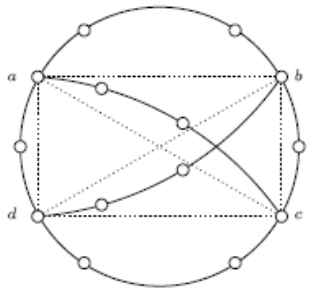


Figure 4.7- Séparateurs voisins

Algorithme 4.1 Reconnaissance des cliques maximales potentielles

```
Entrée : le graphe G = (V, E) et l'ensemble de sommets K ⊆ V
Sortie : booléen indiquant si K est une clique maximale potentielle

calculer les composantes connexes C_i de G\K
calculer les ensembles S_i
si il existe un i t.q. S_i = K alors
    retourner "K n'est pas une clique potentielle"
pour i = 1 à p
    compléter S_i
pour chaque couple de sommets x, y ∈ K
    si x et y ne sont pas adjacents dans G_{S_1,...,S_p} alors
        retourner "K n'est pas une clique potentielle"
retourner "K est une clique potentielle"
```

4.3 Triangulations serrées

Pour construire des triangulations, nous pouvons utiliser le théorème de caractérisation des graphes triangulés 1.19 « un graphe est triangulé si et seulement si tous ses séparateurs minimaux induisent des cliques ». Celui-ci suggère l'algorithme suivant : tant que *G* n'est pas triangulé, choisir un séparateur minimal *S* qui n'induit pas de clique de *G* et compléter *S* en une clique de *G*.

Nous obtenons ainsi une famille croissante de graphes dont le dernier élément est triangulé. De plus, un séparateur minimal S' d'un graphe complété est un séparateur minimal de graphe initial et comme S' ne peut pas séparer les sommets de *S*, les séparateurs minimaux du graphe complété sont parallèles à *S*. La triangulation finale est obtenue en complétant des séparateurs minimaux deux à deux parallèles de *G*.

Notation

Soit Γ une famille de séparateurs minimaux deux à deux parallèles d'un graphe *G*. Le graphe G_Γ est le graphe *G* lequel nous avons ajouté des arêtes de sorte que chaque élément de Γ induise une clique de G_Γ.

Définitions 4.10 (Triangulations minimales) Une triangulation minimale d'un graphe est une triangulation minimale pour l'inclusion.

Les triangulations obtenues par le processus ci-dessus sont des triangulations minimales. De plus, toutes les triangulations minimales sont de cette forme comme l'énonce le théorème de Para et Scheffler.

Théorème 4.11 ([PS97])

Soit Γ une famille de séparateurs minimaux deux à deux parallèles de *G* maximale pour l'inclusion. Le graphe Δ_G est une triangulation minimale de *G*.

Soit *H* une triangulation minimale d'un graphe *G*. L'ensemble Δ_H est une famille de séparateurs minimaux deux à deux parallèles de *G* maximale pour l'inclusion et *H* est égale à G_{Δ_H}.

Une méthode plus générale pour construire des triangulations consiste à utiliser la propriété 4.6 et le théorème 4.7. Ceux-ci montrent que les séparateurs minimaux d'un graphe triangulé *H* sont deux à deux parallèles et que les cliques maximales de *H* forment des blocs[2] minimaux. Ainsi, les cliques maximales de *H* sont entièrement caractérisées par les séparateurs minimaux de *H*. Pour construire une triangulation d'un graphe *G*, il suffit de se donner une famille $\Im$ de séparateurs minimaux deux à deux parallèles et de compléter les blocs minimaux que forment les séparateurs de $\Im$ en des cliques. Si la famille $\Im$ est maximale pour l'inclusion, nous retrouvons les triangulations minimales.

Pour caractériser les triangulations que produit ce second processus, nous pouvons remarquer que les séparateurs minimaux d'une telle triangulation sont des séparateurs minimaux de G. Cependant, comme l'illustre la figure 4.8, ce n'est pas suffisant. Par construction, les cliques maximales d'une telle triangulation sont des blocs de *G* et donc les séparateurs minimaux de H doivent induire les mêmes composantes connexes dans *G* et dans *H*.

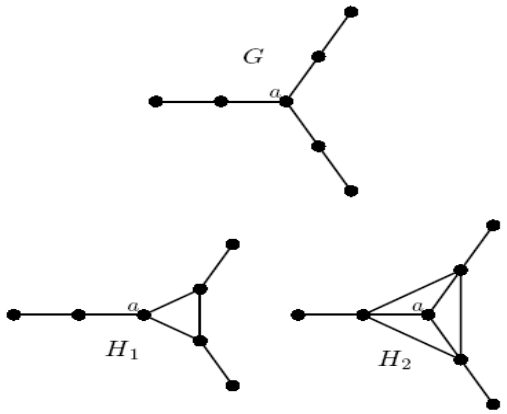


Figure 4.8–Un bloc de *H* n'induit pas forcément un bloc de *G*.

[2] Un bloc de *G* est soit une partie entre séparateurs voisins, soit un ensemble de la forme $S \cup C$ avec *C* un élément de $C_G^*(S)$.

Les séparateurs minimaux des graphes H_1 et H_2 sont des séparateurs minimaux du graphe triangulé G.

Le séparateur {a} n'induit pas les mêmes composantes connexes dans G et dans H_1.
Le graphe H_1 n'est pas une triangulation serrée de G alors que le sur-graphe H_2 de H_1 en est une.

Définition 4.12 (Triangulation serrée) Une triangulation *H* d'un graphe *G* est serrée si les séparateurs minimaux de *H* sont des séparateurs minimaux de *G* et si les séparateurs minimaux de *H* induisent les mêmes composantes connexes de *H* et de *G*.

Dans la suite de ce mémoire, nous nous intéressons uniquement aux triangulations serrées.

4.4 Familles complètes de blocs d'un graphe

Nous nous intéressons au problème suivant : étant donnée une famille de blocs *F*, existe-t-il une triangulation dont les blocs appartiennent à *F* ? Une telle triangulation n'existe pas toujours mais plus nous avons de blocs, plus il y a de chances que ce soit le cas. En contrepartie, plus la famille initiale est grande, plus la construction d'une éventuelle triangulation prend de temps.

Définitions 4.13 (Famille complète de blocs)
Si *F* est une famille de blocs d'un graphe *G*, nous dirons d'un séparateur minimal qu'il borde *F* s'il borde un bloc de *F*. Nous notons Γ_F l'ensemble $\bigcup_{\Omega \in F} \Gamma_\Omega$ des séparateurs minimaux qui bordent *F*[3].
Une famille *F* de blocs est complète si pour tout séparateur *S* de Γ_F et pour toute composante connexe pleine *C* de $G \setminus S$, il existe un bloc Ω de *F* bordé par S et inclus dans $C \cup S$.
Une famille complète peut être vide mais si ce n'est pas le cas, nous montrons que nous pouvons en extraire une triangulation de *G*. En particulier, si *H* est une triangulation serrée de *G*, l'ensemble des cliques maximales de *H* forme une famille complète de blocs de *G*.

Lemme 4.14 [Tod99] L'ensemble des cliques maximales d'un graphe triangulé *H* forme une famille complète de blocs de *H*.
L'ensemble des cliques maximales d'une triangulation serrée *H* d'un graphe *G* forme une famille complète de blocs de *G*.

[3] Γ_Ω désigne l'ensemble des séparateurs minimaux qui bordent Ω.

À partir d'une famille de blocs *F*, nous cherchons à construire une famille complète F_c incluse dans *F*. Pour cela, nous pouvons procéder par élimination. En effet, si un bloc Ω de *F* appartient à une famille complète, par définition, pour chaque séparateur *S* le bordant et chaque composante pleine *C* de $\zeta_G^*(S)$[4] , il existe un bloc Ω' de *F* bordé par *S* et inclus dans $C \bigcup S$.

S'il n'existe pas de tel bloc Ω' , nous pouvons éliminer Ω car il n'appartient à aucune famille complète incluse dans *F*. Ainsi, tous les blocs qui restent à l'issue de ce processus appartiennent à une famille complète incluse dans *F*.

En fait, l'ensemble de bloc obtenu est lui-même une famille complète ; c'est la plus grande famille complète incluse dans *F*.

L'algorithme 4.2 formalise ce processus.

Algorithme 4.2 elimination

- `Entrée` : G, ses séparateurs minimaux et ses cliques maximales potentielles ; l'entier k.
- `Sortie` : la plus grande famille complète $\mathcal{F}$ de cliques maximales potentielles, dont tous les éléments ont au plus $k+1$ sommets.

1. Calculer toutes les cliques maximales potentielles de G ayant au plus $k+1$ sommets. Soit $\mathcal{F}$ la famille de ces cliques potentielles et soit $\Gamma_{\mathcal{F}}$ l'ensemble des séparateurs minimaux contenus dans ces cliques.
2. Extraire de $\mathcal{F}$ la plus grande sous-famille complète de cliques potentielles :
   `tant que` il existe un $S \in \Gamma_{\mathcal{F}}$ et un bloc (S, C) de S tels qu'aucune clique potentielle $\Omega \in \mathcal{F}$ ne satisfait $S \subset \Omega \subseteq (S, C)$ `faire`
   - $\mathcal{F} = \mathcal{F} \backslash \{\Omega | S \subset \Omega\}$.
   - mettre à jour $\Gamma_{\mathcal{F}} = \{S | \exists \Omega \in \mathcal{F}$ tel que $S \subset \Omega\}$.
3. `retourner` $\mathcal{F}$.

Il est clair -par la définition même d'une famille complète- que la famille *F* de cliques maximales potentielles retournées par cette procédure est complète. Montrons qu'elle contient toutes les « bonnes » cliques potentielles maximales. Nous verrons plus loin qu'il s'agit bien de la plus grande famille complète de cliques maximales potentielles de taille au plus *k* +1.

[4] L'ensemble des composantes pleines par rapport à *S*.

Lemme 4.15 [Tod99] Soient *G* un graphe et *H* une triangulation minimale de *G* telle que $tw(G) \leq k$, i.e. $\omega(G) \leq k+1$. Si l'on applique à *G* l'algorithme d'élimination, toute clique maximale de *H* appartiendra à la famille *F* retournée par l'algorithme.

Algorithme 4.3 extraction_aux

entrée

G : un Graphe
S : un séparateur minimal de G ou $\emptyset$
$C \in C_G^*(S)$: une composante pleine de S
$\mathcal{F}$: une famille complète de G
Γ : l'ensemble des séparateurs qui bordent $\mathcal{F}$

sortie :

$\mathcal{T}$: un arbre des cliques enraciné d'une triangulation H de $G[C \cup S]$ dont les cliques maximales sont des blocs de $\mathcal{F}$ et dont la racine contient S.

début
  si $S = \emptyset$ **alors**
    Soit $\Omega \in \mathcal{F}$ avec $\Omega \subseteq C \cup S$
  sinon
    Soit $\Omega \in \mathcal{F}$ tel que $S \subseteq \Omega \subseteq C \cup S$
  $\mathcal{T} \leftarrow$ l'arbre ayant comme unique étiquette Ω
  si $\Omega \neq C \cup S$ **alors**
    pour chaque $S' \in \Gamma_G(\Omega) \backslash \{S\}$ non inclus dans S et
        $C' \in C_G^*(\Omega)$ tels que $\Omega \not\subseteq C' \cup S'$ **faire**
      $\mathcal{T}' \leftarrow$ `extraction_aux`$(G, S', C', \mathcal{F}, \Gamma)$
      créer une arête entre les racines de $\mathcal{T}'$ et $\mathcal{T}$
  rendre $\mathcal{T}$
fin

Algorithme 4.4 extraction

entrée :

$\mathcal{F}$: une famille complète de blocs d'un hyper-graphe G connexe.
Γ : l'ensemble des séparateurs qui bordent $\mathcal{F}$

sortie :

$\mathcal{T}$: un arbre des cliques d'une triangulation H de G dont les cliques maximales sont des blocs de $\mathcal{F}$

début
  rendre `extraction_aux`$(G, \emptyset, V, \mathcal{F}, \Gamma)$
fin

Corollaire 4.16 Si *F* est une famille complète de blocs d'un graphe connexe, il existe une triangulation de *G* dont les cliques maximales sont des blocs de *F*. L'algorithme **extraction** est calcule une.
Ces deux algorithmes sont exactement ceux décrits dans la thèse de Todinca [Tod99] mais présentés dans un cadre plus général. Il donne la propriété suivante :

Propriété 4.17 Si nous notons *r* le nombre de séparateurs minimaux d'un graphe *G*, *b* le nombre de couples (C_S, S) où *S* est un séparateur minimal de *G* et C_S une composante pleine dans $\zeta_G^*(S)$ et *p* le nombre de blocs, la complexité de l'algorithme **elimination** est $O(prn + bn)$ et celle de l'algorithme extraction est $O(prn \log n)$.
Comme *b* est au plus égal à *nr* et *r* au plus à *np*, ces algorithmes fonctionnent en temps polynomial en fonction du nombre de blocs de la famille d'entrée.

4.5 Conclusion

Nous avons introduit les cliques maximales potentielles d'un graphe qui sont exactement les ensembles de sommets du graphe *G* formant des cliques maximales dans les triangulations minimales *H* de *G*. Nous verrons dans le chapitre qui suit en quoi cet objet est un outil fondamental pour le calcul de la largeur arborescente et de la complétion minimale.

Le but de ce chapitre était de remplacer cette définition combinatoire par une caractérisation de nature algorithmique des cliques maximales potentielles. Nous avons vu que, étant donné un ensemble *K* de sommets de *G*, nous pouvons vérifier facilement s'il s'agit d'une clique maximale potentielle.

5. Calcul des Invariants

Une bonne idée est un moyen de rendre simple
et de résoudre un problème de manière inattendue.
R.E. Tarjan

Dans ce chapitre nous présentons nos différents résultats obtenus sur le calcul des invariants ; la largeur arborescente, la longueur arborescente et la longueur de branches.

Nous traitons dans ce qui suit la classe des graphes *k*-cordaux et les graphes planaires. Nous présentons dans cette première partie nos résultats, propositions et théorèmes, sur le calcul de la largeur arborescente ce qui nous a conduit à développer un algorithme efficace, exact pour quelques graphes vérifiant certaines conditions, tandis que, il donne une borne serrée pour d'autres.

5.1 Calcul de la largeur arborescente

Notations

Pour la bonne compréhension nous adaptons les notations suivantes :

- S_a : Un sac contient le sommet *a*.
- $\deg_{\min}(G)$: Le degré minimum du graphe *G*.
- $\deg_a$: Le degré du sommet *a*.

Proposition 5.1
La valeur de la largeur arborescente, $tw(G)$, de la décomposition minimale d'un graphe *G* est supérieure ou égal à la valeur de son degré minimum, $tw(G) \geq \deg_{\min}(G)$.

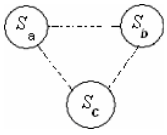
□ Par absurde, supposons que $tw(G) < \deg_{\min}(G)$).

1-Cas particulier (les cliques)

Soit K_n une clique de cardinalité *n*, le degré de tous ses sommets est égal à $n-1$.

Supposons que $tw(K_n) < n-1$; la taille des sacs de la décomposition arborescente de la clique K_n ne dépasse pas $n-1$.

Sans perdre de généralité, supposons que nous avons les sommets *a*, *b*, *c* de la clique K_n, qui sont deux-à-deux adjacents. D'après la définition 3.2, nous avons au moins deux sacs, un sac contenant le sommet *a* et un autre contenant le sommet *b*, tel que $c \in S_a$ et $c \in S_b$. Alors on aura une chaîne, ou bien une arête, qui lie S_a et S_b. Ces derniers sont forcement liés à S_c, ce qui induit un cycle. Ce qui est absurde avec le principe de la décomposition arborescente.



2-Cas général

Etant donné un graphe $G(V,E)$, sans perdre de généralité, soit *a* le sommet de degré minimum, noté $\deg_a$.

Par absurde, supposons que $tw(G) \geq \deg_a$.

Le principe de l'algorithme de triangulation de la décomposition arborescente réduite donné par Berry [Ber98], est $\forall x \in V$, l'ensemble des voisins de *x* forme une clique.

Partant de ce principe, la décomposition arborescente réduite d'une clique est composée d'un seul sac contenant tous les sommets de cette clique. On en déduit que $\exists x \in V$ tel que *x* est voisin à au moins un sommet *y* de ce sac, *x* pouvant être *a*.

Contradiction avec l'hypothèse du départ, car le choix du premier sac de la décomposition arborescente influe sur la valeur de la largeur arborescente minimale.

Cependant, l'inégalité $tw(G) \geq \deg_a$ est toujours vérifiée. Si on commence la décomposition par le sommet *a*, le sac S_1 est composé de *a* et l'ensemble de ses voisins. Nous avons $|S_1| = \deg_a$, alors on a l'inegalité $tw(G) \geq \deg_a$ est toujours vérifié et si on choisit un autre sommet de degré supérieur à celui de *a*, on aura $tw(G) > \deg_a$.

Donc la largeur arborescente de *G* est toujours supérieure au degré minimum de ses sommets.

■

Proposition 5.2

La largeur arborescente d'un graphe $G(V,E)$ est égale à la largeur arborescente de son sous-graphe induit G', le graphe G' est déterminé par élimination des sommets pendants du graphe *G*, $tw(G) = tw(G')$ / $tw(G) = tw(G \setminus \ell)$ (ℓ est l'ensemble des sommets pendants).

□ Soit *a* un sommet pendant de *G*. On a $a \notin G^{'}$ et $ab \in E$.

Lorsqu'on décompose le graphe $G^{'}$ en sacs, on aura au moins un qui contient le sommet *b*. L'ajout d'un sac contenant seulement le sommet *a* et le sommet *b*, noté S_{ab}, ne viole aucune règle de la décomposition arborescente donné par la définition 3.2, ce qui implique que cette arborescente est une décomposition arborescente pour le graphe *G*.

La décomposition arborescente minimale de *G* reviens à décomposer $G^{'}$ et de lui ajouter des sacs de type S_{ij}, tel que *i* est le sommet pendant et $ij \in E$. Voir la Figure 5.1.

On a $tw(G) = \max(tw(G^{'}), 1)$ et comme $tw(G^{'}) \geq 1$ on aura $tw(G) = tw(G^{'})$.

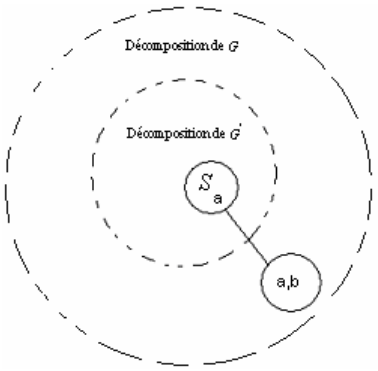


Figure 5.1-Décomposition de $G^{'}$ par rapport à *G*.

■

Algorithme 5.1

entrée //Décomposition du graphe $G^{'}$ et l'ensemble ℓ est l'ensemble des sommets pendants.

sortie // Décomposition de G .

début

tant que $\ell \neq \phi$ **faire**

- choisir *a* un sommet de ℓ ;

- choisir un sac de la décomposition $G^{'}$ qui contient le sommet adjacent à ce sommet, et on relie le sac avec un sac qui contient le sommet pendant et le sommet adjacent.

$\ell \leftarrow \ell \setminus \{a\}$.

fait.

fin.

Complexité de l'algorithme 5.1

L'algorithme est linéaire d'ordre $O(\ell)$.

Remarque

Par la suite de ce mémoire, on considère que les graphes sont sans sommets pendants, $G = G \setminus \ell$.

Notations

$\deg_{\max}(G)$: le degré maximum du graphe *G*.

NC désigne le nombre de cycles dans *G*.

$V(a)$ désigne l'ensemble des voisins de *a*.

Théorème 5.3

Si le $\deg_{\max}(G)$ est égale au nombre de cycles dans *G* alors la largeur arborescente $tw(G)$ doit être l'entier 2 ou 3, $2 \le tw(G) \le 3$.

□ Soit *a* un sommet de degré maximum. On a deux cas possible :

1er cas $\deg(a) = n - 1$.

Posons $V(a)$ l'ensemble des voisins de *a*.

$G = \{a\} \cup V(a)$ (le sommet *a* est adjacent a tous les sommets de $G \setminus \{a\}$) voir Figure 5.2.

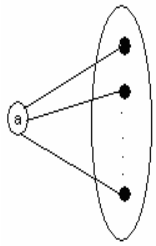


Figure 5.2 -Exemple $G = \{a\} \cup V(a)$

Le graphe de la Figure 5.3 représente le seul cas possible pour la condition donnée (par construction).

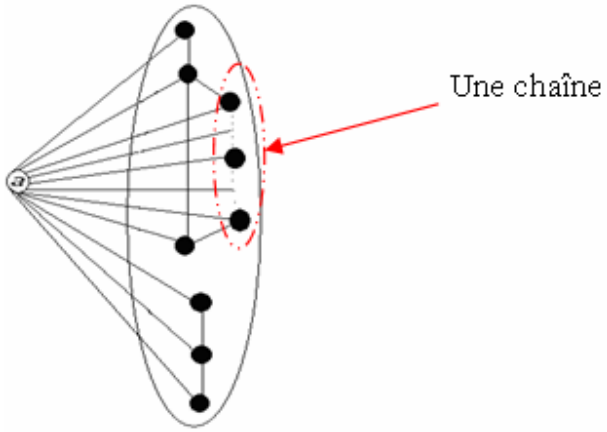
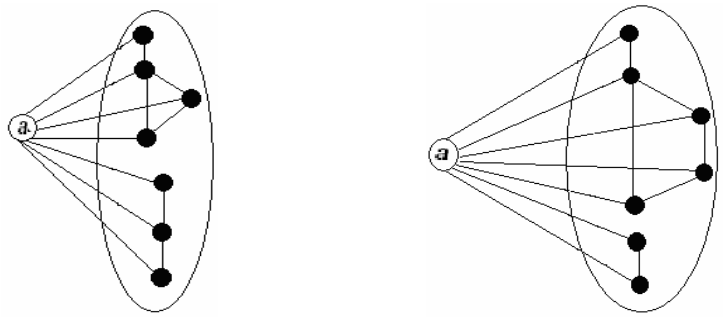


Figure 5.3- Le graphe *G* '1er cas'



'A' 'B'

Figure 5.4 – Exemple du '1er cas'

Le graphe 'A' est un graphe triangulé tel que sa clique maximale est de taille 4.

La construction du graphe dans ce premier cas, impose que pour avoir la condition $NC = \deg_{\max}(G)$, il faut que le graphe *G* admet le graphe *H* comme graphe induit. Voir la Figure 5.5.

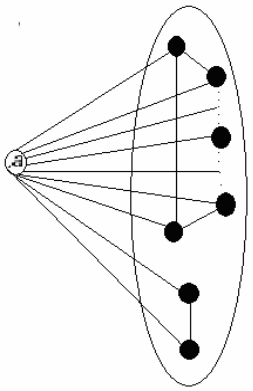
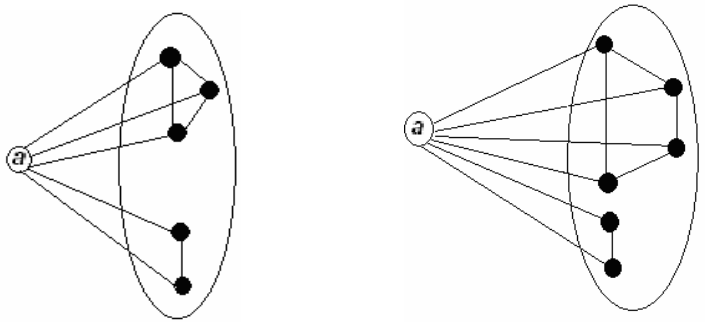


Figure 5.5- Graphe induit *H* du '1er cas'

Les graphes de la Figure 5.4 ('A' et 'B') sont des graphes qui admettrent les graphes de la figure 5.6 ('A' et 'B') comme des graphes induits.

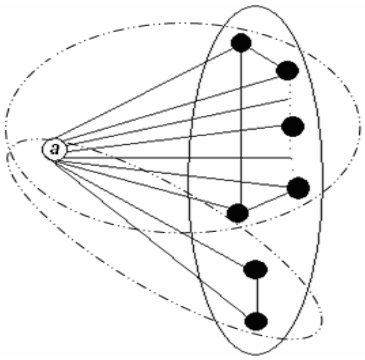


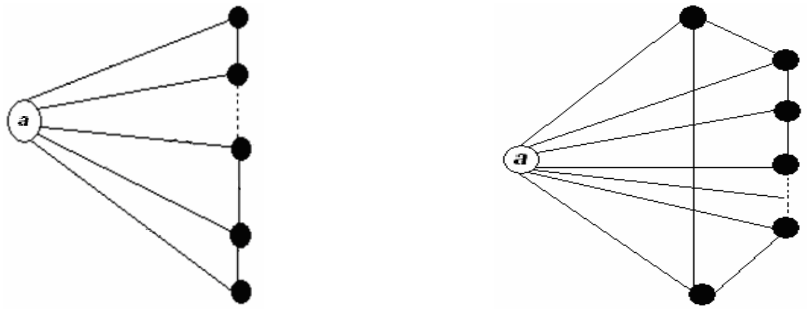
'A' **'B'**

Figure 5.6- Exemples des graphes induits pour le 1er cas

Pour ce graphe, nous avons le sommet *a* est un sommet d'articulation pour le graphe *G* où sa suppression induit deux composantes connexes ; une des composante est un cycle (noté : ς) et l'autre est une chaîne (noté : *P*). Voire la Figure 5.6.

La décomposition arborescente de ces graphes, revient à décomposer en sacs le graphe induit par le sommet *a* et la composante connexe des sommets du cycle ς, car les autres sommets (les sommets de la composante chaîne *P* et les sommets liées aux sommets de la composante cycle ς) sont décomposées en sacs de taille trois.





$G_1 = P \cup \{a\}$ $G_2 = \varsigma \cup \{a\}$

Figure 5.7- $G = G_1 \cup G_2$

On a la largeur arborescente de *G* égale à la valeur maximum entre la décomposition des graphes G_1 et G_2 ; $tw(G) = \max(tw(G_1), tw(G_2))$.

Pour mieux comprendre cette décomposition on procède étape par étape.

1^er^ étape : Décomposition arborescente minimale du graphe induit par le sommet *a* et l'ensemble des sommets de la composante chaîne *P* (le graphe G_1).

Soit un graphe G_1

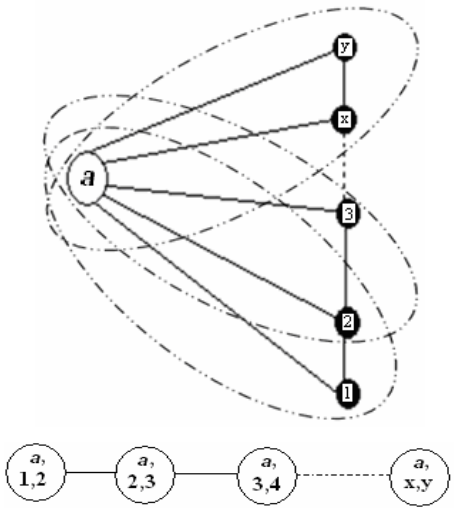


Figure 5.8- décomposition arborescente de G_1 ($tw(G_1)=2$)

Chaque sac contient trois sommets implique que la largeur arborescente de G_1 égale à 2.

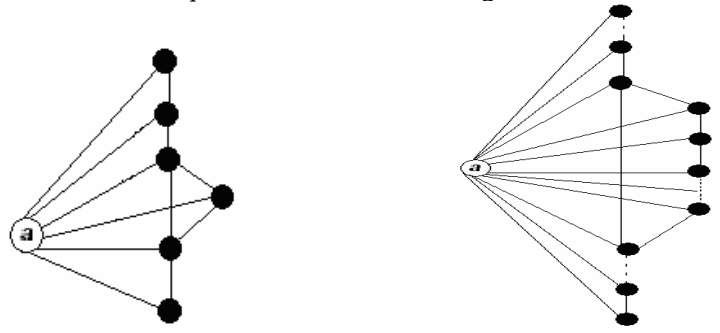
2^ieme étape : Décomposition arborescente minimale du graphe induit par le sommet *a* et l'ensemble des sommets de la composante cycle ς (le graphe G_2).

Dans cette étape on distingue deux types de graphes différents :

Les graphes triangulés et les graphes non triangulés. Voir la Figure 5.9.

Le graphe 'B' est un graphe non triangulé, la décomposition minimale de ce graphe nous oblige a décomposer le graphe en sacs de taille trois sauf un, il est de taille quatre. Voir la Figure 5.10.

Le graphe 'A' est un cas particulier du graphe 'B' et il est triangulé, sa décomposition nous oblige a le décomposer en sacs de taille trois sauf un, il est de taille quatre a cause de l'existence de la clique de taille 4. Voir la Figure 5.11.



'A' 'B'

Figure 5.9-Le graphe G_2

On décompose le graphe 'B' par parties :

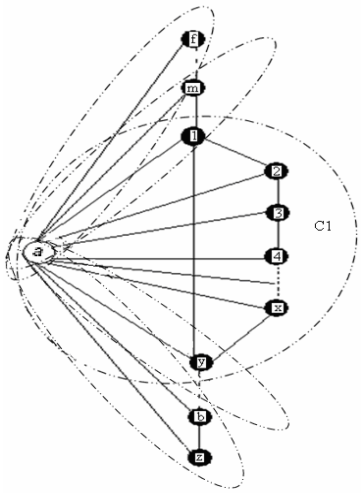


Figure 5.10- le graphe induit G_2

On commence par la décomposition arborescente de $C1$ et on utilise ce résultat pour décomposer G_2.

La décomposition minimale de la composante $C1$ en sacs de taille quatre.

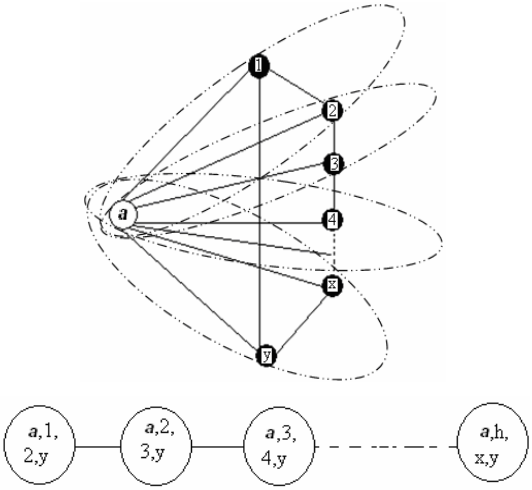


Figure 5.11-Décomposition de la composante $C1$ ($tw(C1)=3$).

Après avoir décomposer la composante $C1$, on peut alors faire la décomposition de G_2. Voir Figure 5.12.

Tous les sacs de $G_2 \setminus C1$ sont de taille trois.

$tw(G_2)=\max(tw(C1),tw(G_2 \setminus C1))$.

$tw(C1)=3$ et $tw(G_2 \setminus C1)=2$ alors $tw(G_2)=3$.

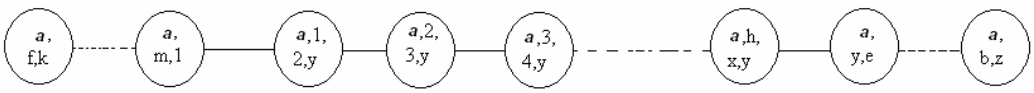


Figure 5.12- Décomposition arborescente de G_2

Dans la Figure 5.9 le graphe 'A' est un cas particulier de cette décomposition minimal.

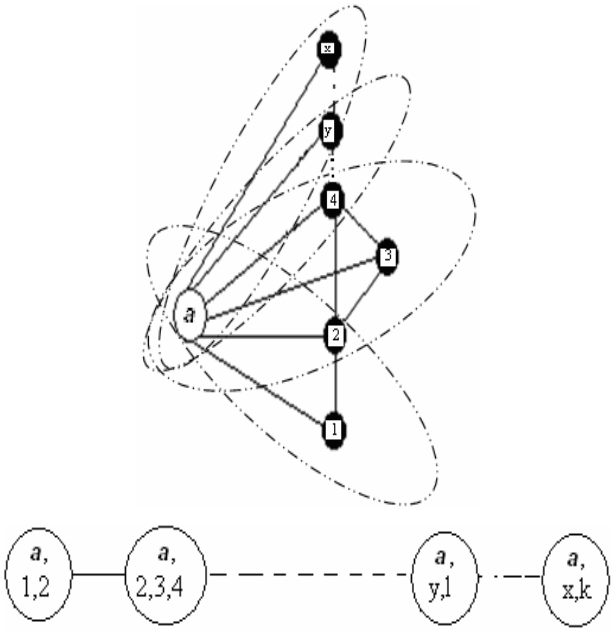


Figure 5.13-Exemple de décomposition arborescente.

On conclu pour le 1er cas que si $\deg_{\max}(G) = NC$ et $G = V(a) \cup \{a\}$ alors on a $tw(G) = 3$.

2ième cas

Si $G = \{a\} \cup V(a) \cup G'$ (G' est l'ensemble des sommets qu'ils ne sont pas adjacent à *a*. Voir la Figure 5.14

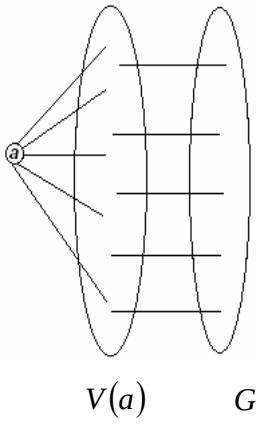


Figure 5.14- 2ième cas

Dans ce cas on a deux cas de figures possibles pour $NC = \deg_{\max}(G)$.

1^er^ Cas de figure

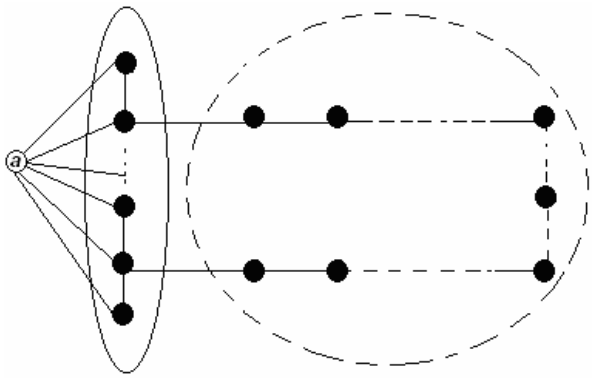


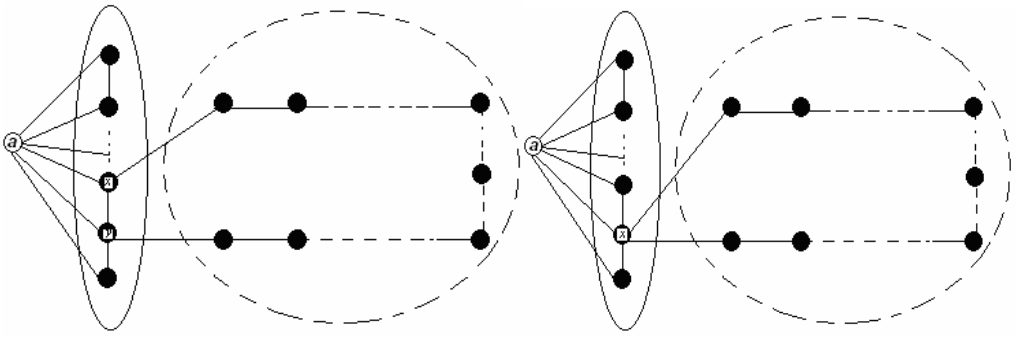
Figure 5.15- Exemple de 1^er^ configuration du 2^ième^ cas de la proposition 3.

Dans ce cas de figure nous avons l'ensemble des voisins de *a* est une chaîne simple, alors on auras $\deg_a - 1$ cycles dans le sous-graphe induit par *a* et l'ensemble de ses voisins.

Nous avons l'ensemble des sommets de $G^{'}$ est une chaîne tel que ses deux extrémités sont adjacents aux sommets de $V(a)$ (posons x et y ces deux sommets).

On distingue ici deux cas :

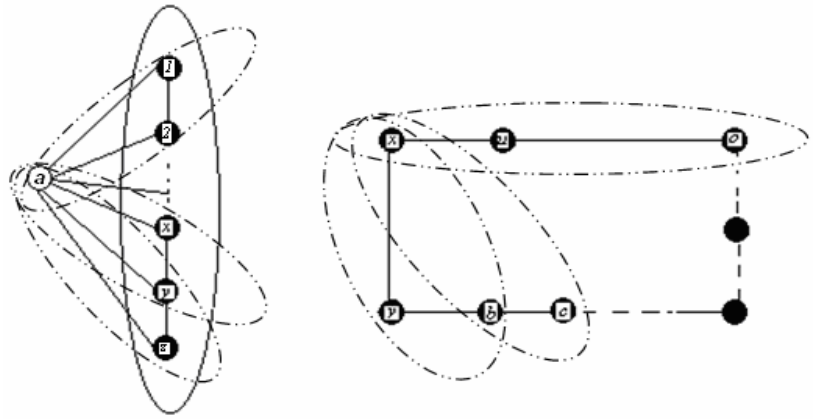
1- ***Cas particuliers*** Si *x*, *y* sont distincts mais, ils sont adjacent. Voir le graphe 'A' de la Figure 5.16.



'A' 'B'

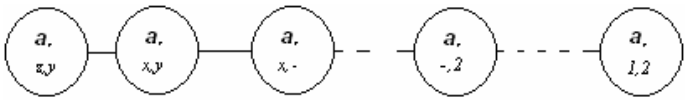
Figure 5.16-cas de Figure 2

La décomposition de ces graphes revient à décomposer leurs composantes connexes 'p' et 'd' et on fusionne les deux décompositions pour avoir la décomposition de 'A' ou de 'B' voir la Figure 5.17.

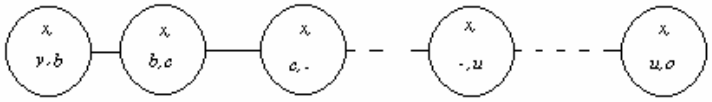


‘p’ **‘d’**

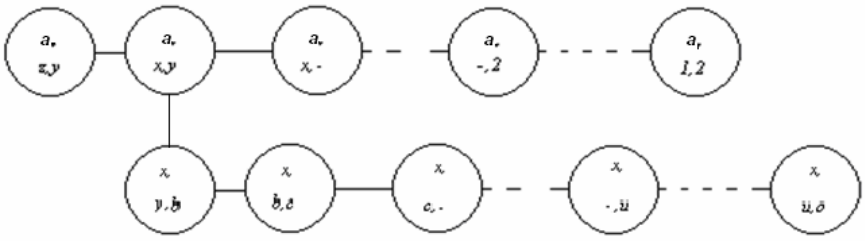
Figure 5.17-Regroupement des deux composantes connexes.



Décomposition arborescente de ‘p’ $tw(p)=2$.



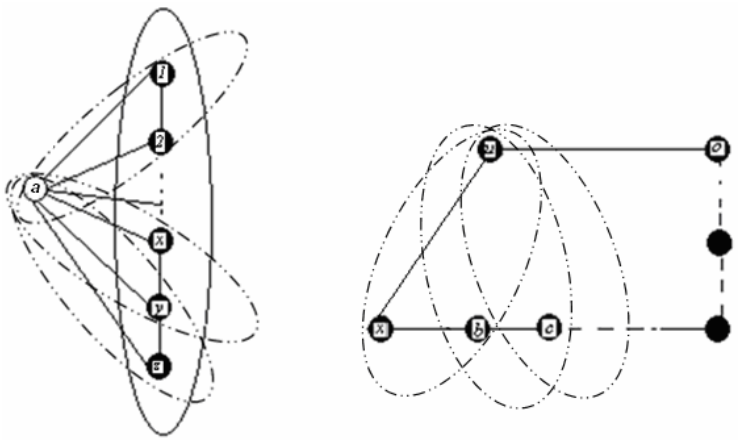
Décomposition arborescente de ‘d’ $tw(d)=2$.



La décomposition arborescente de ‘A’ $tw(A)=2$.

Figure 5.18- Déroulement de la décomposition arborescente de ‘A’

On décompose le graphe ‘d’ de la même manière que celle du graphe ‘p’ la seule différence entre elles, c’est que les sommets *x* et *y* sont confondu. Voir la Figure 5.19.



'p' **'d_1'**

Figure 5.19- Regroupement des sommets.

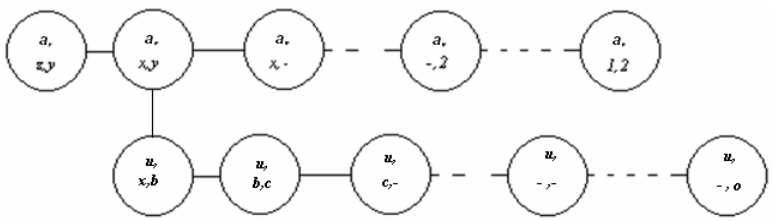


Figure 5.20- Déroulement de la décomposition arborescente de 'B' $tw(b)=2$.

Résumé

Dans les graphes 'A' et 'B' nous avons les sommets $V(a)$ construisent une chaîne simple.

Pour le graphe 'A', l'intersection de $V(a)$ avec la composante 'd' est en deux sommets adjacents *x* et *y*, par ailleurs, pour le graphe 'B' l'intersection de $V(a)$ avec la composante 'd_1' est le point d'articulation *x*.

La décomposition arborescente de 'A' et 'B' en sacs de taille trois on aura $tw(G)=2$.

2- ***Cas général*** Si les sommet *x* et *y* sont non adjacent. Voir la Figure 5.20.

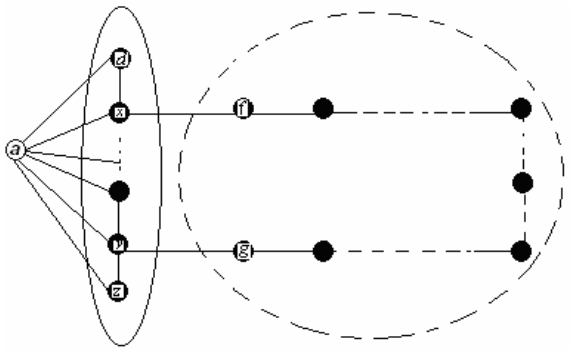
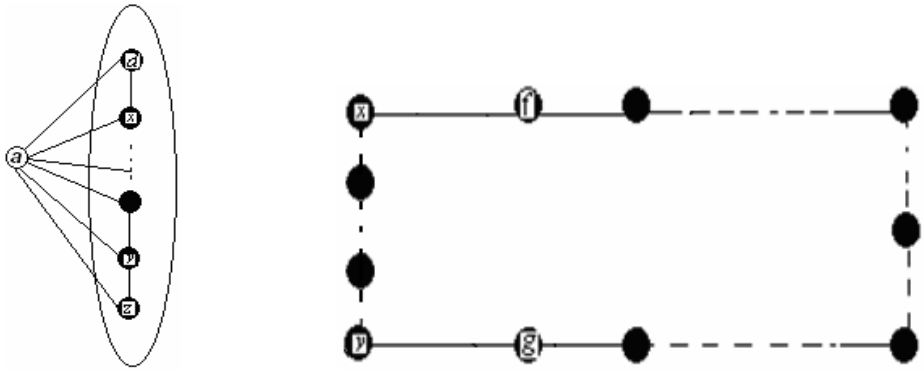


Figure 5.20-Graphe *G* : cas général

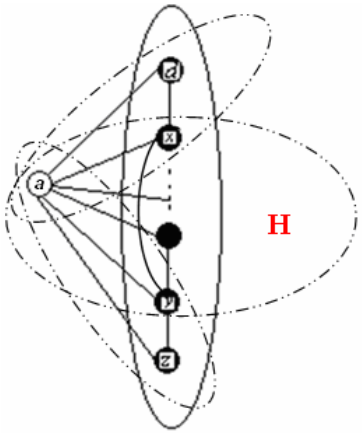
Pour la décomposition du graphe de la Figure 5.20 on commence par la décomposition des deux composantes connexes ‘A’ et ‘B’ de la figure 5.21.



‘A’ ‘B’

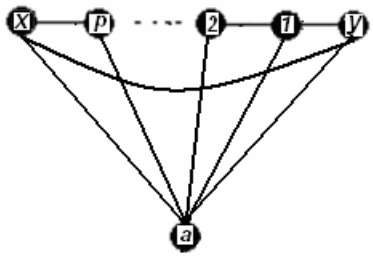
Figure 5.21- Les deux sous graphes du graphe de la figure 5.20

La décomposition de la composante connexe ‘A’ pour ce cas de figure revient a décomposé la composante ‘a1’.

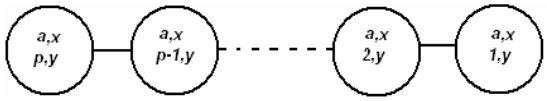


‘a1’

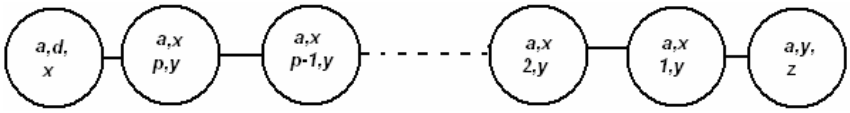
La décomposition de ‘H’



‘H’



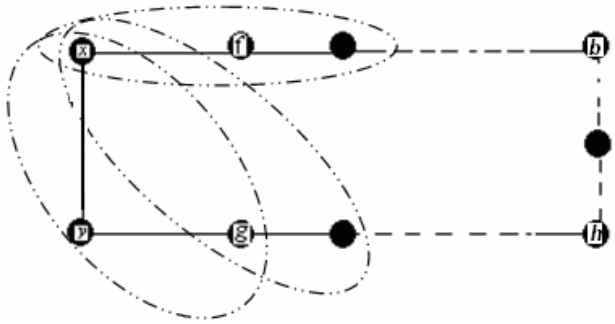
Et pour la décomposition de ‘A’ :



$$tw(A) = 3.$$

- $tw(a1) = tw(A) = 2.$

La décomposition de la composante connexe ‘B’ pour ce cas de figure revient a décomposé la composante ‘b1’.



‘b1’

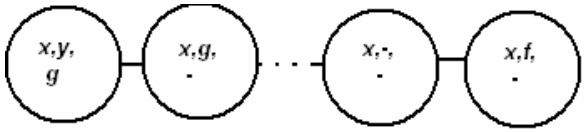


Figure 5.22- Déroulement de la décomposition arborescente du graphe 5.20 1^{ier} partie

$$tw(b1) = 2.$$

- $tw(b1) = tw(B) = 2.$

Décomposition du graphe *G* de la figure 5.20.

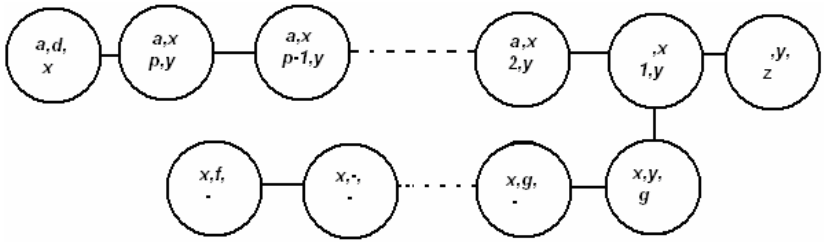


Figure 5.23-Décomposition arborescente du graphe *G*

$$tw(G) = \max(tw(a), tw(b)) = 3.$$

2^{ième} cas de figure

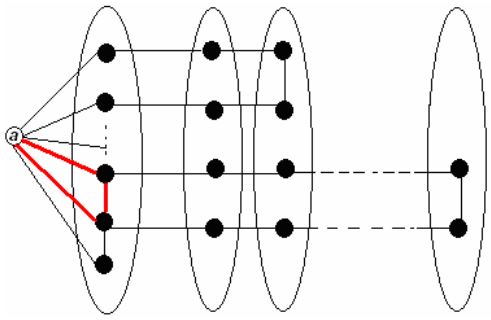


Figure 5.24-Graphe du deuxième cas de figure

Le graphe ci-dessus est un graphe où nous avons seulement un triangle qui est lié à *a*.

Pour la décomposition de ce graphe on doit passer par la triangulation du graphe en suite par le regroupement des sommets en cliques. Voir figure 5.25.

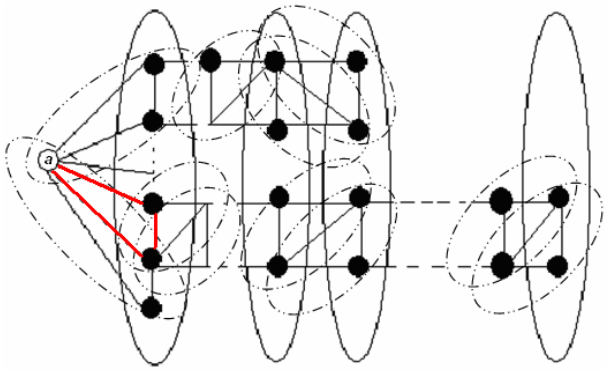


Figure 5.25-Triangulation du graphe *G*

La décomposition arborescente de ce graphe est de la forme suivante :

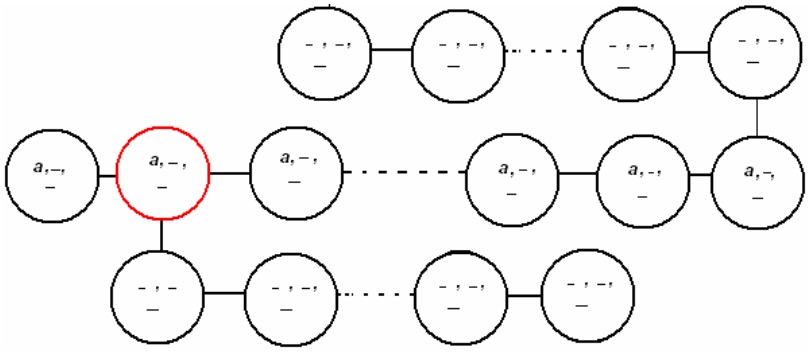


Figure 5.26- Décomposition de graphe du deuxième cas de figure

Chaque sac contient exactement trois sommets alors $tw(G)=2$.

■

Conds.1

Soit *G* un graphe simple et connexe.

- Le graphe *G* admet une chaîne de longueur $p-2$. Supposons que c'est (a_0, a_1, a_2,…, a_{p-2}) ;
- Le nombre de sommets voisin commun entre a_0 et a_1 est égal à $p-2$;
- Si a_i et a_{i+2} sont adjacent alors, si le nombre de sommets voisins communs entre eux est égal à au moins $p-i-2$ sinon, il faut que le nombre de sommets voisins entre les deux soit égal à au moins $p-i-1 \ \forall \ 0 \le i \le p-4$.

Théorème 5.4

Soit $G(V,E)$ un graphe *p*-réglier, tel que $p = \left\lceil \frac{|V|}{2} \right\rceil$, si les conds.1 sont vérifiées alors, la largeur arborescente $tw(G)$ est égal à *p* sinon elle est supérieur ou égale à $p+1$.

□ a_0 et a_1 ont $p-2$ sommet voisin commun $|V(a_0) \cap V(a_1)| = p-2$.

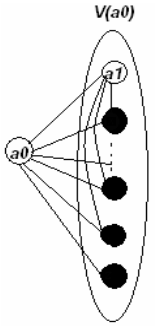


Figure 5.27-Ensemble des voisins de 'a_0'

La configuration simple de ces conditions est donnée par le schéma suivant :

Une chaîne de longueur $p-2$.

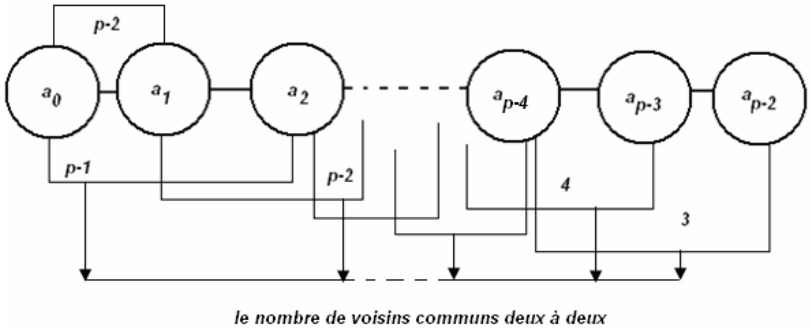


Figure 5.28- Schéma de 'conditions 1'.

Pour la décomposition arborescente du graphe *G* on regroupant les sommets comme suit :

- Le premier sac contient le sommet 'a$_0$' et l'ensemble de ses voisins $\{a_0\} \cup V(a_0)$.

- Le second sac contient l'ensemble des sommets $\{a_1\} \cup \{V(a_1) \setminus \{a_0\}\}$.

- Le troisième sac contient l'ensemble des sommets $\{a_2\} \cup \{V(a_2) \setminus \{a_1\}\}$.

Ainsi de suit jusqu'au sac numéro $p-2$ qui contient l'ensemble des sommets $\{a_{p-2}\} \cup \{V(a_{p-2}) \setminus \{a_{p-1}\}\}$.

La cardinalité de chaque sac est égale à p.

■

Exemple

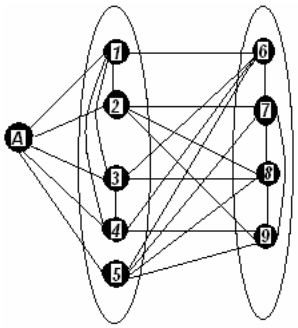
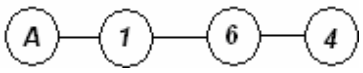


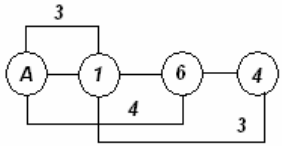
Figure 5.29-Exemple de 'Proposition 4'

Tous les sommets du graphe sont de degré 5 (p=5). $\left\lceil \frac{10}{2} \right\rceil$

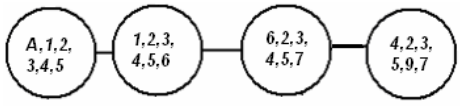
L'enchaînement de la chaîne de largeur $p-1$ est comme suit :



Nombre de sommet voisin commun entre '*A*' et '1' égal à 3 ($p-2$).
Nombre de sommet voisin commun entre '*A*' et '6' égal à 4 ($p-1$).
Nombre de sommet voisin commun entre '1' et '4' égal à 3 ($p-2$).



La décomposition arborescente du graphe *G*.



On a encore le sac qui contient $\{2,3,5,9,7,9\}$ qui sort automatiquement.

La largeur arborescente $tw(G)$ égal à 5.

Conds.2

Soit *G* un graphe simple et connexe.

- Le graphe *G* admet une chaîne de longueur $p-1$. Supposons que c'est (a_1, a_2,…, a_p) ;
- Le nombre de sommets voisins communs entre a_1 et a_2 est égal à $p-1$;
- a_1 est adjacent aux sommets a_3,…, a_{p-1} ;
- a_2 et a_p sont adjacent.

Théorème 5.5

Soit $G(V,E)$ un graphe *p*-régulier, tel que $p < \left\lceil \frac{|V|}{2} \right\rceil$, si les conds.2 sont vérifiées alors, la largeur arborescente $tw(G)$ est égal à p sinon elle est supérieur au égal à $p+1$.

□ a_1 et a_2 ont $p-1$ sommet voisin commun $|V(a_1) \cap V(a_2)| = p-1$.

La configuration simple de ces conditions est donnée par le schéma suivant :
Une chaîne de longueur $p-1$.

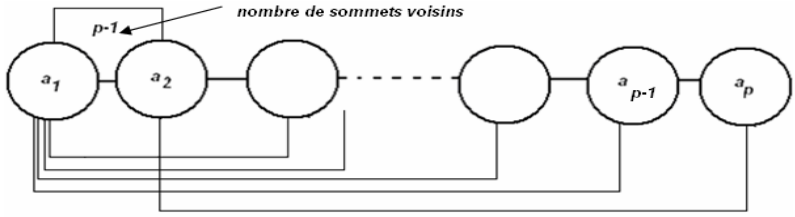


Figure 5.30 – Schéma de 'conditions 2'.

Pour la décomposition arborescente du graphe *G* on regroupant les sommets comme suit :

Le premier sac contient le sommet 'a$_1$' et l'ensemble de ses voisins $\{a_1\} \cup V(a_1)$.

Le second sac contient l'ensemble des sommets $\{a_2\} \cup \{V(a_2) \setminus \{a_1\}\}$.

Le troisième sac contient l'ensemble des sommets $\{a_2\} \cup \{V(a_2) \setminus \{a_1\}\}$.

Ainsi de suit jusqu'au sac numéro p qui contient l'ensemble des sommets $\{a_p\} \cup \{V(a_p) \setminus \{a_{p-1}\}\}$.

La cardinalité de chaque sac est égale à p.

■

Conds.3

Soit *G* le graphe simple et connexe.

- Il existe une chaîne de longueur $n-p-1$; supposons qu'on a (a_1, a_2,…, a_{n-p}) ;
- Le nombre de sommets voisins commun entre a_1 et a_2 est égal à $p-2$;
- Si a_i et a_{i+2} sont adjacents alors, si le nombre de sommets voisins communs entre eux est égal à au moins $p-i-1$ sinon, il faut que le nombre de sommets voisins entre les deux soit égal à au moins $p-i$ $\forall\ 1 \le i \le n-p-2$.

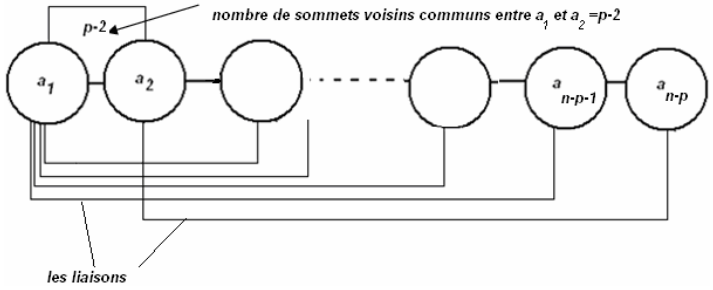
Théorème 5.6

Soit $G(V,E)$ un graphe *p*-régulier, tel que $p > \left\lceil \frac{|V|}{2} \right\rceil$, si les conds.3 sont vérifiées alors, la largeur arborescente $tw(G)$ est égal à p sinon elle est supérieur au égal à $p+1$.

□ a_1 et a_2 ont $p-2$ sommet voisin commun $|V(a_1) \cap V(a_2)| = p-2$.

La configuration simple de ces conditions est donnée par le schéma suivant :

Une chaîne de longueur $n-p-1$.



Pour la décomposition arborescente du graphe *G* on fait le regroupement des sommets comme suit

Le premier sac contient le sommet 'a$_1$' et l'ensemble de ses voisins $V(a_1) \cup \{a_1\}$.

Le second sac contient l'ensemble des sommets $\{a_2\} \cup \{V(a_2) \setminus \{a_1\}\}$.

Le troisième sac contient l'ensemble des sommets $\{a_2\} \cup \{V(a_2) \setminus \{a_1\}\}$.

Ainsi de suit jusqu'au sac numéro $n-p$ qui contient l'ensemble des sommets $\{a_{n-p}\} \cup \{V(a_{n-p}) \setminus \{a_{n-p-1}\}\}$.

La cardinalité de chaque sac est égale à p.

■

Conds.4

Soit *G* un graphe simple et connexe.

- Le graphe *G* admet chaîne de longueur $|V|-4$ (*n*-4). Supposons que c'est (a_1, a_2,…, a_{n-3}) ;
- Le sommet a_1 est adjacent au sommet a_3 ;
- Le sommet a_i est adjacent au sommet a_{i+3} tel que i est un indice pair $i>1$.

La représentation de ces conditions est donnée par le schéma suivant :

Une chaîne de longueur $p-4$.

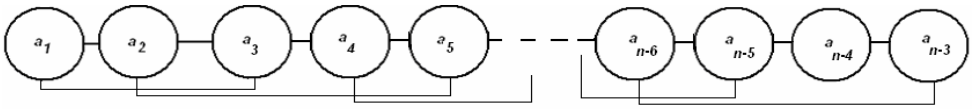


Figure 5.31 – Schéma de 'conditions 4'.

Théorème 5.7

Soit un graphe $G(V,E)$, si $(|V|\geq 8)$ et $(\forall\ x\in V,\ \deg_x = 3)$ alors

si $(\text{les conds.4 sont vérifiées})$ alors $tw(G)=3$
sinon $tw(G)=4$.

□ 1. Le cas où les conditions 4 sont vérifiées.

Algorithme 5.2

entrée : $G(V,E)$ // Un graphe simple et connexe;

sortie : // Décomposition du graphe *G* qui vérifié conds.4.

début

pour *i* :=1 **jusqu'a** *n*-3 **faire**

$sac_i \leftarrow N_G(a_i)\cup\{a_i\}$ Tel que $N_G(a_i)$ est l'ensemble de voisins de a_i dans le graphe *G* ;

$G \leftarrow G \setminus \{a_i\}$ **;**

fait ;

fin.

Complexité de l'algorithme 5.2

L'algorithme est linéaire d'ordre $O(n)$.

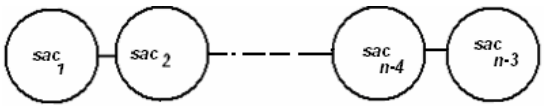


Figure 5.32- Présentation du premier cas

La taille du premier sac égal à 4 et a chaque itération de l'algorithme on a *n* sommets qui rentre et un autre sommet qui sort alors on aura $tw(G)=3$.

2. Le cas ou les conditions 4 ne sont pas vérifiées alors :
Si la deuxième ou bien la troisième condition n'est pas vérifiée alors on au moins un sac de cardinalité 5 ce qui implique $tw(G)=4$ parce que à chaque itération de l'algorithme on a au maximum un sommet qui rentre dans un sac et un autre qui sort.
On donne l'exemple de la figure suivante :

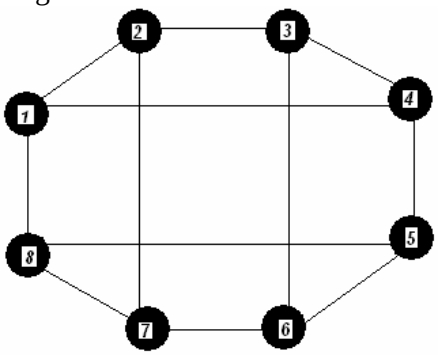


Figure 5.33-Exemple du deuxième cas

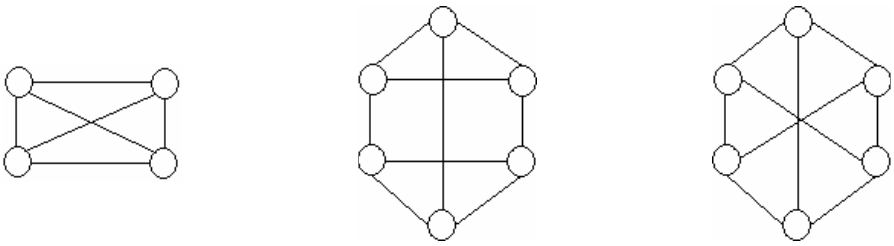
k=4 ;
$tw(G)=4$;

■

Remarque

Si le nombre de sommets de *G* est inférieur à 7 on a $tw(G)=3$.

□ On a dans ce cas la seulement 3 cas possibles.



Dans les trois cas $tw(G)=3.$

■

Théorème 5.8

Etant donné un graphe $G(V,E)$, si $\exists\ x\in V$ tel que $\deg_x = n-1$ et $\forall\ y\in V\setminus\{x\}$, $\deg_y = 3$ alors $tw(G)=3.$

□ Par construction, si le graphe *G* est un graphe triangulé alors on aura $tw(G)=3.$ la clique maximale est de taille 4 voir figure 5.34.

G_1 G_2

Figure 5.34- Exemple du théorème 5.7

Le graphe triangulé G_1 est composé de deux cliques de cardinalité égale à '4'.
Le graphe G_2 est une clique de cardinalité '4' sa décomposition c'est elle-même. K_4 représente le plus petit graphe vérifiant le théorème.

Si *G* n'est pas triangulé, la décomposition arborescente du graphe *G* consiste à regrouper les sommets dans des sacs de taille quatre, où deux sacs consécutifs différents d'un seul sommet. Voir la figure 5.35.

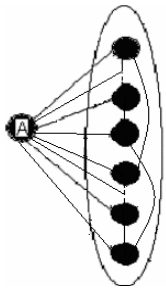
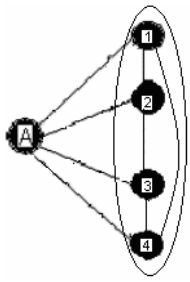


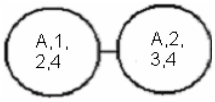
Figure 5.35- Cas général

Le graphe de la figure de 5.35 n'est pas triangulé, le regroupement en sacs de ses sommets se fait de telle sorte que chaque sac soit de taille quatre pour mieux illustrer cette proposition on donne l'exemple suivant.

Exemple



La décomposition arborescente de ce graphe est donnée par :



Chaque sac est composé de quatre sommets, ce qui implique que la largeur arborescente du graphe est égale à '3'.

■

Notations

N_i : Ensemble de sommets du niveau *i.*

N(a) : Ensemble de voisins du sommet *a.*

Algorithme 5.3

entrée : $G=(V,E)$ // graphe planaire,

sortie // Dessiner *G* par niveau.

début

$V' \leftarrow V$;

Choisir un sommet *a* de degré minimum.

$N_1 \leftarrow \{a\}$;

$nb_n \leftarrow 2$;

tant que $V' \neq \phi$ **faire**

$N_{nb_n} \leftarrow$ Ensemble des voisins de N_{nb_n-1} dans *G*.

$V' :\leftarrow V' \backslash N_{nb_n}$.

$nb_n \leftarrow nb_n + 1$;

fait

$nb_n \leftarrow nb_n - 1$;

fin

Complexité de l'algorithme 5.3

L'algorithme est linéaire d'ordre $O(n)$.

nb_n désigne le nombre de niveaux du dessin de graphe *G*.

Algorithme 5.4

entrée G' **//** Dessin du graphe *G* par l'algorithme 5.3

sortie //Décomposition du graphe *G*

début

Construire $sac_1 \leftarrow \{a\} \cup N\{a\}$;

$G' :\leftarrow G' \setminus \{a\}$ **;**

pour t :=2 **à** nb_n **faire**

Compléter le niveau N_t en clique $K_{|N_t|}$ **;**

tant que $N_t \neq \phi$ **faire**

Choisir le sommet *b* tel que : $b \leftarrow$ le sommet de degré minimum N_t ;

$sac_t \leftarrow \{b\} \cup N(b)$ **;**

$N_t \leftarrow N_t \setminus \{b\}$ **;**

fait ;

$t \leftarrow t+1$ **;**

fait;

fin.

Complexité de l'algorithme 5.4

L'algorithme est hyper-linéaire d'ordre $O(nb_n n)$, sachant que nb_n est le nombre de niveaux du dessin du graphe *G* avec l'algorithme 5.3.

Notations

$\Re_l = \left\{ x / \forall\ x, y \in N_l \Rightarrow xy \notin E,\ \forall\ z \in N_{l-1} \Rightarrow xz \in E\ et\ \forall\ d \in N_{l+1} \Rightarrow xd \notin E \right\}$

N^* Désigne l'ensemble des entiers naturels non nul.

Proposition 5.9

Etant donné le graphe planaire, $G(V,E)$, sa largeur arborescente $tw(G)$ est :

$$tw(G) = \begin{cases} \max\limits_{2 \le l \le nb_n} \left(|N_l \setminus \Re_l| \right) + 1 & \textbf{si } \exists\ l \ge 2 \text{ tel que} \\ & |N_l \setminus \Re_l| = |N_{l+1} \setminus \Re_{l+1}| \ge |N_k \setminus \Re_k| \ \ pour\ \ k \in \left\{ \mathrm{N}^* \setminus \{1, l, l+1\} \right\} \\ & \forall\ x \in N_l \setminus \Re_l,\ \exists\ y, z \in N_{l+1} \setminus \Re_{l+1}\ tel\ que\ xy, xz \in E \\ \max\limits_{2 \le l \le nb_n} \left(|N_l \setminus \Re_l| \right) & \textbf{sinon} \end{cases}$$

□ *G* est un graphe planaire, d'après Kuratowski un graphe est planaire si et seulement si n'admet pas K_5 et $K_{3,3}$ comme graphe induit. On utilisant l'algorithme 5.3 pour dessiner le graphe *G*.

Sachant que à chaque itération de la décomposition arborescente on complète un niveau par des arêtes de tel sort d'avoir une clique à chaque niveau.

Si $\exists\ l \ge 2$ tel que $|N_l \setminus \Re_l| = |N_{l+1} \setminus \Re_{l+1}| \ge |N_k \setminus \Re_k|$ pour $k \in \left\{ \mathrm{N}^* \setminus \{1, l, l+1\} \right\}$ et $\forall\ x \in N_l \setminus \Re_l$, $\exists\ y, z \in N_{l+1} \setminus \Re_{l+1}$ tel que $xy, xz \in E$. Dans ce cas de figure nous avons deux niveaux consécutifs, *l* et *l*+1, de cardinalité maximum où chaque sommet du niveau *l* est adjacent à deux sommets du niveau *l*+1 alors le degré des sommets simplicials du niveau *l* est égal à $\left(|N_l \setminus \Re_l| \right) + 1$ ce qui implique qu $tw(G) = \max\limits_{2 \le l \le nb_n} \left(|N_l \setminus \Re_l| \right) + 1$.

Dans le deuxième cas où nous avons un niveau l_1 de cardinalité maximum qui peut être adjacent à un niveau de cardinalité inférieure au égale à $|N_{l_1}|$, si chaque sommet du niveau l_1 est adjacent à au plus un sommet du niveau l_1+1 alors pour que les sommets de N_{l_1} deviennent simplicials le degré de ces sommets vont être égaux à $|N_l \setminus \Re_l|$

■

Notation $\nabla_x(l)$ désigne le nombre de sommets dans *l* qui sont adjacents au sommet *x* tel que $x \in (l-1)$

Proposition 5.10

Soit un graphe $G(V,E)$, non planaire tel que $nb_n \geq 3$ sa la largeur arborescente $tw(G) \geq \max_{2 \leq l \leq nb_n} \left(\min_{x \in N_l} \left(|N_l| + \nabla_x (l+1) - 1 \right) \right)$.

□ Nous décomposons le graphe *G*, modélisant le problème en question, à l'aide de l'algorithme 5.4 en sacs. La cardinalité de ceux-ci vérifiée l'inégalité suivante :

$\forall\ x \in V,\ x \in S_j\ \ pour\ \ j = \overline{1, nb_n}\ \ |S_j| \geq \deg_x$.

Considérons un niveau *l*, $\forall\ a \in N_l$. L'utilisation de l'algorithme de Berry, pour la détermination de sommet simplicial, nécessite l'ajout des arêtes au graphe *G* si *a* est non simplicial. Donc le sommet *a*, simplicial, sera de degré $|N_l| + \nabla_a (l+1) - 1$ et la décomposition arborescente de ce nouveau graphe consiste de choisir à itération de l'algorithme 5.4 le sommet de degré minimum dans chaque niveau qui est le sommet de degré $\min_{x \in N_l} \left(|N_l| + \nabla_x (l+1) - 1 \right)$ ce qui signifie que :

$tw(G) \geq \max_{2 \leq l \leq nb_n} \left(\min_{x \in N_l} \left(|N_l| + \nabla_x (l+1) - 1 \right) \right)$.

■

Algorithme 5.5

entrée $G(V,E)$ simple et connexe;

V : l'ensemble des sommets du graphe ;
$|V| = n$;

E : l'ensemble des arêtes du graphe.

NC : le nombre de cycles élémentaire de *G*.
$\deg_{\min}$: Le degré minimum du graphe *G*.
$\deg_{\max}$: Le degré maximum du graphe *G*.

k : la clique maximale dans *G*.

cord : la cordalité du graphe *G*.

sortie
$tw(G)$ **:** la largeur arborescente.

début

$G := G \setminus \ell$; tel que ℓ est l'ensemble des sommets pendants.

si $n-1$ sommets sont de degré trois et on a un sommet de degré *n* **alors** $tw(G) = 3$;
Aller à **Fin** ;
fsi ;

si $NC = \deg_{\max}$ **alors**
si le graphe *G* est un cycle **alors** $tw(G) = 2$

sinon $2 \le tw(G) \le 3$ **;**

fsi ;

Aller à Fin ;

fsi ;

si tous les cycles de G sont de longueur trois

alors ***G*** est un graphe triangulé et $tw(G) = |k|$;

Aller à **Fin** ;

fsi ;

si tous les sommets de *G* sont de même degré *p* **alors**

si $p = 3$ **alors**

si les conditions 4 sont vérifiées **alors** $tw(G) = 3$

sinon $tw(G) = 4$ **;**

fsi ;

sinon

si $p = \left\lceil \frac{|V|}{2} \right\rceil$ **alors si** les conditions 1 sont vérifiées **alors** $tw(G) = p$

sinon $tw(G) \ge p + 1$ **;**

fsi ;

fsi ;

si $p > \left\lceil \frac{|V|}{2} \right\rceil$ **alors**

si les conditions 3 sont vérifiées **alors** $tw(G) = p$

sinon $tw(G) \ge p + 1$ **;**

fsi;

fsi ;

si $p < \left\lceil \frac{|V|}{2} \right\rceil$ **alors**

si les conditions 2 sont vérifiées **alors** $tw(G) = p$

sinon $tw(G) \ge p + 1$ **;**

fsi;

fsi ;

Aller à **Fin** ;

fsi ;

fsi ;

si *G* **est** planaire **alors**

$$tw(G) = \begin{cases} \max\limits_{2 \le l \le nb_n} \left(|N_l \setminus \Re_l| \right) + 1 & \textbf{si } \exists\ l \ge 2 \text{ tel que} \\ & |N_l \setminus \Re_l| = |N_{l+1} \setminus \Re_{l+1}| \ge |N_k \setminus \Re_k| \ \ pour\ \ k \in \left\{ \mathbb{N}^* \setminus \{1, l, l+1\} \right\} \\ & \forall\ x \in N_l \setminus \Re_l,\ \exists\ y, z \in N_{l+1} \setminus \Re_{l+1}\ tel\ que\ xy, xz \in E \\ \max\limits_{2 \le l \le nb_n} \left(|N_l \setminus \Re_l| \right) & \textbf{sinon} \end{cases}$$

Sinon $tw(G) \geq \max_{2 \leq l \leq nb_n} \left(\min_{x \in N_l} \left(|N_l| + \nabla_x (l+1) - 1 \right) \right)$

fsi ;

Fin.

La complexité l'algorithme 5.5

L'algorithme est polynomial d'ordre $O(n)$.

Dans la proposition qui suit on montre la relation entre l'invariant lié à la décomposition arborescente, la largeur arborescente (*tw*(*G*)) et le paramètre lié à la décomposition en branches, la frontière (*F*).

Proposition 5.11
La taille de la frontière d'un graphe *G* est égale à sa largeur arborescente, $tw(G) \geq |F|$.

□ En appliquant l'algorithme 5.4 on a chaque niveau correspond a un ensemble de séparateurs et en plus chaque niveau peut partitionner le graphe en deux composantes alors, on peut considérer les sommets des niveaux comme des frontières

■

La deuxième partie est consacrée au calcul de la longueur arborescente. Nous avons pu déterminé une borne plus serrée par rapport a celle donné par Berry au théorème 3.20 ; un graphe *k*-cordal est de longueur arborescente $\left\lfloor \frac{k}{2} \right\rfloor$.

5.2 Calcul de longueur arborescente

Définition 5.12 (bloc)

Un bloc est un sous ensemble de sommets qui contient les sommets de deux niveaux consécutifs.

Notation

$bloc_l$: désigne le bloc numéro *l*.

Proposition 5.13

Soit $G(V,E)$ un graphe, si *G* est un *k*-cordal alors la longueur arborescente du graphe est bornée par $\max_{1 \leq l \leq nb_n - 1} \left(\min_{u,\ v \in bloc_l} (dist(u,v)) \right)$.

□ Pour illustrer la proposition 5.13 on donne les exemples suivants.

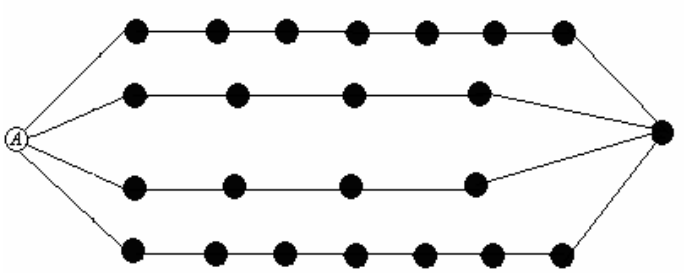
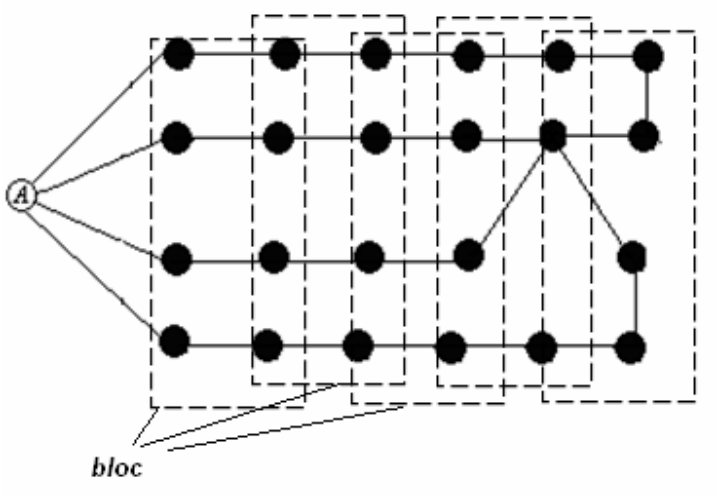


Figure 5.36- Exemple '1' le graphe *G* est de cordalité *k*=16.

D'après Berry la longueur arborescente de ce graphe est borné par $\left\lfloor \frac{k}{2} \right\rfloor$, alors on aura pour l'exemple '1' $tl(G) \leq 8$.



Et en appliquant l'algorithme 5.4 on trouve $tl(G) \leq 5$.

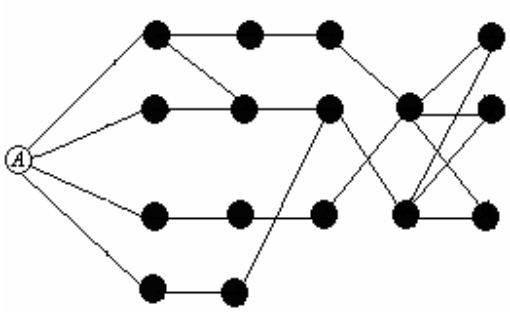


Figure 5.37- Exemple '2' le graphe *G* est de cordalité *k*=10.
D'après Berry $tl(G) \leq 5$.
en appliquant l'algorithme 5.4 $tl(G) \leq 5$

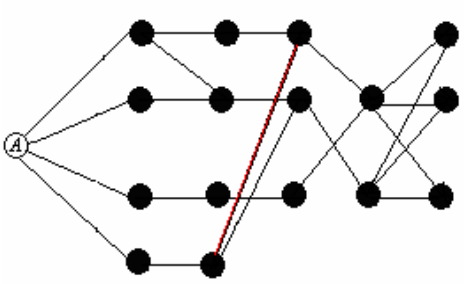


Figure 5.38- Exemple '3' le graphe *G* est de cordalité *k*=10.
D'après Berry $tl(G) \leq 5$.
en appliquant l'algorithme 5.4 $tl(G) \leq 4$

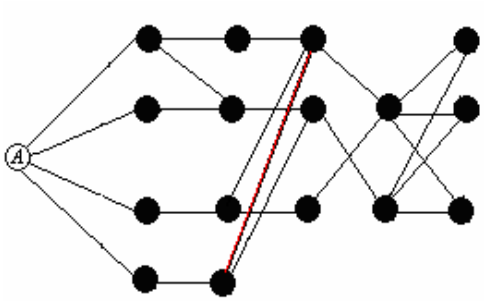


Figure 5.38- Exemple '4' le graphe *G* est de cordalité *k*=10.
D'après Berry $tl(G) \leq 5$.
en appliquant l'algorithme 5.4 $tl(G) \leq 3$

On remarque qu'à chaque fois, la longueur arborescente donnée par l'algorithme 5.4 est inférieure ou égale à celle donnée par Berry.

■

Remarque importante

Pour le calcul des deux invariants, la largeur et la longueur arborescente nous utilisons l'algorithme polynomial que nous avons développé (algo.5.4), ce qui signifie que nous avons pu minimisé au même temps la longueur et la largeur arborescente.

5.3 Calcule de la largeur de branches

L'intégration de nos résultats dans le théorème de Robertson et Seymour [RS91] (voir théorème 3.31) nous a conduit à déterminer la valeur de la largeur de branches.

Corollaire 5.14

Si le $\deg_{max}(G)$ est égale au nombre de cycles dans *G* alors la largeur de branches $bw(G)$ est comprise entre $2 \leq bw(G) \leq 4$.

□ D'après Robertson et Seymour [RS91] nous avons :
$bw(G) \leq tw(G) + 1 \leq \lfloor 3bw(G) / 2 \rfloor$ et le théorème 5.3 on aura $2 \leq bw(G) \leq 4$

■

Corollaire 5.15

Soit un graphe $G(V,E)$, si $(|V| \geq 8)$ et $(\forall\ x \in V\ \ \deg_x = 3)$ alors

si $(\text{les conds.4 sont vérifiées})$ alors $3 \leq bw(G) \leq 4$

sinon $4 \leq bw(G) \leq 5$.

□ D'après Robertson et Seymour [RS91] nous avons
$bw(G) \leq tw(G) + 1 \leq \lfloor 3bw(G) / 2 \rfloor$ et le théorème 5.7 on aura $4 \leq bw(G) \leq 5$

■

Corollaire 5.16

Etant donné un graphe $G(V,E)$, si $\exists\ x \in V$ tel que $\deg_x = n-1$ et $\forall\ y \in V \setminus \{x\}$, $\deg_y = 3$ alors $tw(G) = 3$.

Soit le graphe $G = (V,E)$, si $|V|-1$ sommets sont de degré trois et on a un sommet de degré $|V|-1$ alors $3 \leq bw(G) \leq 4$

□ D'après Robertson et Seymour [RS91] on a $bw(G) \leq tw(G) + 1 \leq \lfloor 3bw(G) / 2 \rfloor$.
et le théorème 5.8 on aura $3 \leq bw(G) \leq 4$.

■

5.4 Application à quelques classes de graphes

On applique les résultats obtenus sur quelques graphes.

5.4.1 Les graphes scindés

Définition 5.17 Un graphe scindé (split-graph) est un graphe *G = (V,E)* dont l'ensemble des sommets admet une partition (*S,K*), o`u *S* est un stable et *K* est une clique (voir figure 5.39).

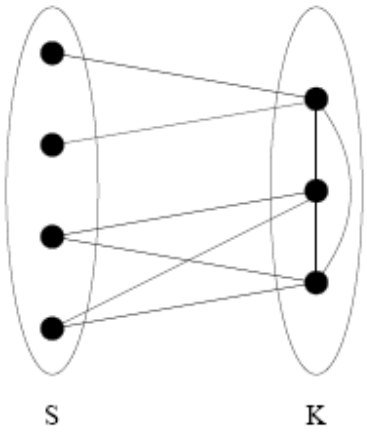


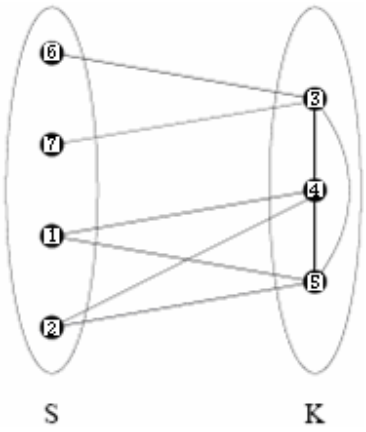
Figure 5.39 - Graphe scindé

Le diamètre de *G* est bornée par : $2 \leq diam(G) \leq 3$.

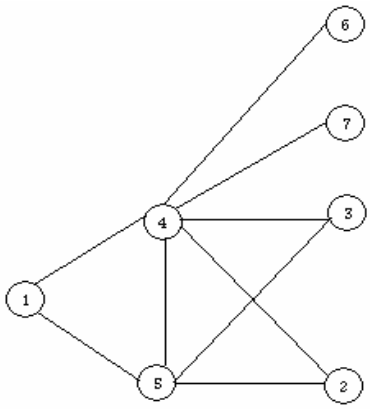
Un graphe scindé appartient à la classe des graphes de cordalité, il est de cordalité *k*=3 (tous les cycles élémentaires supérieurs à quatre possèdent une corde).

Par définition les graphes scindés sont de largeur arborescente égale à $|K|-1$. On peut le vérifier en utilisant l'algorithme 5.4.

En appliquant l'algorithme 5.4 sur le graphe de la figure 5.39

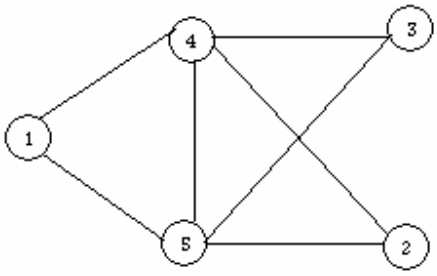


1. Dessiner le graphe par niveaux→



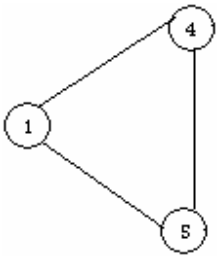
on a trois niveaux $N_1 = \{1\},\ N_2 = \{4,\ 5\},\ N_3 = \{6,\ 7,\ 3,\ 2\}$.

2. Eliminer les sommets pendants→



$\ell = \{6,\ 7\}$.

3. Eliminer les sommets ayants leurs voisins dans le niveau avant→



$\Re_3 = 2$. (Les sommets sont le 2 et 3).

On peu vérifier facilement que $tw(G) = 2$ **et** $tl(G) = 1$**.**

5.4.2 Les graphes à seuils

Définition 5.18 Un graphe à seuil est un graphe scindé dans lequel les voisinages des sommets du stable sont imbriqués (voir figure 6.40).

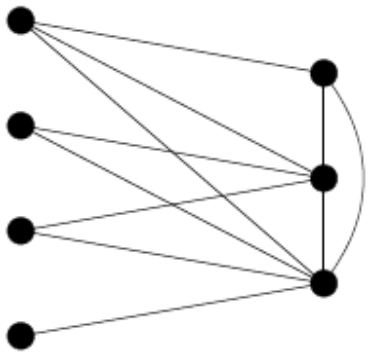


Figure 6.40 – Graphe à seuil

Le diamètre de *G* est bornée par : $diam(G) = 2$.

Le graphe à seuil est de cordalité *k*=3 (tous les cycles élémentaires supérieur à quatre possèdent une corde).

La largeur arborescente de largeur égal à $|K|$ (on a au moins un sommet qui est de degré égal à la cardinalité de la clique *K*, il forme avec cette clique une clique de cardinalité égal à *K*+1) et de longueur arborescente égale à 1.

Conclusion Générale

Construire un algorithme pour réaliser un schéma de télécommunications comme une diffusion (un noeud envoie un message vers tous les autres noeuds), un échange total (tous les noeuds doivent échanger des messages) dans un type de réseau donné (optique, sans fil,...) correspond à déterminer s'il existe un algorithme dans une classe donnée pour le résoudre. On cherche par exemple un algorithme polynomial pour réaliser une diffusion sur une topologie correspondante dans un graphe simple.

Une démarche usuelle utilisée dans les graphes est d'être ramené à l'étude de l'arbre. Pour ce passage certaines conditions sont mises en évidence et pour le retour on doit calculer les paramètres liés à la décomposition.

Dans cette thèse, nous nous sommes intéressés à deux types de décompositions de graphes introduites par Robertson et Seymour ; la décomposition arborescente et la décomposition en branches. À ces dernières se sont associés trois paramètres des graphes ; la largeur et la longueur arborescente et la largeur de branches, qu'ils sont nécessaires à la réalisation de schéma de routage pour les graphes.
Nous avons présenté les résultats obtenus pour le calcul des invariants cités auparavant et nous avons proposé, d'une part, un algorithme efficace pour le calcul de la largeur arborescente pour les graphes *k*-cordaux ; dans des cas particuliers on a trouvé la valeur exacte et une borne inférieur pour d'autres. L'intégration de ce résultat dans le théorème de Robertson et Seymour nous a conduit à déterminer la valeur de la largeur de branches, ainsi nous avons déterminé une borne plus serrée pour le calcul de la longueur arborescente de graphe planaire par rapport à celle donnée par Berry. D'autre part, nous avons montré la relation entre la largeur arborescente et la valeur de la frontière de la décomposition en branches.

Perspectives

Ainsi des recherches pourraient être menées afin de répondre à ces questions : « le calcul de la longueur arborescente d'un graphe quelconque », est-il un problème NP-complet ?
Peut-on déterminer la longueur de branche pour la décomposition en branches ? et qu'elle est sa relation avec la longueur arborescente ?
Peut-on généraliser l'algorithme 6.5 pour d'autres classes de graphes ?

Résumé

Le problème qui se pose dans le routage en général est de trouver une route d'une extrémité à une autre à travers une succession de relais interconnectés. Ce genre de problème porte le nom de routage même les algorithmes et protocoles permettant de résoudre sont également dits de routage.

Le routage consiste à faire communiquer, de la façon la plus efficace possible ces composants. Les liens de communications sont bidirectionnels. C'est pourquoi, on peut modéliser le plan du réseau sous forme d'un graphe simple, connexe et non orienter tel que les sommets sont les machines et les arêtes représentent les liens de communications.

Dans le problème de routage, les informations ne circulent pas de manière homogène dans tout le réseau. En effet, un réseau est souvent décomposé en zones différentes. Ceci fait apparaître les notions de séparateur et de décomposition des graphes. Il existe plusieurs types de décompositions de graphes mais les décompositions les plus intéressantes sont de types arborescentes ayant comme invariants d'études la largeur arborescente et la longueur arborescente et la décomposition en branches celles-ci à comme invariant d'étude la largeur de branches.

Il a été constaté assez tôt que certains problèmes réputés difficiles sont résolubles pour des graphes de largeur arborescente petite. Beaucoup de problèmes classiques de l'algorithmique de graphes, qui sont NP-difficiles en général, peuvent être résolus en temps polynomial pour les graphes ayant une largeur arborescente bornée par une constante. Plusieurs problèmes on été montré ouvert pour la recherche de largeur arborescente, alors que le problèmes devient polynomial pour la recherche de largeur de branches, par exemple le graphe planaire.

Pour notre objectif on se propose d'étudier certaines invariants cités précédemment pour la détermination d'un schéma de routage valide pour des classes de graphes non encore étudié.

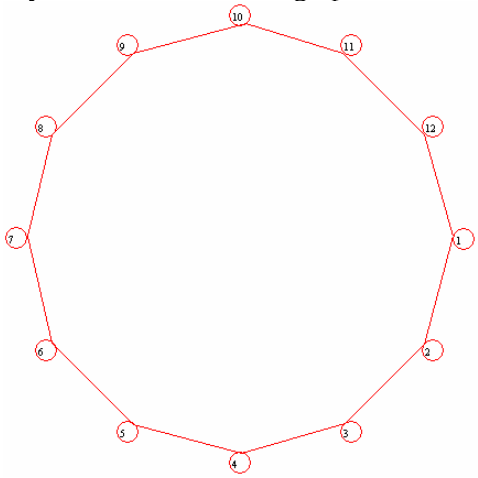
Nous présentons les résultats obtenus pour le calcul des invariants cités auparavant et nous proposons, d'une part, un algorithme efficace pour le calcul de la largeur arborescente pour les graphes *k*-cordaux et les graphes planaires ; pour des cas particuliers on a trouvé la valeur exacte et une borne inférieur pour d'autres. L'intégration de ce résultat dans le théorème de Robertson et Seymour nous a conduit à déterminer la valeur de la largeur de branches, ainsi nous avons déterminé une borne plus serrée pour le calcul de la longueur arborescente de graphe planaire par rapport à celle donnée par Berry. D'autre part, nous avons montré la relation entre la largeur arborescente et la valeur de la frontière de la décomposition en branches.

Mot clé : Routage, Décomposition Arborescente, Décompositions en Branches, Largeur Arborescente, Longueur Arborescente, Largeur de Branches.

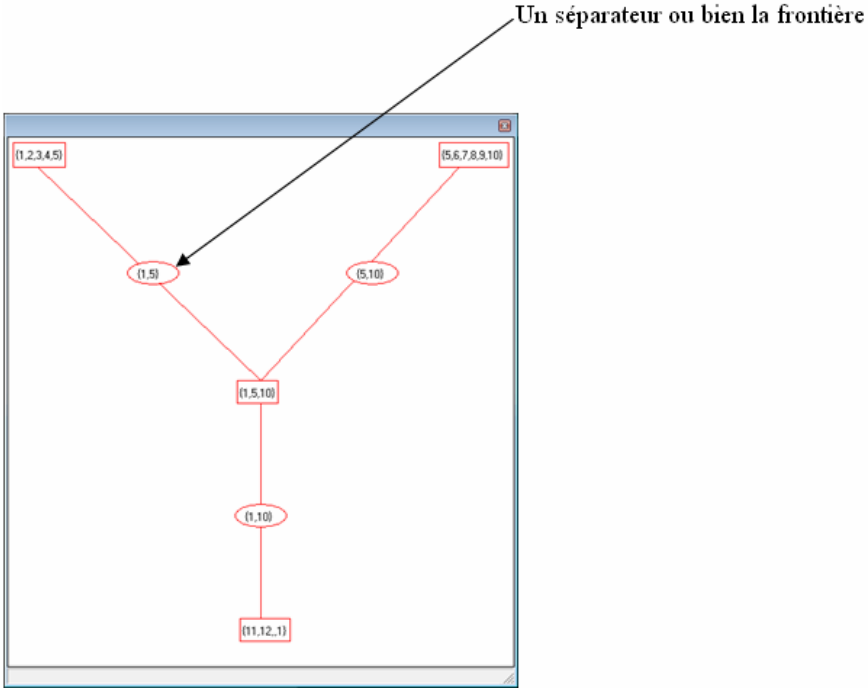
Annexe

1. Instances d'application de notre algorithme 5.4

Lors de l'exécution du logiciel, on peut avoir valeurs suivantes : la cordalité du graphe, la longueur arborescente, la largeur arborescente, la taille de la frontière et on peut avoir aussi tous les cycles élémentaires du graphe.

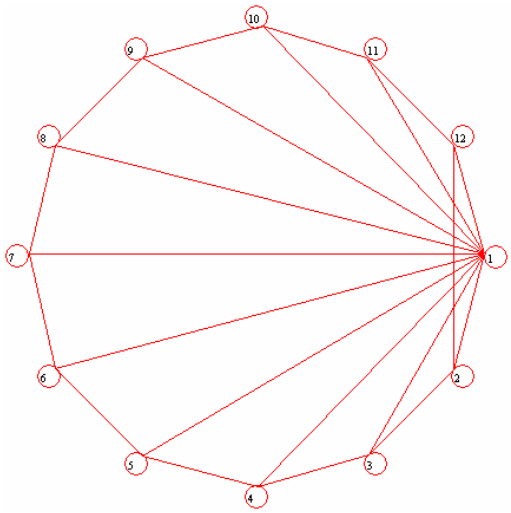


Exemple 1 Le graphe *G* un cycle

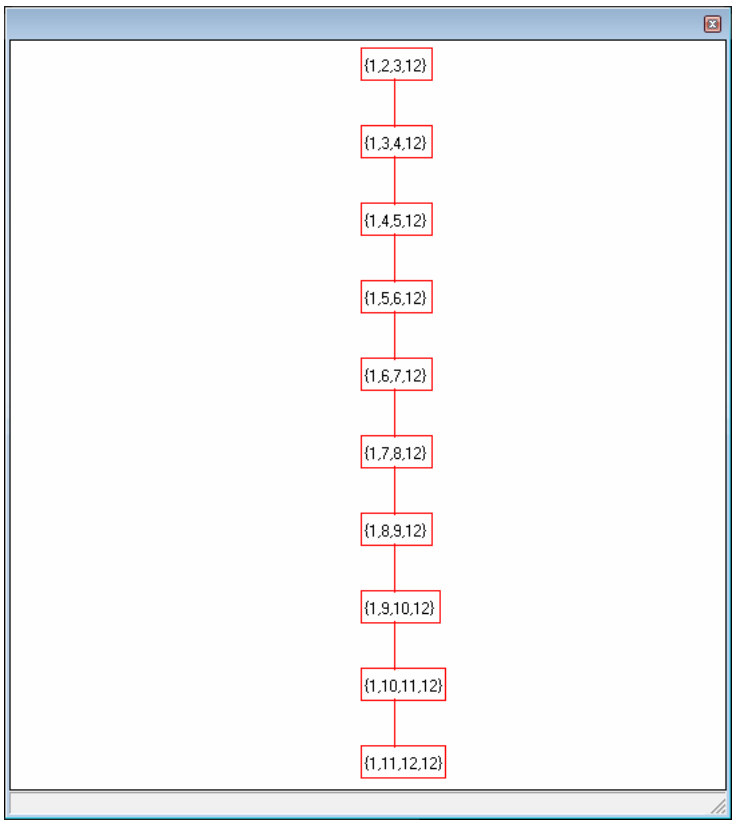


L'arbre '1'

Lors de l'exécution de notre logiciel nous avons trouvé la valeur de cordalité k=12, nous avons un seul cycle élémentaire qui est C=1,2,3,4,5,6,7,8,9,10,11,12 et pour la décomposition arborescente donnée par l'arbre '1' du graphe G, les invariant de la décomposition arborescente $tw(G)=5$, $tl(G)=5$ et la taille de la frontière $|F|=2$.



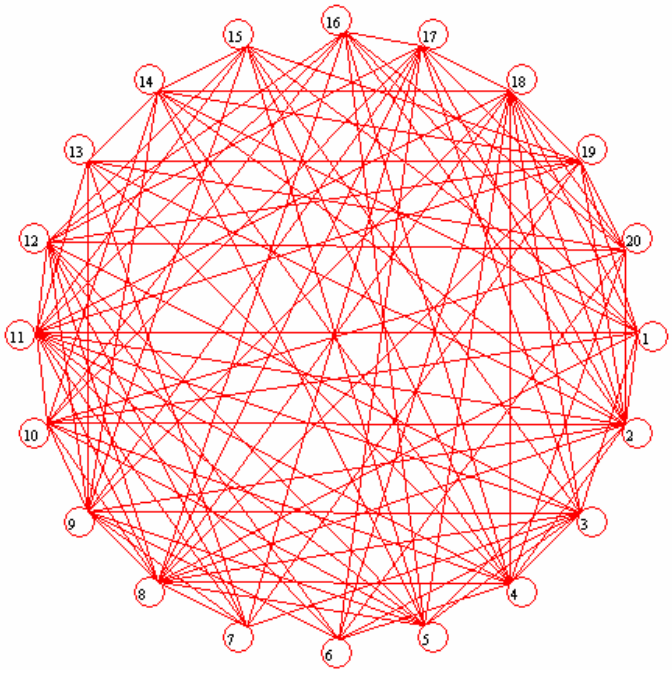
Exemple 2 Le sommet ‘1’ est de degré 11 et tous les autres sommets sont de degré ‘3’



L’arbre ‘2’

Lors de l’exécution de notre logiciel nous avons trouvé la valeur de cordalité k=11, nous avons 11 cycles élémentaires qui sont : C_1=1,2,3 - C_2=1,3,4 - C_3=1,4,5 - C_4=1,5,6 - C_5=1,6,7 – C_6=1,7,8 – C_7=1,8,9 - C_8=1,9,10 – C_9=1,10,11 – C_{10}=1,11,12 - C_{11}=2,3,4,5,6,7,8,9,10,11,112.

Pour la décomposition arborescente du graphe G donnée par l’arbre ‘2’, tel que $tw(G)=3$, $tl(G)=2$ et $|F|=3$.



Exemple 3 Un graphe quelconque

Lors de l'exécution de notre logiciel nous avons trouvé la valeur de cordalité k=7, nous avons 489 cycles élémentaires C_1=1,2,4,3 - … - C_{489}=11,12,20,2.
Le dessin de l'arbre de la décomposition arborescente pour le graphe G est très difficile mais le logiciel calcul les invariants liés à la décomposition arborescente, tel que $tw(G)=8$, $tl(G)=4$ et $|F|=8$.

Remarques

- Vu la difficulté du dessin du graphe en programmation, le logiciel donne pour des cas particuliers l'arbre de la décomposition arborescente.
- La cardinalité maximum des sommets est limitée par 300.
- Pour la décomposition en branches on a pas pu représenté l'arbre de la décomposition.
- Le logiciel donne une borne pour la largeur de branches, mais pour les autres paramètres la largeur arborescente, la longueur arborescente, la valeur de la distance et la taille de la frontière la majorité des instances il donne des valeurs exactes.

2. Qu'est-ce que le routage

Dans un réseau de communication point à point ou dans les ordinateurs parallèles, une fonction de routage est utilisée pour envoyer des messages entre processeurs. Quand les réseaux grossissent, il devient important de réduire la quantité de mémoire

contenue dans chaque noeud pour l'opération de routage. En même temps, il est essentiel d'acheminer les messages avec les chemins les plus courts possibles.

Un schéma de routage universel est un algorithme qui génère une fonction de routage pour tout réseau donné. Un type de schéma de routage universel trivial est basé sur des schémas dans lesquels chaque noeud contient une table de routage complète qui spécifié un port de sortie pour toute destination. Même si ce schéma peut garantir un routage selon les plus courts chemins, chaque routeur doit stocker localement $n\log d$ bits de mémoire, où d est le degré du routeur (c.a.d. le nombre de ports de sortie) et n est le nombre de noeuds du réseau.

Cependant, ce schéma est impraticable quand on traite de grands réseaux.

2.1 Schéma de Routage

Un schéma de routage est composé de :

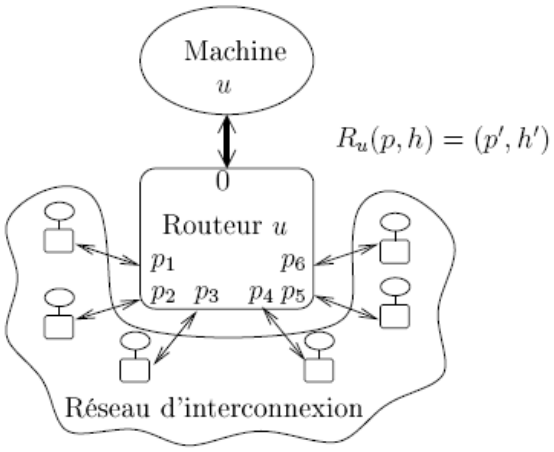
1. Les adresses données aux machines,

2. Les en-têtes ajoutés à tout message,

3. La numérotation des ports au niveau d'un routeur,

4. La fonction locale de routage de chaque routeur,

5. La mémoire locale de chaque routeur.



Exemple de Routeur

Un schéma de routage est dit valide si on est capable d'envoyer un message entre tout couple de sommets. C'est-à-dire si tout sommet u est capable, en utilisant sa mémoire locale et l'adresse de n'importe quel autre sommet v, de construire un en-tête de message permettant de faire circuler un message dans le réseau entre u et v.

Et dans notre mémoire on est à la recherche des invariants qu'ils vont être utilisés pour la construction d'un schéma de routage valide.

2.2 Caractéristiques d'un schéma de routeur

Nous allons à présent nous intéresser aux trois caractéristiques principales d'un schéma de routage : la taille des informations nécessaires à sa réalisation ; la longueur des routes produites entre deux sommets ; la rapidité de décision des routeurs.

2.2.1 Taille des informations

- La mémoire locale d'un sommet *u*, notée *mem*(*u*)
- La taille des adresses
- La taille en-têtes de message

Il est en effet possible de stocker, dans l'adresse de tout sommet, la route permettant d'y arriver depuis n'importe quel autre sommet.

2.2.2 Longueur des routes

Étant donné un schéma de routage dans un graphe *G*, la longueur de la route produite par la fonction de routage *R* entre deux sommets *u* et *v*, notée $\rho_R(u,v)$, est le nombre d'arêtes traversées par le message depuis la source avant d'atteindre la destination. Un schéma de routage est dit de plus courts chemins si pour tout couple de sommets $((u,v),\ \rho_R(u,v) = dist(u,v)$.

2.2.3 Rapidité de décision

Le temps nécessaire à un routeur pour décider vers où il doit faire transiter le message.

3. Justification du passage

3.1 Graphe →Décomposition arborescente

Au lieu d'étudier tous le graphe on décompose le graphe en arbre enraciné pour avoir deux sommets reliés par une chaîne.

3.2 Décomposition arborescente→Calcul de la largeur arborescente

Si la largeur arborescente est bornée par une constante suffisamment petite, certains problèmes NP-difficile devient polynomial.

D'autre part, plus la largeur arborescente est petite plus le graphe ressemble à un arbre, pour voir combien ce graphe est poche d'un arbre.

Par ailleurs, si la connexité du graphe est petite on dit que le graphe est peu connexe.

3.3 Calcul de la largeur arborescente →Recherche de la clique maximale potentielle

Si la détermination de l'ensemble des cliques maximales potentielles est polynomial alors la largeur arborescente est bornée.

3.4 Détermination Clique maximal potentiel→ Recherche des séparateurs minimaux

Si la détermination des séparateurs est polynomiale alors la détermination des cliques maximums est aussi polynomiale.

3.5 Triangulation minimale→ Séparateurs minimaux

Toute triangulation minimale d'un graphe *G* peut être obtenue en prenant un ensemble maximal de séparateurs minimaux deux à deux parallèles et en les complètes en clique pour trouver la clique maximal potentiel.

3.6 Décomposition arborescente →Calcul de la longueur arborescente

Tous graphe de longueur arborescente δ admet un schéma de routage de déviation 2δ et dont les adresses et les mémoires sont de taille $O(\delta \log^3 n)$ bits par sommet.

Remarque

Ce qui justifie la recherche des valeurs minimums de la longueur arborescente et la largeur arborescente.

Définition (LB – simplicial**)** Un sommet x est dit LB – simplicial si tout séparateur minimale inclus dans son voisinage est une clique.

Algorithme LB – Triang [Ber98]

Entrée : Un graphe G = (V, E)

Sortie : Une triangulation minimale de G

Pour tout $x \in$ V faire

Rendre *x* LB – simplicial

Fin

1. Qu’est-ce que le routage

Dans un réseau de communication point à point ou dans les ordinateurs parallèles, une fonction de routage est utilisée pour envoyer des messages entre processeurs. Quand les réseaux grossissent, il devient important de réduire la quantité de mémoire contenue dans chaque noeud pour l'opération de routage. En même temps, il est essentiel d'acheminer les messages avec les chemins les plus courts possibles.

Un schéma de routage universel est un algorithme qui génère une fonction de routage pour tout réseau donné. Un type de schéma de routage universel trivial est basé sur des schémas dans lesquels chaque noeud contient une table de routage complète qui spécifié un port de sortie pour toute destination. Même si ce schéma peut garantir un routage selon les plus courts chemins, chaque routeur doit stocker localement $n\log d$ bits de mémoire, où d est le degré du routeur (c.a.d. le nombre de ports de sortie) et n est le nombre de noeuds du réseau.

Cependant, ce schéma est impraticable quand on traite de grands réseaux.

2. Justification du passage

2.1 Graphe →Décomposition arborescente

Au lieu étudier tous le graphe on décompose le graphe en arbre enraciné pour avoir deux sommets reliées par une chaîne.

2.2 Décomposition arborescente→Calcul de la largeur arborescente

Si la largeur arborescente est bornée par une constante suffisamment petite, certain problème NP-difficile devient polynomial.

Autre part, Plus la largeur arborescente est petite plus le graphe ressemble à un arbre, pour voir combien ce graphe est poche d’un arbre.

Par ailleurs, si la Connexité de graphe est petite le graphe est peu connexe.

2.3 Calcul de la largeur arborescente →Recherche de la clique maximale potentielle

Si la détermination de l’ensemble des cliques maximales potentielles est polynomial alors la largeur arborescente est bornée.

2.4 Détermination Clique maximal potentiel→ Recherche des séparateurs minimaux

Si la détermination des séparateurs est polynomial alors la détermination des cliques maximums est aussi polynomial.

Triangulation minimale→ Séparateurs minimaux

Toute triangulation minimale d'un graphe *G* peut être obtenue en prenant un ensemble maximal de séparateurs minimaux deux à deux parallèles et en les complètes en clique pour trouver la clique maximal potentiel.

2.5 Décomposition arborescente →Calcul de la longueur arborescente

Bibliographie

[ACP87] Arnborg (S.), Corneil (D.) et Proskurowski (A.), « Complexity of finding embeddings in a k-tree », *SIAM Journal on Algebraic and Discrete Methods*, vol. 8, 1987, p. 277–284.

[ALS91] Arnborg (S.), Lagergren (J.) et Seese (D.), « Easy problems for tree-decomposable graphs », *Journal of Algorithms*, vol. 12, 1991, p. 308–340.

[BBC92] Beauquier (D.), Bersetl (J.) et Chrétienne (P.), *Éléments d'algorithmique*. Masson, 1992.

[BBC00] Berry (A.), Bordat (J.) et Cogis (O.), « Generating all the minimal separators of a graph », *International Journal of Foundations of Computer Science*, vol. 11, no 3, 2000, p. 397–403.

[Ber95] Berry (A.), « Treillis de Galois des Séparateurs Minimaux d'un Graphe non-orienté » Mémoire de DEA, LIRMM, Monpellier Août 1995.

[Ber98] Berry (A.), « Désarticulation d'un graphe » Thèse (PhD Dissertation), LIRMM, Monpellier, Décembre 1998.

[BKK95] Bodlaender (H.), Kloks (T.) et Kratsch (D.), « Treewidth and pathwidth of permutation graphs », *SIAM Journal on Discrete Mathematics*, vol. 8, 1995, p. 606–616.

[BKKM02] Broersma (H.), Kloks (T.), Kratsch (D.) et Müller (H.), « A generalization of AT-free graphs and a generic algorithm for solving triangulation problems », *Algorithmica*, vol. 33, no 4, 2002, p. 461–493.

[BMT03] Bouchitté (V.), Mazoit (F.) et Todinca (I.), « Chordal embeddings of planar graphs », *Discrete Mathematics*, vol. 273, 2003, p. 85–102.

[Bod96] Bodlaender (H.), «A linear-time algorithm for finding treedecompositions of small treewidth», *Siam Journal on Computing*, vol. 25, 1996, p. 1305–1317.

[Bou05] Boutiche (M. A.), «Graphes et problèmes de Routage», Thèse de Magister, USTHB , Octobre 2005.

[BP94] Blair (J. R. S.) and Peyton (B. W.), «On finding minimum-diameter clique trees», Nordic Journal of Computing, 1 (1994), pp. 173-201.

[BPT91] Borie (R.), Parker (R.) et Tovey (C.), « Deterministic decomposition of recursive graph classes », *SIAM Journal on Discrete Mathematics*, vol. 4, 1991, p. 481–501.

[BT97] Bodlaender (H.) et Thilikos (D.), « Constructive linear time algorithms for branchwidth », dans *Proceedings of the 24th*

International Colloquium on Automata, Languages and Programming (ICALP'97), vol. 1256, p. 627–637, 1997.

[BT98] Bouchitté (V.) and Todinca (I.). «Minimal triangulations for graphs with "few" minimal separators». In Proceedings 6th Annual European Symposium on Algorithms (ESA'98), volume 1461 of Lecture Notes in Computer Science, pages 344_355. Springer-Verlag, 1998.

[BT98a] V. Bouchitté and I. Todinca. Minimal triangulations for graphs with "few" minimal separators. In Proceedings 6th Annual European Symposium on Algorithms (ESA'98), volume 1461 of Lecture Notes in Computer Science, pages 344_355. Springer-Verlag, 1998.

[BT98b] V. Bouchitté and I. Todinca. Treewidth and minimum _ll-in of weakly triangulated graphs. Research Report RR98-40, LIP-ENS, 1998. A paraître dans Proceedings of STACS'99.

[BT01a] Bouchitté (V.) et Todinca (I.), « Listing all potential maximal cliques of a graph », *Theoretical Computer Science*, vol. 276, no 1-2, 2001, p. 212–323.

[BT01b] Bouchitté (V.) et Todinca (I.), « Treewidth and minimum fill-in : grouping the minimal separators », *SIAM Journal on Computing*, vol. 31, 2001, p. 212–232.

[BT03] Bouchitté (V.) et Todinca (I.), « Approximating the treewidth of at-free graphs », *Discrete Applied Mathematics*, vol. 131, no 1, 2003, p. 11–37.

[BT98a] V. Bouchitté and I. Todinca. Minimal triangulations for graphs with "few" minimal separators. In Proceedings 6th Annual European Symposium on Algorithms (ESA'98), volume 1461 of Lecture Notes in Computer Science, pages 344_355. Springer-Verlag, 1998.

[CLR01] Cormen (T.), Leiserson (C.) et Rivest (R.), *Introduction to algorithms*. MIT Press, 2001.

[Cou89] Courcelle (B.), « The monadic second-order logic of graphs II : Infinite graphs of bounded width », *Mathematical Systems Theory*, vol. 21, 1989, p. 187–221.

[Die00] Diestel (R.), *Graph theory*, vol. 173. Springer-Verlag, 2000.

[Dir61] Dirac (G.), « On rigid circuit graphs », *Abhandlungen Mathematischer Seminare der Universität Hamburg*, vol. 21, 1961, p. 71–76.

[Dou03] Dourisboure (Y.), «Routage compact et longueur arborescente », *Thèse de Doctorat Ecole Doctorale de Mathématiques et D'informatique Université Bordeaux I. 15 Décembre 2003*

[Esc72] Escalante (F.), « Schnittverbände in graphen », dans *Abhandlungen aus dem Mathematischen Seminar des Universität Hamburg*, vol. 38, p. 199–220, 1972.

[Fea88] Feautrier (P.), « Parametric integer programming », *Operationnelle/ Operations Research*, vol. 22, no 3, 1988, p. 243–268. [Gav74] Gavril (F.), « The intersection graphs of a path in a tree are exactly the chordal graphs », *Journal of Combinatorial Theory*, vol. 16, 1974, p. 47–56.

[Gav74] Gavril (F.), « The intersection graphs of a path in a tree are exactly the chordal graphs », *Journal of Combinatorial Theory*, vol. 16, 1974, p. 47–56.

[GKKPP00] Gavoille (C.), Katz (M.),. Katz (N. A), Paul (C.), and Peleg (D.), «Approximate distance labeling schemes», Research Report RR-1250-00, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, Dec. 2000.

[Gol80] Golumbic (M. C.), «Algorithmic graph theory and perfect graphs», Academic Press, New York, (1980).

[Hal76] Halin (R.), « S-functions for graphs », *Journal Geometry*, vol. 8, 1976, p. 171–186.

[Hic00] Hicks (I.), *Branch decompositions and their applications*. Thèse de doctorat, Rice University, 2000.

[HL89] Ho (C.) et Lee (R.), « Counting clique trees and computing perfect elimination schemes in parallel », *Information rocessing Letters*, vol. 31, 1989, p. 61–68.

[KK94] Kloks (T.) and Kratsch (D.), Finding «all minimal separators of a graph. In Proceedings» 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS'94), volume 775 of Lecture Notes in Computer Science, p 759-768. Springer-Verlag, 1994.

[KK98] Kloks (T.) et Kratsch (D.), « Listing all minimal separators of a graph », *SIAM Journal on Computing*, vol. 27, no 3, 1998, p. 605–613.

[KKM95] T. Kloks, D. Kratsch, and H. Müller. Approximating the bandwidth for asteroidal triple-free graphs. In Proceedings Third Annual European Symposium on Algorithms (ESA'95), volume 979 of Lecture Notes in Computer Science, pages 434_447. Springer-Verlag, 1995.

[KKM98] Kloks (T.), Kratsch (D.), and Müller (H.), 1998. Private communication

[KKM99] Kloks (T.), Kratochvíl (J.) et Müller (H.), « New branchwidth territories », dans *Proceedings 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, vol. 1563 (coll. *Lecture Notes in Computer Science*), p. 173–183. Springer-Verlag, 1999.

[Klo96] Kloks (T.), « Treewidth of circle graphs », *International Journal of Foundations of Compututer Science*, vol. 7, 1996, p. 111–120.

[Kru60] Kruskal (J.), « Well-quasi ordering, the tree theorem, and

Vàszonyi's conjecture », *Transaction of the American Mathematical Society*, vol. 95, 1960, p. 210–255.

[Lap96] Lapoire (D.), *Structuration des graphes planaires*. Thèse de doctorat, Université Bordeaux I, 1996.

[LB62] Lekkerkerker (C.) et Boland (J.), « Representation of a finite graph by a set of intervals on the real line », *Fundamenta Mathematicae*, vol. 51, 1962, p. 45–64.

[Lun90] Lundquist (M.), *Zero Patterns, Chordal Graphs and Matrix Completion*. Thèse de doctorat, Clemson University, 1990.

[Lya04] Lyaudet (L.), « Largeur de branches des graphes de cordes ». Mémoire de DEA, École normale supérieure de Lyon, 2004.

[Maz04] Mazoit (F.), « Décompositions algorithmiques des graphes » Thèse Doctorat, École normale supérieure de Lyon, Décembre 2004.

[McC03] McConnell (R. M.), « Linear-time recognition of circular-arc graphs », *Algorithmica*, vol. 37, 2003, p. 93–147.

[Pau94] C. Paul Etude structurelle des Séparateurs Minimaux d'un graphe Mémoire de DEA LIRM Montpellier 1994.

[Par96] A. Parra, «Structural and algorithmic aspects of chordal graphs embeddings», 1996. PhD Thesis, Technische Universität, Berlin.

[PS97] Parra (A.) et Scheffler (P.), « Characterizations and algorithmic applications of chordal graph embeddings », *Discrete Applied Mathematics*, vol. 79, no 1-3, 1997, p. 171–188.

[Ros74] Rose (D.), « On simple characterization of k-trees », *Discrete Mathematics*, vol. 7, 1974, p. 317–322.

[RS84] Robertson (N.) et Seymour (P.), « Graphs minors. III. Planar tree-width », *Journal of Combinatorial Theory Series B*, vol. 36, 1984, p. 49–64.

[RS86] Robertson (N.) et Seymour (P.), « Graphs minors. II. Algorithmic aspects of tree-width », *Journal of Algorithms*, vol. 7, 1986, p. 309–322.

[RS91] Robertson (N.) et Seymour (P.), « Graphs minors. X. Obstruction to tree-decomposition », *Journal of Combinatorial Theory Series B*, vol. 52, 1991, p. 153–190.

[RS94] Robertson (N.) et Seymour (P.), « Graphs minors. XI. Circuits on a surface », *Journal of Combinatorial Theory Series B*, vol. 60, 1994, p. 72–106.

[RS95] Robertson (N.) et Seymour (P.), « Graphs minors. XIII. The disjoint paths problem », *Journal of Combinatorial Theory Series B*, vol. 64, 1995, p. 240–272.

[RTL76] Rose (D.), Tarjan (R. E.), and Lueker (G. S.), «Algorithmic aspects of vertex elimination on graphs, SIAM Journal on Computing, 5 (1976), pp. 266-283.

[SL97] Shen (H.) et Liang (W.), « Efficient enumeration of all minimal separators in a graph », *Theoretical Computer Science*, vol. 180, 1997, p. 169–180.

[SSP94] Sundaram (R.), Sher Singh (K.) et Pandu Rangan (C.), « Treewidth of circular-arc graphs », *SIAM Journal Discrete Mathematics*, vol. 7, 1994, p. 647–655.

[ST94] Seymour (P.) et Thomas (R.), « Call routing and the ratcatcher », *Combinatorica*, vol. 14, no 2, 1994, p. 217–241.

[Tod99] Todinca (I.), *Aspects algorithmiques des triangulations minimales des graphes*. Thèse de doctorat, École Normale Supérieure de Lyon, 1999.

[Wag37] Wagner (K.), « Über eine Eigenschaft der ebenen Komplexe », *Mathematische Annalen*, vol. 114, 1937, p. 570–590.