Chapitre 6

Décompositions arborescentes[†]

6.1. Introduction

Ce chapitre est une introduction à la notion de décomposition arborescente d'un graphe, notion qui fait l'objet d'une recherche intense comme le montre le grand nombre de communications à son sujet présentées dans les colloques de théorie des graphes et d'algorithmique.

Les décompositions arborescentes interviennent de façon essentielle dans plusieurs domaines : la construction d'algorithmes polynomiaux, l'étude des classes de graphes caractérisées par des *configurations interdites*, dont le paradigme est la caractérisation de Kuratowski des graphes planaires, la caractérisation des ensembles de graphes pour lesquels la satisfaisabilité d'une formule de la *logique du second-ordre monadique* est décidable, et enfin la théorie des grammaires de graphes.

Ces quatre directions de recherche ont été initiées indépendamment et ont convergé en mettant en évidence les rôles centraux des décompositions de graphes et de la logique du second-ordre monadique. Esquissons l'historique de ces différentes approches.

Beaucoup de problèmes NP-complets ont des algorihmes polynomiaux pour des classes particulières de graphes. Les classes de *graphes structurés*, c'est-à-dire de graphes que l'on peut construire au moyen d'un nombre fini de graphes de base et d'un nombre fini d'opérations de « recollement de graphes » sont vite apparues comme très propices à cette recherche. Le traitement d'exemples tels que la classe des graphes série-parallèles a permis de dégager la notion de k-arbre partiel, c'est-à-dire, sous un autre nom, de graphe de largeur arborescente au plus k. La bibliographie publiée en 1994 par Hedetniemi [Hedet] comporte 238 entrées : elle reflète bien les débuts de cette algorithmique des graphes structurés. Sa mise à jour comporterait sans doute au moins mille titres de plus.

[†]Chapitre rédigé par Bruno Courcelle, *Université Bordeaux 1, LaBRI (CNRS)*, email : courcell@labri.fr

La deuxième approche est celle de Robertson et Seymour. Ils ont commencé vers 1980 un travail de longue haleine qui a débouché sur la preuve du théorème des mineurs de graphes. Ce théorème énonce que toute famille de graphes fermée par passage aux mineurs est caractérisée, telle la famille des graphes planaires, par un nombre fini de mineurs exclus. C'est de leur analyse de la notion de mineur que proviennent les notions de *décomposition et de largeur arborescentes*.

La troisième approche consiste à tenter de caractériser les classes de graphes pour lesquelles la satisfaisabilité d'une formule de *la logique du second-ordre monadique est décidable*, et à montrer, que seules des classes d'arbres ou de graphes qui en sont proches (en un sens précis) possèdent cette propriété de décidabilité. Seese a formulé une conjecture en ce sens ([35]). Il a montré en particulier qu'un ensemble de graphes planaires qui satisfait cette propriété de décidabilité est de largeur arborescente bornée. Dans ce cadre et pour les graphes planaires, la largeur arborescente est un paramètre qui caractérise bien la proximité d'une classe de graphes à une classe d'arbres. Ce résultat a fait l'objet de développements par Courcelle et Oum ([24]) en vue d'une preuve de la conjecture qui est toujours ouverte dans sa formulation initiale.

Enfin, la quatrième approche consiste à étendre aux ensembles de graphes les descriptions de langages (ensembles de mots ou de termes algébriques) par des grammaires et des automates qui sont l'objet de la *théorie des langages formels*. Les recollements de graphes utiles en algorithmique sont vus dans ce contexte comme des généralisations aux graphes de la concaténation des mots. Deux familles de grammaires *non contextuelles* (*context-free*) de graphes ont été identifiées. L'une d'entre elles est intrinsèquement liée aux décompositions arborescentes.

Ce chapitre est centré sur les deux premières approches des décompositions arborescentes. Pour des exposés plus détaillés, on recommandera le livre de Downey et Fellows [9] et l'ouvrage collectif [10].

6.2. Décompositions arborescentes

6.2.1. Définitions

Les notions de décomposition arborescente et de largeur arborescente sont données pour les graphes non orientés. Elles s'appliquent aux graphes orientés par l'intermédiaire de leurs graphes non orientés sous-jacents. Les principales références sont le livre de Downey et Fellows [9] et l'article de synthèse de Bodlaender [1].

Tous les graphes considérés dans ce chapitre sont finis. Pour un graphe G, $s(G)$ désigne le nombre de sommets, $e(G)$ le nombre d'arcs ou d'arêtes. Un graphe est *simple* s'il n'a pas de boucles ni d'arcs ou arêtes multiples.

Un graphe est nécessairement défini par des ensembles de sommets, d'arcs ou d'arêtes et des relations d'incidence. Mais deux graphes isomorphes ont les mêmes propriétés. On ne s'intéresse en fait qu'aux graphes *abstraits*, c'est-à-dire aux classes d'équivalence des graphes pour l'isomorphisme. On parlera d'un graphe *concret* ou *étiqueté* pour insister sur le fait que ses ensembles de sommets et d'arcs ou d'arêtes sont des ensembles précis. Les graphes concrets sont nécessaires comme étapes intermédiaires de constructions destinées à prouver des propriétés de graphes abstraits. La distinction entre graphe abstrait et graphe concret sera le plus souvent passée sous silence.

Définition 6.2.1.1 : décompositions arborescentes

Soit $G = (X, E)$ un graphe dont X est l'ensemble des sommets et E l'ensemble des arêtes. Une *décomposition arborescente* (*tree decomposition*) de G est un couple (T, f) tel que T est un arbre, $T = (N, A)$, et f est une application qui associe à tout nœud u de T un ensemble $f(u) \subseteq X \cup E$, et qui satisfait les conditions suivantes :

(1) tout sommet de G appartient à quelque ensemble $f(u)$,

(2) toute arête de G appartient à un et un seul ensemble $f(u)$,

(3) si une arête appartient à $f(u)$, ses extrémités appartiennent aussi à $f(u)$,

(4) pour tout sommet x, l'ensemble des nœuds u tels que $x \in f(u)$ induit un sous-graphe connexe de T (donc un arbre).

On utilisera le terme *nœud* pour désigner les sommets des arbres. Cette convention est commode lorsque l'on parle à la fois d'un graphe et d'un arbre qui représente sa structure. Les ensembles $f(u)$ sont les *boîtes* de la décomposition (T, f). Sa *largeur* est l'entier $wd(T, f) = Max\{Card(X \cap f(u)) \mid u \in N\} - 1$.

La *largeur arborescente de* G (*treewidth*) est la plus petite largeur d'une décomposition arborescente de G. Elle sera notée $twd(G)$, notation qui fait référence à la terminologie anglaise. Une décomposition arborescente est *optimale* si sa largeur est la plus petite possible pour le graphe considéré.

Une décomposition arborescente peut ne comporter qu'une boîte qui contient alors tout le graphe. Il en résulte que $twd(G) \leq s(G) - 1$. Les arbres sont de largeur arborescente 1 et les graphes dont toutes les arêtes sont des boucles sont de largeur arborescente 0. Le cycle C_m, dont l'ensemble des sommets est $X = \{1, 2, 3, ..., m\}$ et les arêtes sont $1-2$, $2-3$, ..., $(m-1)-m$, $m-1$, ce que l'on note brièvement C_m=$1-2-3-...-m-1$, a une largeur arborescente 2 si $m \geq 3$. Il admet comme décomposition arborescente (T, f) où T est l'arbre (en fait la chaîne) $1-2-3-...-(m-1)$, $f(1) \cap X = \{1, 2\}$ et $f(i) \cap X = \{1, i, i+1\}$ pour $i = 2, ..., m-1$. Le lecteur précisera sans peine les ensembles $f(i) \cap E$. Cela montre que $twd(C_m) \leq 2$. On a $twd(C_m) = 2$ si $m \geq 3$, car il n'existe pas de décomposition de C_m de largeur 1. Cela sera

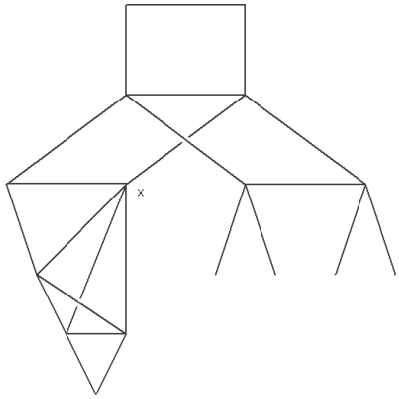

Figure 6.1. – *Un graphe G*

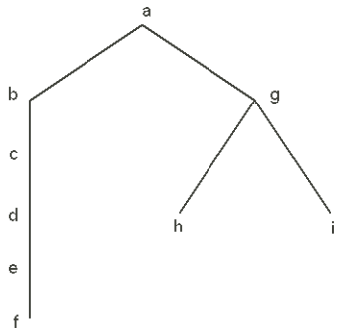

Figure 6.2. – *L'arbre T d'une décomposition arborescente de G*

montré plus loin (proposition 6.8) mais le lecteur pourra le vérifier directement pour se familiariser avec la définition.

Interprétation intuitive

Soit (T, f) une décomposition arborescente de G. Pour chaque nœud u de T soit $G(u)$ le sous-graphe de G constitué des sommets et des arêtes qui appartiennent à $f(u)$. Le graphe G est ainsi partitionné en sous-graphes sans arêtes communes, et ces sous-graphes sont « recollés » au niveau de leurs sommets communs, selon la configuration globale de l'arbre T. Intuitivement, la condition (4) de la définition interdit des recollements formant des cycles. Elle impose que si $G(u)$ et $G(v)$ ont des sommets en commun et donc sont recollés au niveau de ces sommets, alors, ils sont recollés par ces sommets à tous les graphes associés aux nœuds de la chaîne qui relie u à v.

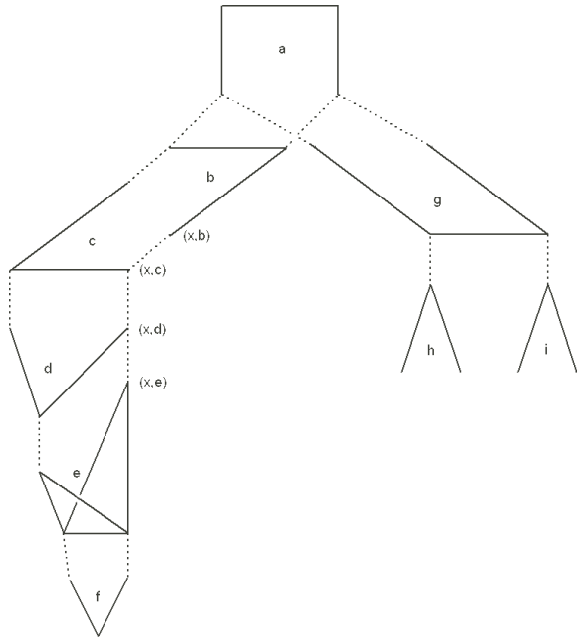

Figure 6.3. – *Une décomposition arborescente de G. Le graphe G est obtenu par contraction des arêtes en pointillé*

Les figures 6.1, 6.2 et 6.3 montrent un graphe G et une décomposition arborescente (T, f) de ce graphe. Sur la figure 6.3, les boîtes de la décomposition sont représentées comme des graphes disjoints (traits pleins). Les arêtes en pointillés relient les sommets de ces graphes qui sont à fusionner, de manière à recoller les graphes induits par les boîtes de la décomposition. Le sommet x du graphe G appartient aux boîtes b, c, d, e, lesquelles forment une partie connexe de T, conformément à la condition (4). Sur la figure 6.3, les sommets $(x, b), (x, c), (x, d), (x, e)$ de ces quatre boîtes correspondent à ce sommet x. Le graphe G est obtenu par contraction des arêtes en pointillés du graphe de la figure 6.3. (La contraction d'arête est définie dans la section 6.2.3). Le fusionnement de (x, e) et de (x, b) résulte par transitivité des fusionnements indiqués par les arêtes en pointillés.

Décompositions linéaires

Si dans la définition d'une décomposition aborescente, on impose à l'arbre T d'être une chaîne, on obtient la notion de *décomposition linéaire* (*path decomposition*) et celle de *largeur linéaire* (*pathwidth*), notée $pwd(G)$. La décomposition proposée plus haut des cycles C_m est une décomposition linéaire et l'on a $pwd(C_m) = 2$ pour $m \geq 3$. Puisque toute chaîne est un

arbre, on a $twd(G) \leq pwd(G)$ pour tout graphe G. Les arbres sont de largeur arborescente 1 mais de largeur linéaire non bornée.

On désignera par $TWD(\leq k)$ et $PWD(\leq k)$ les ensembles des graphes de largeur arborescente au plus k et de largeur linéaire au plus k, respectivement.

Quelques exemples

On a pour quelques graphes classiques les valeurs suivantes :
$twd(K_m) = pwd(K_m) = m - 1$,
$twd(P_m) = pwd(P_m) = 1$,
$twd(T_n) = 1$, $pwd(T_n) = \lfloor n/2 \rfloor$, (exercice 1),
où K_n est le graphe complet à n sommets, P_n est la chaîne à $n-1$ arêtes et n sommets et T_n est l'arbre binaire complet de hauteur $n-1$, $n \geq 1$, dont l'ensemble des nœuds est $\{0, 1, ..., 2^n - 2\}$, et les arêtes sont $i - (2i+1)$ et $i - (2i+2)$ pour $i = 0, ..., 2^{n-1} - 2$ et $n \geq 2$. On notera $G_{n \times m}$ la grille carrée définie comme le produit cartésien $P_n \times P_m$. Ses sommets sont les couples (i, j) pour $1 \leq i \leq n, 1 \leq j \leq m$ et ses arêtes relient (i, j) et (i', j') si et seulement si $\mid i - i' \mid + \mid j - j' \mid = 1$. Sauf si $n = m = 1$, on a $twd(G_{n \times m}) = pwd(G_{n \times m}) = Min\{n, m\}$ ([1]). Si G est un graphe *planaire extérieur* alors $twd(G) \leq 2$. Dans tous ces cas, il n'est pas très difficile de construire des décompositions qui établissent les majorations. Il est plus difficile de montrer qu'une décomposition est optimale. Nous verrons plus loin quelques méthodes pour ce faire (proposition 6.8).

Variantes des définitions

On n'a pas imposé que chaque boîte soit non vide. Mais toute décomposition arborescente (T, f) d'un graphe non vide peut être transformée en une décomposition arborescente de ce graphe de même largeur dont aucune boîte n'est vide. En effet, si $f(u)$ est vide, si v est un nœud voisin de u, il suffit de fusionner ces nœuds (en contractant l'arête qui les relie, une définition formelle est donnée plus bas) et en prenant $f(v)$ comme boîte associée au nœud résultant de cette fusion. On vérifie sans peine que l'on obtient une décomposition arborescente du même graphe et de même largeur. On répète la transformation autant de fois que nécessaire.

D'autre part, on peut remplacer la condition (2) par la condition :
(2') Toute arête de G appartient à au moins une boîte $f(u)$.
Dans ce cas les graphes $G(u)$ peuvent avoir des arêtes en commun. Leurs ensembles d'arêtes forment un recouvrement et non plus une partition des arêtes du graphe G.

On peut également définir une décomposition arborescente (T, f) en demandant que $f(u) \subseteq X$ et en remplaçant les conditions (2) et (3) par l'unique condition que toute arête a ses extrémités dans une même boîte.

Ces variantes, qui peuvent être utiles pour faciliter certaines constructions ne modifient pas les notions de largeur arborescente et de largeur linéaire.

Sauf indication contraire, les décompositions arborescentes et linéaires seront toujours conformes à la définition initiale.

Décompositions arborescentes enracinées

Une décomposition arborescente (T, f) d'un graphe G est dite *enracinée* si l'arbre T est muni d'une racine, d'où l'on déduit une unique orientation telle que tout nœud est accessible depuis la racine par un chemin, ce qui fait de T une *arborescence*. La racine est de degré entrant 0. On notera $\leq_T$ la relation d'ordre telle que $x \leq_T y$ si et seulement si, ou bien $x = y$, ou bien il existe un chemin de x à y. Il résulte alors de la condition (4) de la définition 6.2.1.1 que pour tout sommet x de G, il existe un unique nœud de T, le plus proche possible de la racine, donc minimal pour $\leq_T$ (nous dirons le *plus haut dans l'arbre*, car nous dessinons les arbres avec la racine en haut, comme sur la figure 6.2) dont la boîte associée contient x. On dira que ce nœud est le *nœud d'introduction de* x et on le notera $NI(x)$. Cette terminologie se réfère à une construction du graphe par laquelle on introduit les sommets des différentes boîtes dans l'ordre d'un *tri topologique* qui débute à la racine de T, c'est-à-dire selon un ordre total $\leq'_T$ qui étend $\leq_T$. Avec ces définitions et notations :

LEMME 6.1. *Si deux sommets x et y de G sont adjacents, alors on a $NI(x) \leq_T NI(y)$ et $x \in f(NI(y))$ ou l'inverse, en échangeant les rôles de x et y.*

On peut avoir $NI(x) = NI(y)$ et alors, $x \in f(NI(y))$ et $y \in f(NI(x))$.

PREUVE. Soit u un nœud dont la boîte contient x et y. On a $u \geq_T NI(x)$ et $u \geq_T NI(y)$ et donc, $NI(x)$ et $NI(y)$ sont comparables pour $\leq_T$ (d'après les propriétés des arbres). Si $NI(x) \leq_T NI(y)$ alors $NI(y)$ est sur le chemin dans T de $NI(x)$ à u, et donc $x \in f(NI(y))$ d'après la condition (4) de la définition 6.2.1.1. □

Un *ordre d'élimination de type k pour un graphe simple G* est une énumération $x_1, ..., x_n$ de l'ensemble de ses sommets telle que pour tout i, l'ensemble des indices j tels que $j < i$ et x_i est adjacent à x_j a au plus k éléments. Cet ordre fournit une construction du graphe par ajouts successifs de $x_1, ..., x_n$. Le terme d'élimination (conservé car il est usuel) fait référence au processus inverse de suppression des sommets dans l'ordre $x_n, x_{n-1}, ..., x_1$; il permet de démontrer une propriété d'un graphe à n sommets en supposant prouvée la propriété pour les graphes à $n - 1$ sommets.

PROPOSITION 6.2. *Un graphe de largeur arborescente au plus k possède un ordre d'élimination de type k.*

PREUVE. Soit (T, f) une décomposition arborescente d'un graphe $G = (X, E)$, enracinée et de largeur au plus k. Soit r sa racine et la suite $r = n_0, n_1, ..., n_p$ un tri topologique de T . Pour tout $i = 0, ..., p$, soit X_i l'ensemble des sommets de G qui sont dans $f(n_i)$ mais dans aucune boîte $f(n_j)$ pour $j < i$. C'est l'ensemble des sommets « introduits à l'étape i » si on envisage $n_0, n_1, ..., n_p$ comme une suite d'étapes qui ajoutent des sommets et des arêtes. On a $X_i = NI^{-1}(n_i)$. On choisit une énumération quelconque $\overrightarrow{X_i}$ de X_i et l'on prend la concaténation $\overrightarrow{X_0}...\overrightarrow{X_p}$ comme l'énumération cherchée de X. Soit x un sommet de G et y un sommet adjacent, situé avant x (donc x est introduit dans le graphe après y). On a $NI(y) \leq_T NI(x)$ et $y \in f(NI(x))$ (lemme 6.1). Il en résulte que ces sommets y sont en nombre au plus k puisque $f(NI(x))$ a au plus $k + 1$ éléments et contient x. On a donc bien construit un ordre d'élimination de type k. □

COROLLAIRE 6.3. *(1) Un graphe de largeur arborescente au plus k est $(k+1)$-coloriable. Pour toute décomposition arborescente de ce graphe de largeur au plus k, on peut choisir une $(k+1)$-coloration telle que deux sommets d'une même boîte sont de couleurs différentes.*

(2) Un graphe simple (donc sans boucle) à n sommets et de largeur arborescente au plus k a au plus $nk - k(k-1)/2$ arêtes.

PREUVE. (1) Soit $G = (X, E)$ un graphe de largeur arborescente au plus k muni d'un ordre d'élimination $x_1, ..., x_n$ construit par la proposition 6.2 à partir d'une décomposition (T, f). Le sommet x_1 est dans la boîte racine. On utilise $C = [1, k + 1]$ (c'est à dire l'ensemble d'entiers $\{1, ..., k + 1\}$) comme ensemble de couleurs. On définit un coloriage $c : X \longrightarrow C$ de la façon suivante : $c(x_1) = 1$, et pour $i > 1$, $c(x_i)$ est la plus petite couleur qui diffère de $c(y)$ pour tout sommet y dans $f(NI(x_i))$ qui précède x_i. Il existe toujours dans C une couleur disponible pour définir $c(x_i)$ puisque la décomposition donnée est de largeur au plus k et donc que $f(NI(x_i))$ a au plus $k + 1$ éléments. Deux sommets d'une même boîte sont donc de couleurs différentes. Deux sommets adjacents sont dans une même boîte et sont donc de couleurs différentes.

(2) Vérification immédiate en utilisant un ordre d'élimination de type k. □

REMARQUES. 1) Le corollaire 6.3 ne fournit pas nécessairement des colorations optimales. Pour un cycle C_{2n}, 2-coloriable et de largeur arborescente 2, il fournit une 3-coloration. En utilisant un ordre d'élimination de type 2 et en définissant pour $c(x_i)$ la plus petite couleur différente de $c(y)$ pour y adjacent à x_i et avant lui, on obtient une 2-coloration de C_{2n}, et en général, une coloration utilisant moins ou autant de couleurs que pour celle construite par la preuve du corollaire 6.3.

2) Toute grille $G_{n\times n}$ possède un ordre d'élimination de type 2 mais ces grilles sont de largeur arborescente non bornée.

De très nombreux travaux comparent la largeur arborescente et la largeur linéaire à d'autres paramètres tels que la *largeur de bande* (*bandwidth*) ou le *vertex separation number*. Citons le résultat suivant qui compare la largeur arborescente et la largeur linéaire (où l'on rappelle que $s(G)$ est le nombre de sommets de G).

PROPOSITION 6.4. (18) *Pour tout graphe G :*

$$pwd(G) \leq (twd(G)+1).log_2(s(G)).$$

6.2.2. Passage au sous-graphe

Lorsqu'un graphe est transformé en un autre, ou contient un autre graphe, comment se comparent leurs largeurs arborescentes ? Nous allons donner quelques réponses qui fourniront des outils pour obtenir des encadrements et parfois même des valeurs exactes pour les largeurs arborescentes de certains types de graphes.

PROPOSITION 6.5. *Si H est obtenu à partir de G au moyen d'ajouts et/ou de suppressions de boucles et/ou d'arêtes doubles, alors $twd(H) = twd(G)$ et $pwd(H) = pwd(G)$.*

PREUVE. Il suffit de montrer que la largeur arborescente est invariante par ajout d'une boucle ou doublage d'une arête existante.

Si H est obtenu à partir de G en ajoutant une boucle incidente à un sommet x (respectivement en doublant une arête $e : x - y$) et si (T, f) est une décomposition arborescente de G on obtient une décomposition de H en ajoutant cette boucle à une boîte qui contient x (respectivement, en ajoutant la nouvelle arête à une boîte qui contient x et y. On transforme une décomposition arborescente de H en une de G en supprimant la boucle ou l'arête en question. Dans les deux cas, les décompositions optimales de G et H ont mêmes largeurs. □

Il en résulte que l'étude de la largeur arborescente peut se concentrer sur les graphes simples, c'est-à-dire sans boucle ni arête multiple.

PROPOSITION 6.6. *(1) Si H est un sous-graphe de G alors $twd(H) \leq twd(G)$ et $pwd(H) \leq pwd(G)$.*

(2) Si H est obtenu en supprimant de G une arête e, ou bien un sommet x et toutes les arêtes incidentes, alors $twd(H) \leq twd(G) \leq twd(H)+1$ et $pwd(H) \leq pwd(G) \leq pwd(H)+1$.

PREUVE. (1) On construit à partir d'une décomposition (T, f) de G une décomposition de H en supprimant des boîtes $f(u)$ des arêtes et des sommets. La largeur de la décomposition ne peut que diminuer.

(2) Inversement, soit à construire une décomposition (T, g) de G à partir d'une décomposition (T, h) de H : soit y un sommet de l'arête e ou bien le sommet x. On obtient g en ajoutant y à chaque boîte de (T, h), ainsi que, dans les boîtes appropriées, l'arête e ou les arêtes incidentes à x. □

Composantes biconnexes

Un graphe est *biconnexe* s'il est connexe et n'est pas la réunion de deux sous-graphes connexes qui n'ont en commun qu'un sommet et ont chacun au moins une arête. Une *composante biconnexe* est un sous-graphe biconnexe maximal. Une boucle et un isthme sont des composantes biconnexes. Les composantes biconnexes d'un graphe G définissent une partition de son ensemble d'arêtes. Si G est connexe, on peut définir un arbre dont les nœuds correspondent à ses composantes biconnexes et dont toute arête $A - B$ correspond à un sommet commun aux composantes A et B. On dira que cet arbre est *un arbre des composantes biconnexes* de G. Cet arbre n'est pas défini de manière unique. Par exemple, si G est la réunion de 3 composantes biconnexes A, B, C qui ont un sommet en commun, on peut prendre comme arbre : $A - B - C$ ou $B - A - C$ ou $A - C - B$.

PROPOSITION 6.7. *La largeur arborescente d'un graphe est le maximum des largeurs arborescentes de ses composantes biconnexes.*

PREUVE. Soit G un graphe. Puisque toute composante biconnexe C est un sous-graphe, on a $twd(C) \leq twd(G)$ et donc le maximum des $twd(C)$ est au plus $twd(G)$.

Pour démontrer l'inverse considérons les composantes biconnexes $C_1, ..., C_n$ et pour chacune d'elles C_i, une décomposition arborescente optimale (T_i, f_i). Nous allons les regrouper en une décomposition arborescente de G. On suppose d'abord G connexe. Soit U un arbre des composantes biconnexes de G.

On prend l'union disjointe des arbres T_i (c'est-à-dire que certains des arbres T_i sont remplacés si besoin par des copies isomorphes disjointes de toutes les autres). A chaque arête $C_i - C_j$ de U correspond un sommet x de G. Soit u un nœud de T_i et v un nœud de T_j tels que $x \in f_i(u)$ et $x \in f_j(v)$. On crée une arête entre u et v. En faisant ainsi pour chaque arête de U on obtient un arbre T dont chaque T_i est un sous-arbre. On prend comme fonction f le prolongement commun des fonctions f_i. Les conditions (1)-(3) de la définition 6.2.1.1 sont clairement vérifiées. Pour vérifier la condition (4) on observe que lorsqu'un sommet x est commun à plusieurs composantes C_i, l'ensemble des arêtes $C_i - C_j$ de U auxquelles il correspond forme un

sous-arbre de U. D'autre part, dans chaque T_i l'ensemble A_i des nœuds w tels que $x \in f_i(w)$ est connexe. L'union de ces ensembles A_i forme donc dans T un unique ensemble connexe. □

La proposition qui suit permettra de minorer les largeurs arborescentes de certains graphes. Si Y et Z sont deux ensembles disjoints de sommets, on désigne par $Y \otimes Z$ le graphe biparti complet formé de toutes les arêtes entre Y et Z.

PROPOSITION 6.8. *Soit (T, f) une décomposition arborescente d'un graphe $G = (X, E)$.*

(1) Si Y est une partie de X qui induit un sous-graphe complet, il existe un nœud u de T tel que $Y \subseteq f(u)$,

(2) Si Y et Z sont deux parties disjointes de X telles $Y \otimes Z$ est un sous-graphe de G, alors il existe un nœud u de T tel que $Y \subseteq f(u)$ ou $Z \subseteq f(u)$.

(3) Pour tous entiers m, n, $twd(K_m) = m - 1$, et $twd(K_{n,m}) = Min\{n, m\}$, et, si $n \geq 3$, $twd(C_n) = 2$.

PREUVE. (1) On rappelle (voir l'exercice 4) que les parties connexes de l'ensemble des nœuds d'un arbre satisfont la *propriété de Helly* :

Si $A_1, ..., A_m$ sont des ensembles deux à deux d'intersection non vide, il existe un élément commun à tous ces ensembles.

Donc si (T, f) est une décomposition arborescente d'un graphe $G = (X, E)$ et si $Y = \{y_1, ..., y_m\} \subseteq X$ induit un sous-graphe complet, alors les ensembles A_i des nœuds p tels que $y_i \in f(p)$ sont connexes et deux à deux d'intersection non vide d'après les clauses (3) et (4) de la définition 6.2.1.1. Donc il existe un nœud u, commun à tous les A_i et donc $Y \subseteq f(u)$.

(2) Supposons que $Y = \{y_1, ..., y_m\} \subseteq X$ et que y_1 et y_2 ne sont pas dans une même boîte. Les ensembles A_1 et A_2 définis comme ci-dessus sont disjoints. Il existe un unique nœud n_1 dans A_1 et un unique nœud n_2 dans A_2 à distance minimale l'un de l'autre dans l'arbre T. Soit z quelconque dans Z. Il existe m_1 dans A_1 et m_2 dans A_2 tels que $f(m_1)$ et $f(m_2)$ contiennent respectivement les arêtes $y_1 - z$ et $y_2 - z$ et donc contiennent z. Il en résulte que z appartient à la boîte $f(p)$ pour tout nœud p de l'unique chaîne dans T de m_1 à m_2. Or cette chaîne contient la chaîne de n_1 à n_2. Ainsi Z est contenu dans $f(p)$ pour tout p de la chaîne de n_1 à n_2.

Donc si Z n'est pas contenu dans une boîte, cela veut dire que tout couple d'éléments de Y est contenu dans une même boîte. Il en résulte que Y est contenu dans une même boîte, en utilisant la propriété de Helly, comme pour prouver l'assertion (1).

(3) On a $twd(K_m) \geq m - 1$ d'après (1) et donc l'égalité. L'assertion (2) donne $twd(K_{n,m}) \geq Min\{n, m\} - 1$. Mais il n'est pas difficile de voir que K_{n+1} est mineur de $K_{n,m}$ si $n \leq m$ (la notion de mineur est définie dans la

section suivante). On en déduit, avec la proposition 6.10 que $twd(K_{n,m}) \geq twd(K_{Min\{n,m\}+1}) = Min\{n, m\}$. On a $twd(K_{n,m}) = Min\{n, m\}$. De même, K_3 est mineur de C_n si $n \geq 3$. On en déduit que $twd(C_n) = 2$. □

Les notions de décomposition et de largeur arborescentes s'étendent de façon très naturelle aux hypergraphes : les hyperarêtes qui généralisent les arêtes sont des ensembles de sommets de taille non limitée à 2. Il suffit de remplacer la condition (3) de la définition 6.2.1.1 par la condition qui impose que si une hyperarête est dans une boîte, alors tous ses sommets le sont aussi. D'autre part, on peut associer à un hypergraphe H un graphe $K(H)$ qui a les mêmes sommets et une arête entre deux sommets si et seulement si ces deux sommets appartiennent à une même hyperarête. Il résulte de la proposition 6.8(1) que H et $K(H)$ ont les mêmes décompositions arborescentes et donc la même largeur arborescente.

Certaines opérations sur les graphes sont difficilement compatibles avec la largeur arborescente. Notons $s(G)$ le nombre de sommets d'un graphe G, et $Diam(G)$ son *diamètre*, défini (pour un graphe supposé connexe) comme la distance maximale de deux sommets. Pour deux graphes disjoints G et H, on note $G \otimes H$ leur union augmentée de toutes les arêtes reliant un sommet de G et un sommet de H. En utilisant en particulier les propositions 6.6 et 6.8(3) on montre que :

$$Min\{s(G), s(H)\} \leq twd(G \otimes H) \leq Min\{s(G) + twd(H), s(H) + twd(G)\}.$$

Le *produit cartésien* de deux graphes $G = (X, A)$ et $H = (Y, B)$ est le graphe $G \times H$ dont $X \times Y$ est l'ensemble des sommets et où $(x, y) - (u, v)$ si et seulement si $x = u$ et $y - v$ (dans H) ou bien $y = v$ et $x - u$ (dans G). Ainsi $G_{n\times m} = P_n \times P_m$. On a donc pour G et H connexes :

$$Min\{Diam(G) + 1, Diam(H) + 1\} \leq twd(G \times H) \leq Min\{(s(G) + 1).twd(H), (s(H) + 1).twd(G)\}.$$

Pour des majorations fines, on consultera [26]. Ainsi, on ne peut pas majorer la largeur arborescente de $G \otimes H$ ni celle de $G \times H$ en fonction des seules largeurs arborescentes de G et de H. On a observé une situation contraire pour les composantes biconnexes.

6.2.3. Mineurs

L'étude des mineurs des graphes est l'une des origines et des motivations de la notion de décomposition arborescente. Nous allons examiner les liens entre ces différentes notions.

Définition 6.2.3.1 : mineurs d'un graphe

On définit d'abord la notion de *graphe quotient*. Soit $G = (X, E)$ un graphe et $\approx$ une relation d'équivalence sur X. Le *graphe quotient* $G/\approx$ est défini comme le graphe $(X/\approx, E)$ avec une relation d'incidence telle que si l'on a $e : x - y$ dans G (ce qui veut dire « l'arête e relie les sommets x et y ») alors $e : [x]_\approx - [y]_\approx$ dans $G/\approx$.

Soit $G = (X, E)$ un graphe et $F \subseteq E$. Le graphe obtenu par *contraction des arêtes de* F est le graphe $G \setminus F = H/\approx$ où $H = (X, E - F)$ et $\approx$ est la relation d'équivalence sur X définie par $x \approx y$ si et seulement si $x = y$ ou il existe une chaîne de x à y dont toutes les arêtes sont dans F. Cette opération peut créer des boucles et des arêtes multiples. Contracter une arête de C_3 (C_n désigne le cycle à n sommets, cf. 6.2.1.1) donne C_2 formé d'une paire d'arêtes ayant les mêmes extrémités, et contracter une de ces arêtes fournit une boucle.

On dit que H est un *mineur de* G si H est isomorphe à $G' \setminus F'$ où G' est un sous-graphe de G et F' un ensemble d'arêtes de G'. On note cela $H \trianglelefteq G$. Si un graphe G est obtenu à partir de H en subdivisant certaines de ses arêtes et en ajoutant d'autres sommets et arêtes, alors $H \trianglelefteq G$.

On notera $\simeq$ l'isomorphisme des graphes et $[G]_\simeq$ la classe d'isomorphie de G, donc le graphe abstrait défini par G. (Voir la section 6.2.1 pour la notion de graphe abstrait.)

Proposition 6.9. *La relation de mineur est un préordre. La relation d'équivalence associée est l'isomorphisme des graphes.*

Preuve. Il est clair que la relation $\trianglelefteq$ est réflexive. Elle est aussi transitive, mais ce n'est pas immédiat d'après la définition. (Voir l'exercice 5). C'est donc un préordre.

Afin de montrer la seconde assertion définissons comme *taille d'un graphe* la somme du nombre de ses arêtes et du nombre de ses sommets.

Si $H \trianglelefteq G$, la taille de H est inférieure ou égale à celle de G. Si de plus $G \trianglelefteq H$, les graphes G et H sont de même taille. D'après la définition 6.2.1.1, en prenant $H \simeq G' \setminus F'$, G' sous-graphe de G, on ne peut avoir égalité des tailles de H et G que si F' est vide et $G' = G$. Mais alors $H \simeq G$. Donc $H \trianglelefteq G \trianglelefteq H$ implique que G et H sont isomorphes. La relation d'équivalence associée au préordre $\trianglelefteq$ est l'isomorphisme des graphes. (Ceci est faux pour les graphes infinis). □

La notion de mineur fournit une caractérisation combinatoire de la planarité des graphes. Le *Théorème de Kuratowski* (dans une formulation voisine due à Wagner) montre qu'un graphe est planaire si et seulement si il ne contient comme mineur ni K_5 ni $K_{3,3}$, qui sont tous deux non planaires. Une direction de la preuve est facile : à partir d'un dessin planaire de G, on peut construire par suppressions et par « contractions progressives » d'arêtes (on

laissera le lecteur formaliser cette dernière notion) un dessin planaire d'un mineur donné de G. Comme les graphes K_5 et $K_{3,3}$ ne sont pas planaires, un graphe planaire ne peut contenir aucun d'entre eux comme mineur. La réciproque est plus difficile.

Parmi les graphes non planaires, les graphes K_5 et $K_{3,3}$ sont minimaux pour la relation $\trianglelefteq$. On dit que ce sont les *obstructions à la propriété de planarité*.

Ce théorème se généralise aux graphes représentables sur des surfaces telles que le tore et le plan projectif. L'ensemble des obstructions à la représentabilité sur le tore est fini mais la liste n'est pas connue exactement. Elle comporte au moins 2200 graphes. Dans le cas du plan projectif, les obstructions sont connues, il y en a 35. Le lecteur consultera le livre de Mohar et Thomassen [13]. Seymour [36] donne pour chaque surface une majoration de la taille des obstructions associées.

La problématique des caractérisations de familles de graphes par mineurs interdits peut être présentée de la façon suivante. Une famille de graphes $\mathcal{F}$ (ou de manière équivalente, une propriété de graphes $\mathcal{P}$, mais nous formulerons les définitions en termes de familles) est *préservée par minoration* si tout mineur d'un graphe de $\mathcal{F}$ est dans $\mathcal{F}$. Si $\mathcal{F}$ est une famille de graphes préservée par isomorphisme et par minoration, on définit l'ensemble $Obst(\mathcal{F})$ de ses *obstructions* par :

$$Obst(\mathcal{F}) = \{[G]_{\simeq} \mid G \notin \mathcal{F} \text{ et pour tout } H, \text{ si } H \lhd G \text{ alors } H \in \mathcal{F}\}.$$

On note $H \lhd G$ si $H \trianglelefteq G$ et H n'est pas isomorphe à G et on dit que H *est un mineur propre de* G. Alors la famille $\mathcal{F}$ est caractérisée en termes de ses obstructions par :

$$\mathcal{F} = \{G \mid \text{pour tout graphe } H \trianglelefteq G,\ [H]_{\simeq} \notin Obst(\mathcal{F})\}.$$

Le résultat majeur de cette théorie est le *théorème des mineurs de graphes* de Roberston et Seymour (*Graph Minor Theorem*) qui dit que dans un ensemble infini de graphes, il en existe toujours deux comparables pour le préordre $\trianglelefteq$ ([33]). Il en résulte que l'ensemble $Obst(\mathcal{F})$ est toujours fini. C'est un résultat important sur la structure des graphes, qui a des conséquences également importantes en termes de complexité. Pour tout graphe simple G à n sommets, on peut vérifier en temps $O(n^3)$ si un graphe fixé H est isomorphe à un mineur de ce graphe ([32]). Il en résulte que toute famille $\mathcal{F}$ préservée par isomorphisme et minoration possède un algorithme d'appartenance en $O(n^3)$.

Remarque. Si $\mathcal{E}$ est un ensemble de graphes, fini ou infini et $\mathcal{F}$ est l'ensemble des graphes G tels qu'aucun graphe $H \trianglelefteq G$ n'est isomorphe à un graphe de $\mathcal{E}$, on vérifie sans peine en utilisant les définitions que $Obst(\mathcal{F})$ est l'ensemble :

$\{[G]_{\simeq} \mid G \in \mathcal{E}$ et aucun mineur propre de G n'est isomorphe à un graphe de $\mathcal{E}\}$.

En termes intuitifs, $Obst(\mathcal{F})$ est obtenu en prenant dans $\mathcal{E}$ des graphes non isomorphes et minimaux pour la minoration propre.

Nous allons maintenant examiner le comportement des notions de largeur arborescente et de largeur linéaire lors du passage d'un graphe à l'un de ses mineurs.

PROPOSITION 6.10. *(1) Si H est un mineur de G alors $twd(H) \leq twd(G)$ et $pwd(H) \leq pwd(G)$.*

(2) Si H est obtenu en contractant une arête de G alors $twd(H) \leq twd(G) \leq twd(H)+1$ et $pwd(H) \leq pwd(G) \leq pwd(H)+1$.

PREUVE. (1) Avec la proposition 6.6 et la transitivité de la relation de mineur, il suffit d'examiner le cas où H est obtenu par contraction d'une arête e. On construit à partir d'une décomposition (T, g) de G une décomposition (T, h) de H en supprimant e de la boîte où elle figure et en remplaçant chaque sommet de $g(u)$ par son image dans H (sa classe pour l'équivalence associée à la contraction de e).

Vérifions la condition (4) pour le sommet x issu de la contraction de e. L'ensemble des nœuds u de T tels que $x \in h(u)$ est la réunion de deux sous-arbres de T qui ont en commun au moins le nœud v tel que $g(v)$ contient e ; il est donc connexe. Les autres conditions sont évidentes. La largeur de la décomposition ne peut que diminuer.

(2) Inversement, soit à construire une décomposition (T, g) de G à partir d'une décomposition (T, h) de $H = G \setminus e$ où $e : x - y$. On considère que la contraction de e supprime y et redirige vers x les arêtes incidentes à y. On obtient g en ajoutant y à chaque boîte de (T, h). On place les arêtes de G incidentes à x et/ou à y dans les boîtes appropriées, de manière à satisfaire les conditions (2) et (3) de la définition 6.2.1.1. □

Les familles $TWD(\leq k)$ et $PWD(\leq k)$ sont préservées par isomorphisme et par minoration. La famille $TWD(\leq k)$ est donc caractérisée par des obstructions en nombre fini. Mais elles ne sont connues que dans trois cas. Ainsi $Obst(TWD(\leq 1)) = \{K_3\}$, $Obst(TWD(\leq 2)) = \{K_4\}$. Ces deux familles contiennent respectivement les arbres et les *graphes séries-parallèles* (définis dans la section 6.4.1). Enfin $Obst(TWD(\leq 3))$ est constitué de K_5 et de 3 graphes à 6,8 et 10 sommets (voir [1]).

On peut néanmoins en dire un peu plus sur les obstructions de $TWD(\leq k)$. On a mentionné que $twd(G_{k\times k}) = k$. Il en résulte qu'un graphe H dans $TWD(\leq k)$ n'a pas $G_{(k+1)\times(k+1)}$ pour mineur. Donc $G_{(k+1)\times(k+1)}$ ou l'un de ses mineurs, nécessairement planaire, fait partie des obstructions de $TWD(\leq$

k). Les obstructions de $TWD(\leq k)$ comportent donc au moins un graphe planaire.

Il est clair que $twd(H) \geq n$ n'implique pas $G_{n\times n} \trianglelefteq H$: il suffit de considérer $H = K_{n+1}$. Par contre $twd(H) \geq 2^{9n^5}$ implique $G_{n\times n} \trianglelefteq H$. Il s'agit d'un résultat difficile dont plusieurs preuves ont été publiées. Du fait que tout graphe planaire est mineur d'une grille $G_{n\times n}$ pour n assez grand (exercice 5), il résulte que si une famille de graphes est telle qu'un graphe planaire P n'est un mineur d'aucun de ses graphes, alors elle est de largeur arborescente bornée. En effet si P est un graphe planaire simple et sans boucle et H est un graphe qui ne contient pas P comme mineur, alors $twd(H) \leq 20^{2(2s(P)+4e(P))^5}$ où $s(P)$ est le nombre de sommets de P, et $e(P)$ son nombre d'arêtes ([34]).

Des résultats similaires s'appliquent aux familles $PWD(\leq k)$ qui sont fermées par isomorphisme et par minoration. Les arbres sont de largeur linéaire non bornée (voir l'exercice 1). Réciproquement, si un graphe H ne contient pas comme mineur une forêt F (union disjointe d'arbres) telle que $s(F) \geq 3$ alors $pwd(H) \leq s(F) - 2$ et cette borne est optimale (une preuve due à Diestel est donnée dans [9]).

L'utilisation de ces résultats pour la construction d'algorithmes suppose que l'on connaisse les ensembles $Obst(\mathcal{F})$ pour les classes $\mathcal{F}$ considérées. Or la preuve du théorème des mineurs de graphes ne fournit aucune méthode permettant cette construction.

Pour les classes $PWD(\leq k)$ et $TWD(\leq k)$ cette construction est théoriquement possible. En effet les graphes de $Obst(PWD(\leq k))$ et de $Obst(TWD(\leq k))$ ont respectivement, au plus 2^{ak^4} arêtes et au plus $exp(2, bk^5)$ arêtes avec $exp(1, n) = 2^n$ et $exp(i + 1, n) = 2^{exp(i,n)}$ ([Lag]). Il existe donc des « algorithmes » qui à tout entier k associent les ensembles finis de graphes $Obst(PWD(\leq k))$ et $Obst(TWD(\leq k))$. Ils consistent à énumérer tous les graphes sans sommets isolés dont les nombres d'arêtes sont au plus les valeurs ci-dessus, et à tester pour chacun d'eux s'il appartient à $Obst(PWD(\leq k))$ ou à $Obst(TWD(\leq k))$, ce qui est possible en utilisant les définitions. Ces procédures sont des preuves de calculabilité et non des algorithmes utilisables.

Pour les obstructions des familles de graphes dessinables sans croisement d'arêtes sur les surfaces on dispose grâce au résultat de Seymour [36] d'un résultat semblable de calculabilité théorique.

D'autre part, les ensembles d'obstructions sont en général très grands. Ainsi l'ensemble $Obst(PWD(\leq k))$ contient plus de $(k!)^2$ arbres et ces arbres ont tous $(5.3^k - 1)/2$ sommets ([38]).

6.2.4. Decompositions particulières et algorithmes

Un *graphe triangulé* (*chordal graph*) est un graphe non orienté, simple, connexe tel que tout cycle de longueur au moins 4 a une corde (donc sans

cycle induit C_n pour $n > 3$). Ce sont aussi les graphes simples et connexes qui ont une décomposition arborescente dont chaque boîte induit une clique.

Ces graphes ont d'autres caractérisations : un graphe est triangulé si et seulement si c'est le graphe des intersections des éléments d'un ensemble de sous-graphes connexes d'un arbre. Un graphe est triangulé si et seulement si il possède un ordre d'élimination qui est *parfait*, c'est-à-dire une indexation des sommets : $x_1, x_2, ..., x_n$ telle que les sommets x_j pour $j < i$, adjacents à un sommet x_i induisent une clique. Pour tout G triangulé, on a $twd(G) = \omega(G) - 1$ où $\omega(G)$ est l'ordre (nombre de sommets) maximum d'une clique. Un *k-arbre* (*k-tree*) est un graphe triangulé dont toutes les cliques maximales pour l'inclusion sont d'ordre $k + 1$. Ces graphes peuvent être construits récursivement de la façon suivante : le k-arbre minimal est K_{k+1}, il a $k + 1$ sommets. On définit un k-arbre à n sommets en ajoutant à un k-arbre G à $n - 1$ sommets un sommet supplémentaire et k arêtes reliant ce sommet aux sommets d'une k-clique de G. (Voir [1]). Tout k-arbre possède un ordre d'élimination de type k (cf. la section 6.2.1) qui est parfait. Un *k-arbre partiel* (*partial k-tree*) est un sous-graphe d'un k-arbre.

Proposition 6.11. *(1) Pour un graphe simple G, $twd(G) = k$, où $k + 1$ est la valeur minimum de $\omega(H)$ pour un graphe triangulé H qui contient G comme sous-graphe.*

(2) Pour un graphe simple G, $twd(G) = k$, où k est l'entier minimal tel qu'il existe un k-arbre H qui contient G comme sous-graphe. Donc $twd(G) \leq k$ si et seulement si G est un k-arbre partiel.

Le terme de largeur arborescente s'est imposé. La définition correspondante a l'avantage sur celle de k-arbre partiel de ne pas concerner que les graphes simples et de s'étendre sans peine aux hypergraphes.

Preuve. (1) Conséquence immédiate des définitions.

(2) En ajoutant si besoin des boîtes intermédiaires et des sommets dans les boîtes, on peut toujours transformer une décomposition arborescente d'un graphe G de largeur k en une décomposition arborescente du même graphe dont toute boîte a $k + 1$ sommets et telle que deux boîtes voisines ont en commun exactement k sommets. On ajoute alors des arêtes pour que chaque boîte induise une clique et l'on obtient ainsi un k-arbre H dont G est un sous-graphe. □

Décompositions arborescentes soumises à des conditions supplémentaires

Il peut être utile d'imposer des conditions supplémentaires aux décompositions arborescentes pour certaines applications algorithmiques, quitte à perdre l'optimalité. On peut considérer que les décompositions

linéaires entrent dans ce cadre. La proposition suivante regroupe quelques résultats à ce sujet.

Proposition 6.12. *1) Tout arbre à n nœuds possède une décomposition arborescente de largeur 2 dont l'arbre est de diamètre au plus $3.log(n)$; tout graphe à n sommets possède une décomposition arborescente de largeur au plus $3.twd(G)+2$ dont l'arbre est de diamètre au plus $O(log(n))$.*

2) Tout graphe G a une décomposition arborescente connexe de largeur au plus $twd(G)$.

3) Tout graphe G a une décomposition arborescente mengerienne de largeur au plus $twd(G)$.

4) Il existe une fonction f telle que tout graphe G a une décomposition arborescente de type « domino » de largeur au plus $f(twd(G), deg(G))$ où $deg(G)$ désigne le degré maximum de G.

Références : 1) Voir [19] et [25]. 2) Voir Fraignaud et Nisse [27] ; une décomposition (T, f) est *connexe* si pour tous noeuds u et v adjacents dans T, le sous-graphe de G induit par les sommets des boîtes de T associées aux nœuds accessibles à partir de u par une chaîne qui évite v est connexe. 3) Pour la définition très technique de la notion de décomposition arborescente *mengerienne*, qui concerne les chaînes disjointes comme dans le théorème de Menger, et la preuve du résultat dû à Thomas, le lecteur consultera [9]. 4) Voir [1] pour ce résultat dû à Bodlaender et Engelfriet ; une décomposition est de *type « domino »* si aucun sommet n'appartient à plus de 2 boîtes. Il n'est pas possible de remplacer f par une fonction qui ne dépend que de $twd(G)$.

Les applications algorithmiques que nous exposerons plus loin nécessitent que les graphes considérés soient fournis avec des décompositions arborescentes de largeur la plus petite possible. La construction de telles décompositions est donc un élément important. La proposition qui suit ne comporte que quelques résultats parmi beaucoup d'autres.

Proposition 6.13. *1) Les problèmes consistant à décider si un couple donné (G, k) vérifie $twd(G) \leq k$ ou $pwd(G) \leq k$ sont NP-complets.*

2) Le problème consistant à décider si un couple donné (G, k) vérifie $twd(G) \leq k$ est polynomial si G est de l'un des types suivants : un graphe triangulé ou son complément, un cographe, un graphe d'intervalles *(*interval graph*) ou un* graphe d'intersection des cordes d'un cercle *(*circle graph*).*

3) Pour chaque k, les problèmes consistant à décider si un graphe simple donné G vérifie $twd(G) \leq k$ ou $pwd(G) \leq k$ sont linéaires en le nombre de sommets de G.

4) Pour chaque k il existe un algorithme en temps $O(s(G).\log(s(G)))$ qui, pour un graphe simple donné G ou bien répond que $twd(G) > k$, ou bien produit une décomposition arborescente de largeur au plus $3k + 2$.

5) Il existe un algorithme polynomial qui, pour tout graphe G, construit une décomposition arborescente de ce graphe de largeur $O(twd(G).log(twd(G)))$.

PREUVE. Pour les références précises on consultera [2]. Pour les graphes triangulés et les graphes d'intersection des cordes d'un cercle, le problème $pwd(G) \leq k$ est NP-complet. Noter que dans les assertions 1) et 2), l'entier k fait partie des données. Le résultat 3) est dû à Bodlaender [17]. Le cas de la largeur arborescente est exposé dans [9]. La complexité est de l'ordre de $s(G).2^{32k^3}$ et cet algorithme n'est pas pratiquement utilisable. Le résultat 4) est dû à Reed. Contrairement au précédent, l'algorithme est effectivement programmable et utilisable. Si le graphe G considéré a une largeur arborescente comprise entre k et $3k+2$, l'algorithme peut donner l'une ou l'autre des deux réponses. Dans les assertions 3) et 4) on construit pour chaque entier k un algorithme spécifique, bien sûr d'une manière uniforme par rapport à k. Le résultat 5) est du à Bouchitté et al. [21]. L'algorithme est polynomial mais n'est pas pratiquement programmable, et la décomposition arborescente construite est de largeur au plus $twd(G)(560 + 80.log_2(twd(G)))$. □

Les décompositions non optimales, comme celles produites par l'assertion 4) sont intéressantes pour la construction d'algorithmes linéaires en temps. Bodlaender et Kloks [20] montrent que pour chaque k, il existe un algorithme polynomial qui pour un graphe donné de largeur arborescente au plus k calcule la largeur linéaire de ce graphe et fournit une décomposition linéaire optimale de ce graphe. Bodlaender passe en revue dans [3] les résultats algorithmiques relatifs aux décompositions arborescentes et discute leurs possibilités réelles d'implantation. On ne sait pas si le calcul de la largeur arborescente d'un graphe planaire peut se faire en temps polynomial.

6.3. Ecriture algébrique des décompositions arborescentes

Cette section introduit des *expressions algébriques* (ou *termes* au sens de la théorie des systèmes de réécriture) qui définissent les graphes de largeur arborescente au plus k. Cette démarche offre plusieurs avantages : on peut représenter linéairement et sans ambiguïté les graphes autrement qu'en listant les sommets et les arêtes ; cette représentation est structurée en ce sens que la syntaxe reflète la structuration des graphes en termes de composition (par recollements) de graphes de base en nombre fini à isomorphisme

près ; elle permet des preuves et des calculs par induction sur la structure des termes ; elle permet de définir des grammaires de graphes en évitant complètement les complications techniques liées aux réécritures de graphes. Les applications aux grammaires de graphes ne sont pas présentées dans ce chapitre.

Définition 6.3.1.1. : graphes avec sources

On va considèrer des graphes qui peuvent avoir des boucles et des arêtes multiples, et pour simplifier la présentation, non-orientés. La généralisation aux graphes orientés sera immédiate.

Afin de distinguer certains sommets, on utilise un ensemble dénombrable $\mathcal{C}$ d'étiquettes. Un *graphe avec sources* est un couple G constitué d'un graphe (X, E) et d'une bijection $\mathbf{src}_G : C \longrightarrow S$ où C est une partie finie de $\mathcal{C}$ et S une partie de X. Les sommets de S sont les *sources* de G. Le *nom d'une source* s est l'étiquette c telle que $\mathbf{src}_G(c) = s$. On dira encore que s est la *c-source* de G. L'ensemble C noté $\tau(G)$ est le *type* de G. Un graphe dont le type est vide est un graphe sans sources.

Un *isomorphisme de graphes avec sources* doit préserver les noms des sources de manière évidente. Un *sous-graphe* H d'un graphe avec sources G a comme sources les sommets qui sont des sources de G, et leurs noms sont les mêmes dans H et dans G. Ainsi $\tau(H) \subseteq \tau(G)$.

On appellera *graphe abstrait avec sources* une classe d'isomorphisme d'un graphe avec sources (cf. la section 6.2.1). La distinction entre graphe abstrait et graphe concret, qui alourdit certains énoncés est nécessaire car les expressions algébriques que nous allons définir vont définir des graphes abstraits.

On désigne par $\mathcal{G}$ l'ensemble des graphes abstraits avec sources, et par $\mathcal{G}_C$ le sous-ensemble de ceux qui sont de type C.

Définition 6.3.1.2. : décomposition et composition parallèles des graphes avec sources.

Soit G un graphe avec sources. On écrit $G = H//K$ si H et K sont des sous-graphes de G tels que : H et K n'ont pas d'arête en commun, G est l'union de H et de K et les sommets communs à H et à K sont des sources de G (et donc aussi de H et K, avec les mêmes étiquettes car H et K sont des sous-graphes de G). Il en résulte que :

$$\tau(G) = \tau(H) \cup \tau(K) \tag{1}$$

et que les sommets communs à H et à K sont les c-sources pour c dans $\tau(H) \cap \tau(K)$. On dit que G est la *composition parallèle* de H et de K.

Nous venons de définir un *opérateur de décomposition* d'un graphe en deux sous-graphes sans arêtes communes. On va généraliser cette notion et faire de $//$ un *opérateur de composition parallèle* qui s'applique à tout couple de graphes.

Si H et K sont des graphes avec sources non nécessairement disjoints, on construit $G = H//K$ en définissant H' et K', copies disjointes de H et K respectivement et en « recollant » H' et K' en identifiant leurs sources de mêmes noms. Formellement, on définit G comme le quotient du graphe $H' \cup K'$ par la relation d'équivalence $x \approx y :\Longleftrightarrow x = y$ ou $\{x, y\} = \{\mathbf{src}_{H'}(c), \mathbf{src}_{K'}(c)\}$ pour quelque c dans $\tau(H) \cap \tau(K)$. Le graphe G n'est défini qu'à isomorphisme près, car les copies disjointes H' et K' sont quelconques. Si l'on veut être très précis, on peut prendre $H' = (X \times \{1\}, E \times \{1\})$ avec les incidences évidentes pour $H = (X, E)$ (c'est-à-dire $(e, 1) : (x, 1) - (y, 1)$ dans H' si et seulement si $e : x - y$ dans H) et de même $K' = (Y \times \{2\}, F \times \{2\})$ si $K = (Y, F)$.

Une écriture de la forme $G = H//K$ peut être lue de deux manières : ou bien G est un graphe concret décomposé en H et K, ou bien G est un graphe abstrait, composition parallèle de deux graphes H et K, concrets ou abstraits. Plutôt que d'alourdir les notations en distinguant formellement un graphe concret et le graphe abstrait correspondant, on utilisera une seule notation et le contexte lèvera les ambiguïtés éventuelles. L'écriture $H//H$ n'a qu'une lecture possible, celle qui considère // comme un opérateur de composition sauf si H ne comporte que des sources et aucun arc ou arête auquel cas, $H = H//H$.

On peut voir l'opération // comme une généralisation aux graphes de la concaténation des mots. Cette opération est commutative, ce qui n'est pas le cas de la concaténation.

Définition 6.3.1.3 : opérations de manipulation des sources

On écrit $G = \mathbf{fg}_A(H)$ si G est le même graphe que H avec une fonction $\mathbf{src}_G$ qui est la restriction de $\mathbf{src}_H$ à l'ensemble $\tau(H) - A$, notée $\mathbf{src}_H \restriction (\tau(H) - A)$ où A est une partie finie de $\mathcal{C}$. On a donc :

$$\tau(\mathbf{fg}_A(H)) = \tau(H) - A \tag{2}$$

Cette opération est nommée *oubli de sources* car les a-sources (pour a dans A) sont « oubliées » en tant que sources mais les sommets correspondants existent toujours comme sommets « ordinaires ». La notation **fg** fait référence aux *foncteurs d'oubli* (*forgetful functors*) en théorie des catégories. On notera $\mathbf{fg}_a$ l'opération $\mathbf{fg}_{\{a\}}$. Si $\tau(H) \cap A = \varnothing$ alors $\mathbf{fg}_A(H) = H$.

On écrit $G = \mathbf{ren}_h(H)$ si G est le même graphe que H avec $\mathbf{src}_G = \mathbf{src}_H \circ h^{-1}$ où h est une bijection : $A \longrightarrow A$, A est une partie finie de $\mathcal{C}$, et h est étendue en l'identité en dehors de A. Si l'on impose que h définit une permutation sans point fixe de A, alors A est associé à h de manière unique et cette condition minimise la taille de l'ensemble A qui sert à définir h. On a de toutes manières :

$$\tau(\mathbf{ren}_h(H)) = h(\tau(H)) \tag{3}$$

Cette opération est un *renommage de sources* car la a-source de H devient la $h(a)$-source de G. Pour faciliter l'écriture, on notera $\mathbf{ren}_{a \longleftrightarrow b}$ (ou

indifféremment $\mathbf{ren}_{b\longleftrightarrow a}$) l'opération $\mathbf{ren}_h$ lorsque $h(a) = b, h(b) = a$ et $h(c) = c$ pour $c \neq a, b$.

Les opérations $\mathbf{fg}_A$ et $\mathbf{ren}_h$ forment un ensemble dénombrable car elles sont spécifiées par des sous-ensembles finis A de l'ensemble dénombrable $\mathcal{C}$, ou par des permutations de parties finies de $\mathcal{C}$. D'autre part, elles s'appliquent à tous les graphes avec sources, quels que soient leurs types. Les opérations d'oubli et de renommage de sources s'appliquent aux graphes concrets et donc aussi aux graphes abstraits, car si G et H sont deux graphes isomorphes, $\mathbf{fg}_A(G)$ et $\mathbf{fg}_A(H)$ sont isomorphes, aussi bien que $\mathbf{ren}_h(G)$ et $\mathbf{ren}_h(H)$.

Pour construire des graphes au moyen de ces opérations, on utilisera les constantes suivantes : $\mathbf{a}$, $\mathbf{a}^{bcl}$, $\mathbf{ab}$ qui désignent respectivement un sommet isolé qui est une a-source, un sommet avec une boucle qui est une a-source, et une arête dont les extrémités sont une a-source et une b-source, ceci pour tous a, b avec $b \neq a$ dans $\mathcal{C}$. Ces constantes désignent les graphes abstraits correspondants, de types respectifs $\{a\}, \{a\}$ et $\{a, b\}$.

Les définitions présentées ici simplifient la présentation de [6] en ce qu'elles évitent d'utiliser des *F-algèbres à plusieurs sortes d'objets* ([5]). Cette simplification est due au fait que $G//H$, $\mathbf{fg}_A(H)$ et $\mathbf{ren}_h(H)$ sont bien définis quels que soient les types de G et de H.

L'opération $//$ est associative et commutative. On utilisera donc la notation infixée sans parenthèses. Autrement dit, on considérera que $G//(H//K)$, $(G//H)//K$ et $(G//K)//H$ sont le même terme que l'on notera aussi bien $G//H//K$ que $G//K//H$. D'autre part, toute opération $\mathbf{ren}_h$ peut s'écrire comme une composition finie d'opérations du type $\mathbf{ren}_{a\longleftrightarrow b}$.

On notera $\mathcal{F}$ l'ensemble de ces constantes et de ces opérations et, pour tout C fini, $C \subset \mathcal{C}$, on notera $\mathcal{F}_C$ son sous-ensemble fini défini comme :

$$\{//, \mathbf{fg}_A, \mathbf{ren}_h, \mathbf{a}, \mathbf{a}^{bcl}, \mathbf{ab} \mid A \subseteq C, a, b \in C, \ \ h \text{ est une permutation de } \mathcal{C} \text{ qui est l'identité en dehors de } C\}$$

Rappellons quelques définitions d'algèbre universelle. Une *signature fonctionnelle* est un couple formé d'un ensemble fini ou dénombrable F de *symboles de fonctions* et d'une fonction $\rho : F \longrightarrow \mathbf{N}$ qui associe à chacun d'eux une *arité*, c'est-à-dire un entier positif ou nul qui définira son nombre d'arguments. Un symbole d'arité 0 est une *constante*. Une *F-algèbre* (où F est une signature fonctionnelle) est un objet $\mathbf{M} = (M, (f_{\mathbf{M}})_{f\in F})$, dont M est le domaine et dont, pour chaque $f \in F$, la composante f_M est une fonction totale $M^{\rho(f)} \longrightarrow M$. Chaque constante désigne un élément de M. Un monoïde, un groupe, un anneau, un corps sont des F-algèbres, pour des signatures bien connues.

On notera $T(F)$ l'ensemble des termes finis écrits avec ces symboles et bien formés relativement à leurs arités. Tout terme $t \in T(F)$ a une valeur dans le domaine M, notée $t_{\mathbf{M}}$. On l'obtient en évaluant le terme t dans la

F-algèbre $\mathbf{M}$. Cette évaluation utilise les fonctions $f_{\mathbf{M}}$ comme interprétations des symboles f de fonctions et de constantes.

On a donc défini sur l'ensemble $\mathcal{G}$ des graphes abstraits avec sources une structure de $\mathcal{F}$-algèbre pour une signature fonctionnelle dénombrable $\mathcal{F}$. Si l'on se restreint aux graphes avec sources de types inclus dans un ensemble fini C, on n'utilise plus qu'un nombre fini d'opérations, mais cet ensemble fini ne peut pas engendrer tous les graphes finis ainsi qu'il résulte du théorème 6.15 ci-dessous. Tout terme de $T(\mathcal{F})$ appartient de fait à un ensemble $T(\mathcal{F}_C)$ où C est fini. Les termes de $T(\mathcal{F}_C)$ ne peuvent définir que des graphes de type contenu dans C. Un terme t dans $T(\mathcal{F})$ a pour valeur un graphe abstrait $t_{\mathcal{G}}$ que l'on notera $val(t)$ et que l'on nommera sa *valeur*. On dira que t est un *arbre syntaxique* du graphe $val(t)$. Ce graphe abstrait peut être défini comme la classe d'isomorphisme d'un graphe concret $G(t)$ construit au moyen des occurrences dans t des symboles de constantes. Ces occurrences servent à définir les sommets et les arêtes (ou les arcs) de $G(t)$. Cette construction fait l'objet de l'exercice 6. Le type du graphe $G(t)$, donc de $val(t)$, peut être déterminé à partir du terme t en utilisant les règles de typage (1), (2) et (3) indiquées plus haut.

Au moyen des opérations de $\mathcal{F}$, on peut en définir d'autres appelées *opérations dérivées* (*derived operations*), analogues à des « macros », qui permettront d'écrire des expressions plus concises et plus lisibles. Par exemple, la *composition en série* de deux graphes G et H à deux sources d'étiquettes 1 et 2 peut se définir ainsi :

$$G \bullet H = \mathbf{fg}_3(\mathbf{ren}_{2\longleftrightarrow 3}(G)//\mathbf{ren}_{1\longleftrightarrow 3}(H)).$$

Cette opération peut aussi être définie par $G \bullet H = \mathbf{fg}_4(\mathbf{ren}_{2\longleftrightarrow 4}(G)//\mathbf{ren}_{1\longleftrightarrow 4}(H))$. Notons que les graphes $\mathbf{fg}_3(\mathbf{ren}_{2\longleftrightarrow 3}(G)//\mathbf{ren}_{1\longleftrightarrow 3}(H))$ et $\mathbf{fg}_4(\mathbf{ren}_{2\longleftrightarrow 4}(G)//\mathbf{ren}_{1\longleftrightarrow 4}(H))$ sont bien définis (mais pas égaux) pour tous graphes avec sources G et H, mais on ne s'intéresse qu'au cas où G et H sont de type $\{1, 2\}$.

Une opération dérivée d'arité p est définie par un terme dans $T(\mathcal{F}, \{x_1, ..., x_p\})$ où $x_1, ..., x_p$ sont des variables désignant des graphes.

On utilisera $[1, n] = \{1, ..., n\}$ comme ensemble d'étiquettes de sources.

LEMME 6.14. *Tout graphe à n sommets est défini par un terme de $T(\mathcal{F}_{[1,n]})$.*

PREUVE. On identifie les sommets du graphe donné G supposé sans sources aux entiers $1, ..., n$. On utilise une étiquette $\mathbf{x}$ pour chaque sommet x de G, et on définit G par le terme :

$$t = \mathbf{fg}_{[1,n]}(\mathbf{x}_1\mathbf{y}_1//\mathbf{x}_2\mathbf{y}_2//...//\mathbf{x}_m\mathbf{y}_m//\mathbf{z}_1//... //\mathbf{z}_p//\mathbf{w}_1^{bcl}//\mathbf{w}_2^{bcl}//...//\mathbf{w}_q^{bcl})$$

où les arêtes sont les $x_i - y_i$, les sommets isolés sont les z_i, et les sommets avec des boucles sont les w_i. Si G est un graphe avec sources, la construction est semblable avec un choix approprié des étiquettes de sources et le remplacement de $\mathbf{fg}_{[1,n]}$ par quelque $\mathbf{fg}_A$. □

Exemple. Si G est le graphe de sommets 1,2,3,4 avec deux arêtes entre 1 et 2, une arête entre 2 et 3, deux boucles incidentes à 3 et un sommet isolé 4, le terme t résultant de cette construction est $t = \mathbf{fg}_{[1,4]}(\mathbf{12}//\mathbf{12}//\mathbf{23}//\mathbf{4}//\mathbf{3}^{bcl}//\mathbf{3}^{bcl})$.

Cette construction équivaut à définir le graphe par une liste d'arêtes et de sommets, et ne comporte aucune structuration. Elle représente le graphe de la figure 6.1 par un terme utilisant 16 étiquettes, alors que la décomposition de ce graphe définie par les figures 6.2 et 6.3 permet, comme on va le démontrer, de le définir par un terme n'utilisant que 4 étiquettes.

Une *décomposition arborescente d'un graphe avec sources* est une décomposition arborescente de ce graphe dont une boîte contient toutes les sources. La notion de largeur arborescente (encore notée *twd*) en résulte. Dans le cas d'une décomposition *linéaire*, on demande en plus que les sources soient toutes dans l'une des boîtes situées aux extrémités de la chaîne. Ces définitions s'appliquent de manière évidente à des graphes abstraits.

Théorème 6.15. *Un graphe avec sources est de largeur arborescente au plus $k-1$ si et seulement si il est la valeur d'un terme $t \in T(\mathcal{F}_C)$ pour C de cardinalité k.*

Preuve. Soit G un graphe défini par un terme $t \in T(\mathcal{F}_C)$ pour C de cardinalité k. On va définir par induction sur la structure de t une décomposition arborescente enracinée (T, f) de G de largeur $k-1$, telle que la boîte associée à sa racine est l'ensemble des sources (éventuellement l'ensemble vide). Les boîtes vides seront supprimées si nécessaire dans une étape finale de la construction (voir définition 6.2.1.1).

Si $t = \mathbf{a}, \mathbf{a}^{bcl}$ ou $\mathbf{ab}$, on prend T réduit à un unique nœud qui est la racine. La boîte correspondante contient le ou les sommets et, dans les deux derniers cas, l'arête de G.

Si $t = \mathbf{ren}_h(t')$ et la décomposition (T', f') a été construite pour le graphe G' défini par t' conformément à l'hypothèse d'induction, on prend $(T, f) = (T', f')$. Ceci est correct car G et G' ne diffèrent que par les noms de leurs sources.

Si $t = \mathbf{fg}_A(t')$ et la décomposition (T', f') a été construite pour le graphe G' défini par t', on construit (T, f) en ajoutant un nouveau nœud r qui sera la racine de T, un arc de r vers la racine r' de T', $f(u) = f'(u)$ si u est un nœud de T' et $f(r)$ est l'ensemble des sources de G. Noter que $f(r) \subseteq f(r')$.

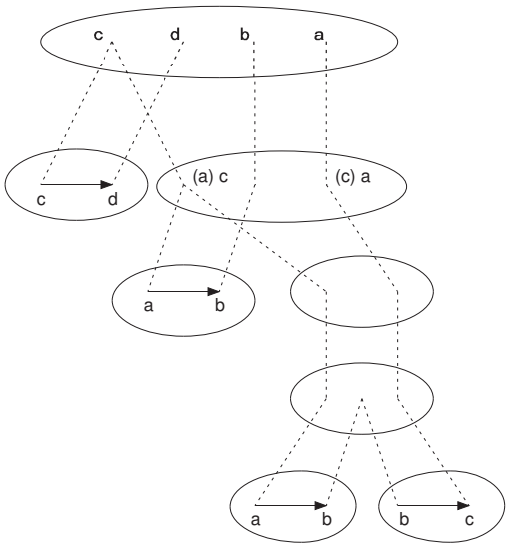

Figure 6.4. – *Décomposition associée à un terme*

Si $t = t_1//t_2$ définit G, on note G_1 et G_2 les sous-graphes de G définis par t_1 et t_2 et, d'après l'hypothèse d'induction, on peut supposer construites des décompositions (T_1, f_1) et (T_2, f_2) de G_1 et G_2 respectivement. On supposera de plus T_1 et T_2 disjoints : s'ils ne le sont pas, on remplace l'un d'eux par une copie isomorphe. On construit T en ajoutant à l'union de T_1 et T_2 un nouveau sommet r qui sera la racine de T et des arcs de r vers les racines r_1 et r_2 de T_1 et T_2. On définit $f(r)$ comme la réunion de $f_1(r_1)$ et de $f_2(r_2)$. C'est bien l'ensemble des sources de G.

On vérifie par induction sur la structure de t que (T, f) est une décomposition arborescente de G. Chacune de ses boîtes, qui est l'ensemble des sources d'un graphe défini par un sous-terme de t, comporte au plus k sommets, puisqu'il n'y a que k étiquettes de sources. Donc (T, f) est de largeur au plus $k - 1$.

La décomposition associée au terme $t = \overrightarrow{\mathbf{cd}}//ren_{a\longleftrightarrow c}(\overrightarrow{\mathbf{ab}}//\mathbf{fg}_b(\overrightarrow{\mathbf{ab}}//\overrightarrow{\mathbf{bc}}))$ est présentée sur la figure 6.4 (cf. exercice 6). Les noms de sources entre parenthèses sont renommés par l'opération $ren_{a\longleftrightarrow c}$.

Réciproquement, soit $G = (X, E)$ un graphe avec sources dont on connaît une décomposition arborescente (T, f) de largeur au plus $k - 1$. Les sources de G sont toutes dans une boîte $f(r)$. On prend le nœud r comme racine de T. D'après le corollaire 6.3, il existe un coloriage $c : X \longrightarrow [1, k]$ tel que deux sommets d'une même boîte ont des couleurs différentes. On va utiliser $[1, k]$ comme ensemble d'étiquettes de sources. La preuve se fait par récurrence sur le nombre de nœuds de T.

Si T est réduit à un seul nœud, le graphe G a au plus k sommets, et on construit un terme de $T(\mathcal{F}_{[1,k]})$ qui le définit en utilisant le lemme 6.14.

Sinon, soit r la racine et $T_1, ..., T_p$ les sous-arbres de T issus de ses fils $n_1, ..., n_p$. Soient $G_1, ..., G_p$ les graphes définis par les décompositions associées aux sous-arbres $T_1, ..., T_p$. (G_i est l'union des graphes associés aux boîtes de T_i). Les sources de ces graphes sont les sommets du graphe (défini par) $f(n_i)$. Plus précisément, x est la j-source de $f(n_i)$ si et seulement si $c(x) = j$. Par hypothèse de récurrence, on peut supposer construits des termes $t_1, ..., t_p$ pour les graphes $G_1, ..., G_p$. Pour chaque i on note A_i l'ensemble des étiquettes des sources de G_i qui ne sont pas dans la boîte racine de T. On définit G'_i comme le graphe $\mathbf{fg}_{A_i}(G_i)$.

Le graphe G est la composition parallèle des graphes $G'_1, ..., G'_p$ augmentée des sommets et des arêtes *propres à* $f(r)$, c'est-à-dire des sommets et des arêtes qui sont dans $f(r)$ mais non dans $G'_1//...//G'_p$. Il en résulte que G est défini par le terme :

$$\mathbf{fg}_{A_1}(t_1)//...//\mathbf{fg}_{A_p}(t_p)//\mathbf{x}_1\mathbf{y}_1//...//\mathbf{x}_m\mathbf{y}_m//\mathbf{z}_1//...//\mathbf{z}_q//\mathbf{w}_1^{bcl}//\mathbf{w}_2^{bcl}//...$$

où $z_1, z_2, ..., z_m$ sont les sommets isolés propres à $f(r)$, $\mathbf{w}_1^{bcl}, \mathbf{w}_2^{bcl}, ...$ définissent les boucles propres à $f(r)$ et $x_1 - y_1, x_2 - y_2, ..., x_m - y_m$ sont les arêtes propres à $f(r)$.

La figure 6.5 illustre cette construction. Elle montre une décomposition arborescente de largeur 2, coloriée au moyen des trois couleurs a, b, c. Ce coloriage satisfait les conditions du corollaire 6.3. Ses boîtes sont numérotées de 1 à 5, et 1 est la racine. On note G_i le graphe défini par la décomposition restreinte au sous-arbre issu du noeud i. On définit des termes t_i qui définissent les graphes G_i, ce qui donne :

$t_5 = \mathbf{bd}//\mathbf{cd}$
$t_4 = \mathbf{ab}//\mathbf{ac}$
$t_2 = \mathbf{a}^{bcl}//\mathbf{cb}//\mathbf{fg}_a(t_4)$
$t_1 = \mathbf{fg}_c(t_2)//\mathbf{cb}//\mathbf{fg}_d(t_5)$.

Le terme t_1 qui définit G est donc
$\mathbf{fg}_c[\mathbf{a}^{bcl}//\mathbf{cb}//\mathbf{fg}_\mathbf{a}(\mathbf{ab}//\mathbf{ac})]//\mathbf{cb}//\mathbf{fg}_d(\mathbf{bd}//\mathbf{cd})$. □

Remarque. Il résulte de cette preuve que l'on peut construire un graphe de largeur arborescente au plus $k-1$ avec k étiquettes, sans utiliser l'opération de renommage de sources. Cette opération est néanmoins très utile. Sans elle, on ne peut pas définir par un terme (donc comme une opération dérivée) la composition en série des graphes de type $\{1, 2\}$ comme le montre l'exemple suivant.

Soit G le graphe défini par le terme $t = \mathbf{12}//\mathbf{fg}_3(\mathbf{13}//\mathbf{32})$. Soit H le graphe $G \bullet G$, défini par le terme $\mathbf{fg}_3(\mathbf{ren}_{2\longleftrightarrow 3}(t)//\mathbf{ren}_{1\longleftrightarrow 3}(t))$. Si l'on s'interdit l'opération de renommage, on peut définir H par le terme $\mathbf{fg}_3(t_1//t_2)$ où t_1 définit le graphe $\mathbf{ren}_{2\longleftrightarrow 3}(G)$ et t_2 définit le graphe $\mathbf{ren}_{1\longleftrightarrow 3}(G)$, en prenant $t_1 = \mathbf{13}//\mathbf{fg}_2(\mathbf{12}//\mathbf{23})$ et $t_2 = \mathbf{32}//\mathbf{fg}_1(\mathbf{31}//\mathbf{12})$.

L'inconvénient est que l'on ne peut pas écrire un terme définissant $G \bullet G'$ en utilisant « tels quels » des termes t et t' qui définissent respectivement G

.

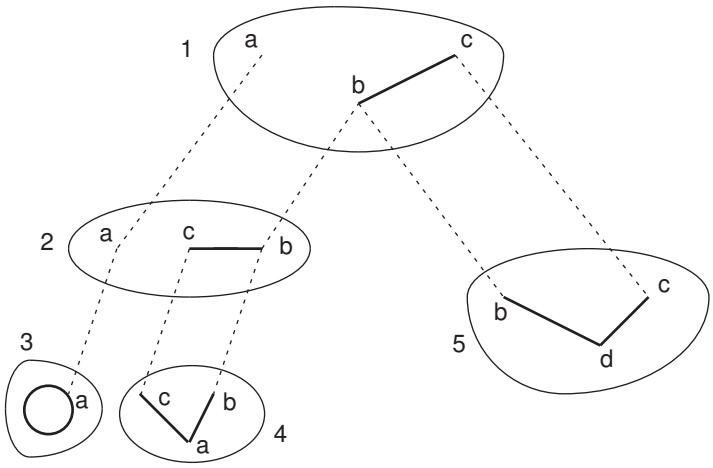

Figure 6.5. – *Une décomposition arborescente coloriée en vue de la construction d'un terme*

et G'. On est obligé de modifier ces termes. D'autre part, dans l'exemple de H défini ci-dessus la syntaxe du terme $\mathbf{fg}_3(t_1//t_2)$ masque le fait intéressant que le même graphe G figure deux fois comme « facteur » de H.

Corollaire 6.16. *Un graphe avec sources est de largeur linéaire au plus $k-1$ si et seulement si il est la valeur d'un terme $t \in T(\mathcal{F}_C)$ pour C de cardinalité k qui est tel que dans tout sous-terme de t de la forme $t_1//.../\!/t_q$ au plus un des termes $t_1, ..., t_q$, n'est pas réduit à une constante.*

Preuve. Soit G défini par un terme t qui satisfait la condition de l'énoncé. Comme pour la preuve du théorème 6.15, on utilise une induction sur la structure de t. On construit une décomposition linéaire et enracinée (T, f) du graphe G dont la racine est de degré sortant 1, (est une extrémité du chemin T), et telle que les sommets de la boîte correspondante sont les sources de G. On a les cas suivants.

Si t est $\mathbf{ren}_h(t')$, $\mathbf{fg}_A(t')$ ou est une constante, la construction est la même que dans la preuve du théorème 6.15.

Autrement $t = t_1//.../\!/t_q$ et $t_2, ..., t_q$ sont des constantes. Utilisant l'hypothèse d'induction, on suppose (T', f') construite pour le graphe G' défini par t_1. On construit (T, f) en ajoutant un nouveau sommet r qui sera la racine de T, un arc de r vers la racine r' de T', on définit $f(u)$ égal à $f'(u)$ si u est un nœud de T' et $f(r)$ comme l'ensemble des sources de G et des arêtes définies par les constantes $t_2, ..., t_q$. Les extrémités de ces arêtes

sont des sources de G et $f(r) \supseteq f(r')$. On obtient une décomposition linéaire enracinée de G dont la racine est de degré sortant 1 et telle que les sommets de la boîte correspondante sont les sources. Donc $pwd(G) \leq k-1$ puisque dans chaque boîte, les sommets qui sont tous des sources sont en nombre au plus k.

Réciproquement, soit (T, f) une décomposition linéaire de G. On lui applique la construction de la preuve du théorème 6.15. Du fait que la décomposition (T, f) est linéaire on a $p = 0$ ou 1. On obtient donc un terme de la forme souhaitée. □

Pour engendrer des graphes orientés, on utilisera la constante $\overrightarrow{\mathbf{ab}}$ qui désigne un arc d'une a-source vers une b-source. Tous les résultats s'étendent de façon immédiate.

6.4. Preuves et calculs inductifs

Cette section montre comment on peut utiliser la structuration des graphes définie par les termes de $T(\mathcal{F})$ qui les représentent pour vérifier des propriétés ou effectuer des calculs sur ces graphes. Il en résultera des méthodes de construction d'algorithmes polynomiaux pour des classes de graphes structurés.

Le calcul des propriétés des circuits électriques peut être considéré comme à l'origine des notions de graphe structuré et de calcul inductif fondé sur la structuration des graphes. Les termes de « composition en série » et de « composition parallèle » y trouvent leur source car ces opérations correspondent à des combinaisons de circuits électriques modélisés par des graphes dont les arêtes sont valuées par des paramètres tels que résistance, inductance ou capacité. Les propriétés globales d'un circuit peuvent être calculées inductivement à partir des propriétés des composants de base et de celles des opérations de composition des circuits. L'ouvrage de Recski [14] comporte plusieurs chapitres sur l'application de la théorie des graphes aux circuits électriques.

6.4.1. Exemples de preuves et de calculs inductifs

Propriétés des graphes série-parallèles

On définit un *k-graphe* comme un graphe avec sources de type $\{1, 2, ..., k\}$.

Les *graphes série-parallèles* sont définis comme les valeurs des termes de $T(\{//, \bullet, \mathbf{12}\})$ où $\bullet$ est la composition en série des 2-graphes (voir la section 6.3). Pour montrer que tout graphe série-parallèle est connexe, il suffit d'observer que $\mathbf{12}$ est connexe, et que si G et H sont des 2-graphes connexes, alors $G//H$ et $G \bullet H$ sont connexes. Ce n'est autre qu'une preuve par induction sur la structure des arbres syntaxiques des graphes série-parallèles.

Pour montrer que tout graphe série-parallèle est planaire, on peut tenter de faire de même, mais on rencontre une difficulté : il n'est pas vrai que si G et H sont des 2-graphes planaires, alors $G//H$ est planaire. Un contre-exemple est fourni par $\mathbf{12}$ et le graphe K_5 moins une arête, les extrémités de l'arête supprimée étant les sources 1 et 2. Il faut raisonner inductivement sur une propriété plus forte que la propriété visée qui est dans le cas présent la planarité. On pourra prendre comme propriété :

« G possède un plongement dans le plan tel que les deux sources sont sur le bord d'une même face », ou encore :

« $G//\mathbf{12}$ est planair ».

On est dans la situation classique où une preuve d'une propriété P se fait par une récurrence sur une propriété de la forme $P \wedge Q$, où Q est une *propriété auxiliaire*. La *Logique du Second-Ordre Monadique* permet de systématiser cette observation et de générer automatiquement (au moins sur le plan théorique) la propriété auxiliaire nécessaire à la conduite d'une preuve inductive. Ce langage est présenté dans [6,22,7].

Recherche d'une 3-coloration.

Prenons l'exemple du problème NP-complet de la recherche d'une 3-coloration d'un graphe. Soit G un graphe de largeur arborescente au plus $k-1$ (k est fixé), donné par un terme de $T(\mathcal{F}_C)$ où C est l'ensemble d'étiquettes $\{1, 2, ..., k\}$. Pour utiliser cette donnée, il suffit de savoir vérifier la 3-colorabilité d'un graphe de la forme $H//K$, $\mathbf{ren}_h(H)$ ou $\mathbf{fg}_A(H)$ en fonction de la 3-colorabilité et de propriétés auxiliaires de H et de K.

On va considérer la 3-colorabilité des graphes avec sources, de type *non vide* inclus dans C. Ne considérer que des graphes ayant au moins une source ne nuit pas à la généralité mais facilite l'écriture. Pour tout graphe $G = (X, E)$, on note $Col(G)$ l'ensemble (éventuellement vide) de ses 3-coloriages, c'est-à-dire des fonctions totales $c : X \longrightarrow \{1, 2, 3\}$ telles que $c(x) \neq c(y)$ pour x adjacent à $y \neq x$. On pourrait calculer $Col(G)$ par induction sur la structure d'un terme qui définit G, mais on aurait ainsi à manipuler des ensembles de taille éventuellement exponentielle en $s(G)$ et l'on n'obtiendrait pas un algorithme efficace.

On va extraire de $Col(G)$ une information de taille bornée en fonction de k (mais k est fixé) qui « passe à l'induction ». A un coloriage c d'un graphe G, on associe sa restriction aux sources, c'est-à-dire l'application $c^{\#} = c \circ \mathbf{src}_G : \tau(G) \longrightarrow \{1, 2, 3\}$ qui associe à chaque étiquette de source la couleur du sommet correspondant. Soit $Col^{\#}(G)$ l'ensemble de ces fonctions $c^{\#}$ pour tous $c \in Col(G)$. Contrairement à $Col(G)$, l'ensemble $Col^{\#}(G)$ est une partie d'un ensemble fini dont la taille (grande certes) ne dépend que de k.

Le point-clé est que l'on peut déterminer $Col^{\#}(H//K)$, $Col^{\#}(\mathbf{fg}_A(H))$ et $Col^{\#}(\mathbf{ren}_h(H))$ en fonction de $Col^{\#}(H)$ et de $Col^{\#}(K)$, ce qui peut se faire au moyen des règles suivantes : $Col^{\#}(H//K) =$

$\{f : \tau(H//K) \longrightarrow \{1, 2, 3\} \mid f \restriction \tau(H) \in Col^{\#}(H), f \restriction \tau(K) \in Col^{\#}(K)\}$

où on rappelle que $f \restriction \tau(H)$ désigne la restriction de la fonction f à l'ensemble $\tau(H)$. On notera ceci plus brièvement :
$Col^{\#}(H//K) = F_{//}(Col^{\#}(H), Col^{\#}(K))$.
De même :
$Col^{\#}(\mathbf{fg}_A(H)) =$
$\{f : (\tau(H) - A) \longrightarrow \{1,2,3\} \mid f = g \restriction (\tau(H) - A), g \in Col^{\#}(H)\}$
ce que l'on notera :
$Col^{\#}(\mathbf{fg}_A(H)) = F_{\mathbf{fg}_A}(Col^{\#}(H))$.
Finalement
$Col^{\#}(\mathbf{ren}_h(H)) =$
$\{g \circ h^{-1} : h^{-1}(\tau(H)) \longrightarrow \{1,2,3\} \mid g \in Col^{\#}(H)\}$ ce que l'on notera :
$Col^{\#}(\mathbf{ren}_h(H)) = F_{\mathbf{ren}_h}(Col^{\#}(H))$.

Une fois connu un terme t de $T(\mathcal{F}_C)$ qui définit $G = val(t)$ on peut calculer l'ensemble $Col^{\#}(G) = Col^{\#}(val(t))$ par induction sur la structure de t :
si $t = t_1//t_2$, on obtient $Col^{\#}(val(t))$ en calculant $F_{//}(Col^{\#}(val(t_1), Col^{\#}(val(t_2)))$,
si $t = \mathbf{fg}_A(t_1)$, on obtient $Col^{\#}(val(t))$ en calculant $F_{\mathbf{fg}_A}(Col^{\#}(val(t_1)))$,
si $t = \mathbf{ren}_h(t_1)$, on obtient $Col^{\#}(val(t))$ en calculant $F_{\mathbf{ren}_h}(Col^{\#}(val(t_1)))$,
et si t est une constante désignant un graphe de base B (à un ou deux sommets) on calcule directement $Col^{\#}(B)$.

Selon que l'ensemble $Col^{\#}(val(t)) = Col^{\#}(G)$ est vide ou non, on obtient la réponse que G n'est pas ou est 3-coloriable respectivement. L'algorithme utilisé est linéaire en temps par rapport à la taille du terme t supposé connu, et pour un nombre d'étiquettes maximal k fixé.

En vue d'une extension de cette méthode à d'autres problèmes (voir la proposition 6.17 ci-dessous), il importe de noter que les informations à calculer inductivement restent dans un ensemble fini (bien que de très grande cardinalité). Cette méthode s'applique à bien d'autres problèmes, en particulier NP-complets tels que la recherche d'un cycle Hamiltonien, et plus généralement à tout problème de graphe que l'on peut exprimer par une formule de la *logique du second-ordre monadique*, c'est-à-dire par une formule logique écrite avec des quantifications existentielles et universelles sur des variables désignant des sommets, des arcs ou arêtes, des ensembles de sommets et des ensembles d'arcs ou arêtes. Nous renvoyons le lecteur à [6] pour la présentation de ce langage.

6.4.2. Ensembles inductifs de propriétés des éléments d'une F-algèbre

Nous nous plaçons dans le cadre des F-algèbres définies à la section 6.3.

Définition 6.4.2.1 : ensembles inductifs de propriétés.

Soit $\mathbf{M} = (M, (f_{\mathbf{M}})_{f \in F})$ une F-algèbre. Une *propriété des éléments de* $\mathbf{M}$ est une fonction totale $P : M \longrightarrow \{Vrai, Faux\}$. Soit $\mathcal{P} = \{P_1, ..., P_m\}$ un ensemble *fini* de propriétés des éléments de $\mathbf{M}$. Pour tout $d \in M$, on pose :
$\overrightarrow{\mathcal{P}}(d) = (P_1(d), ..., P_m(d)) \in \{Vrai, Faux\}^m$.

On dit que $\mathcal{P}$ est *inductif relativement à* F ou F*-inductif* si pour toute fonction $f \in F$ d'arité k, il existe une fonction $f^{\#} : (\{Vrai, Faux\}^m)^k \longrightarrow \{Vrai, Faux\}^m$ telle que pour tous $d_1, .., d_k \in M$,

(1) $$\overrightarrow{\mathcal{P}}(f_{\mathbf{M}}(d_1, ..., d_k)) = f^{\#}(\overrightarrow{\mathcal{P}}(d_1), ..., \overrightarrow{\mathcal{P}}(d_k)).$$

Concrètement, cette condition est vérifiée si pour tous i et f, il existe une fonction booléenne g et des propriétés $P_{1,1}, P_{1,2}, ..., P_{2,1}, ..., P_{k,1}, P_{k,2}, ...$ dans $\mathcal{P}$ telles que pour tous $d_1, ..., d_k \in M$:

(2) $$P_i(f_{\mathbf{M}}(d_1, ..., d_k)) = g(P_{1,1}(d_1), P_{1,2}(d_1), ..., P_{2,1}(d_2), ..., P_{k,1}(d_k), P_{k,2}(d_k), ...)$$

Inversement, si l'on a une fonction $f^{\#}$ qui satisfait (1) et que l'on écrit : $f^{\#}(\overrightarrow{b_1}, ..., \overrightarrow{b_k}) = (f_1^{\#}(\overrightarrow{b_1}, ..., \overrightarrow{b_k}), ..., f_m^{\#}(\overrightarrow{b_1}, ..., \overrightarrow{b_k}))$ pour tous $\overrightarrow{b_1}, ..., \overrightarrow{b_k} \in \{Vrai, Faux\}^m$, alors les égalités (2) sont vérifiées avec :
$P_i(f_{\mathbf{M}}(d_1, ..., d_k)) =$
$f_i^{\#}(P_1(d_1), ..., P_m(d_1), P_1(d_2), ..., P_m(d_{k-1}), P_1(d_k), ..., P_m(d_k))$.

On rappelle que $t_{\mathbf{M}}$ désigne la valeur dans $\mathbf{M}$ d'un terme t.

Proposition 6.17. (6,5) *Soit* $\boldsymbol{M}$ *une* F*-algèbre où* F *est fini et* $\mathcal{P}$ *un ensemble de propriétés* F*-inductif. Pour tout terme* $t \in T(F)$*, on peut calculer* $\overrightarrow{\mathcal{P}}(t_{\mathbf{M}})$ *en temps* $O(\mid t \mid)$.

Preuve. On utilise une définition par induction sur la structure de t comme dans l'exemple du 3-coloriage traité plus haut. On calcule pour tous les sous-termes s de t le vecteur de booléens $\overrightarrow{\mathcal{P}}(s_{\mathbf{M}})$. Le résultat final est obtenu en examinant le vecteur de booléens $\overrightarrow{\mathcal{P}}(t_{\mathbf{M}})$. □

L'exemple traité plus haut du 3-coloriage peut aussi être formulé en termes de propriétés $\mathcal{F}_C$-inductives. On considère l'ensemble des propriétés $P_{D,\gamma}$ où D est une partie non vide de C, γ est une fonction : $D \longrightarrow \{1, 2, 3\}$. On définit $P_{D,\gamma}(G)$ comme $Vrai$ si et seulement si $\tau(G) = D$ et $\gamma \in Col^{\#}(G)$. L'ensemble des propriétés $P_{D,\gamma}$ est fini et il résulte des remarques formulées dans la section 6.4.1 qu'il est $\mathcal{F}_C$-inductif. Le graphe G est 3-coloriable si et seulement si $P_{\tau(G),\gamma}(G) = Vrai$ pour quelque γ.

Ce type de calcul est souvent présenté en termes d'automates d'arbres : consulter [DF] et [TATA] à ce sujet. Cette méthode s'applique aux propriétés et aux fonctions d'optimisation sur les graphes définissables en logique du second-ordre monadique : consulter [6,22,7]. Beaucoup de problèmes NP-complets entrent dans ce cadre.

6.4.3. Mise en œuvre et applications

L'article de Thorup [37] montre que les graphes des programmes *structurés* c'est-à-dire écrits sans GOTOs dans les langages Pascal, Modula-2 et C ont une largeur arborescente au plus 6. Il en résulte des algorithmes polynomiaux d'allocation de registres qui fournissent des allocations utilisant au pire 4 fois le nombre optimum de registres, alors que les problèmes sous-jacents sont NP-complets pour les graphes généraux.

Les allocations de fréquence radio peuvent se formuler en termes de coloriages de graphes. Les problèmes sont presque toujours NP-complets dans les cas les plus généraux. McDiarid et Reed montrent que l'un de ces problèmes est NP-complet même pour les graphes de largeur arborescente au plus 3, mais fournissent un algorithme d'approximation qui est polynomial sur $TWD(\leq k)$, pour chaque k [30].

Bodlaender passe en revue dans [1,3] d'autres applications : calcul numérique sur des matrices creuses, problèmes d'inférence en présence d'incertitudes, dessins de graphes, biochimie (la quasi totalité des graphes représentant les molécules biochimiques sont de largeur arborescente au plus 3 [39]).

Le logiciel MONA développé par N. Klarlund [28] construit des automates finis déterministes associés aux formules de la logique du second-ordre monadique portant sur des mots et des arbres binaires. L'utilisation de *diagrammes de décision booléens* (*Boolean decision diagrams*) pour représenter ces automates permet de faire face au très grand nombre d'états.

L'ouvrage d'algorithmique de Kleinberg et Tardos [12] traite des décompositions arborescentes.

6.5. Extensions de cette approche

Ce chapitre n'est qu'une introduction à un domaine de la théorie des graphes en constante expansion. Parmi les nombreuses directions de recherche on pourra citer sans prétendre être exhaustif, la recherche des valeurs les plus précises possible de la largeur arborescente pour des graphes dotés de propriétés particulières ou construits de telle ou telle façon, la comparaison avec des notions de structuration proches telle que la *largeur de branche* (*branchwidth*) [32], l'extension des résultats aux matroïdes, qui peuvent eux aussi être décomposés de manière arborescente, l'algorithmique, séquentielle et parallèle associée, les raffinements de la notion de décomposition arborescente, dont on a vu quelques exemples dans la section 6.2.4. La notion de largeur arborescente a des formulations équivalentes en termes de *stratégies d'exploration des graphes* : voir [1] et [27].

Il existe une autre mesure de complexité des graphes, liée à un autre type de structuration (arborescente) et qui donne lieu à des algorithmes polyno-

miaux pour des problèmes exprimés en logique du second-ordre monadique. Il s'agit de la *largeur de clique* (*clique-width*) définie et étudiée dans [23], [22]. Cette notion est plus puissante que la largeur arborescente en ce sens que tout ensemble de graphes de largeur arborescente bornée est de largeur de clique bornée, alors que l'inverse n'est pas vrai (les cliques sont de largeur de clique 2 et de largeur arborescente non bornée). Les applications algorithmiques dépassent la seule vérification de propriétés et s'étendent aux problèmes de dénombrement, d'énumération, d'optimisation et de calcul de fiabilité dans les réseaux. Voir en particulier [7].

6.6. Exercices

1) **Largeur linéaire des arbres binaires**

On veut montrer que $pwd(T_n) = \lfloor n/2 \rfloor$ (notation définie dans la section 6.2.1).

a) Si G et H sont deux graphes de type $\{r\}$, on définit $G \boxplus H = ext(G)//ext(H)$ avec $ext(G) = \mathbf{ren}_{s \longleftrightarrow r}(\mathbf{fg}_r(G//\mathbf{rs}))$. Montrer que si G_1, G_2, G_3, G_4 sont des graphes de type $\{r\}$ tels que $pwd(\mathbf{fg}_r(G_i)) \leq k$ pour $i = 1, ..., 4$, alors :

$$pwd((G_1 \boxplus G_2) \boxplus (G_3 \boxplus G_4)) \leq k.$$

b) En déduire que $pwd(T_{2m+1}) \leq m$ pour tout m.

c) L'article [38] démontre le résultat suivant : Pour tout arbre $T = (N, A)$ on a : $pwd(T) \geq k + 1$ si et seulement si T a un noeud v tel $T[N - \{v\}]$ a au moins 3 composantes connexes de largeur linéaire au moins k. En déduire que $pwd(T_{2m}) = pwd(T_{2m+1}) = m$.

2) **Majorations de largeurs arborescentes**

a) En fonction de quels paramètres (degré, largeur arborescente, diamètre,...) relatifs à un graphe G peut-on majorer la largeur arborescente de son *graphe des incidences entre les arêtes* (*line graph*) ?

b) Soit H un graphe obtenu à partir d'un graphe G par subdivisions itérées d'arêtes. Peut-on majorer la largeur arborescente de H en fonction de celle de G ?

3) **Graphes de fonctions**

Le graphe $G(f)$ d'une fonction partielle $f : V \longrightarrow V$ où V est un ensemble fini a pour sommets les éléments de V et pour arcs les couples $x \longrightarrow f(x)$ pour tous $x \in V$. Montrer que ces graphes ont une largeur arborescente bornée indépendante de f. Préciser son maximum.

4) **Propriété de Helly**

La *Propriété de Helly* pour les arbres (graphes non orientés, connexes, sans cycles) énonce que si k sous-arbres d'un arbre sont 2 à 2 d'intersection

non vide, alors l'intersection de ces k sous-arbres est non vide. Montrer que si l'intersection de 2 sous-arbres est non vide, c'est un arbre. Montrer que si 3 sous-arbres sont 2 à 2 d'intersection non vide, alors leur intersection est un sous-arbre non vide. Montrer la propriété de Helly par récurrence sur k.

5) **Mineurs**

a) Montrer la transitivité de la notion de mineur (définition 6.2.3.1). [On montrera que si G est sous-graphe de $H \setminus F$, alors $G = H' \setminus F'$ où H' est un sous-graphe de H et F' un ensemble d'arêtes.]

b) Montrer que tout graphe planaire G est mineur d'une grille carrée $G_{n\times n}$ suffisamment grande. [On montrera que G est un mineur d'un graphe planaire H de degré maximum 3 et on utilisera un arbre recouvrant en profondeur de H.]

6) **Construction du graphe valeur d'un terme**

Le but est de définir une construction concrète du graphe $val(t)$ pour $t \in T(\mathcal{F}_C)$, C fini. Une occurrence d'un symbole dans t peut être définie comme un entier qui représente sa position, le terme t étant écrit comme un mot. A t on associe l'ensemble $E(t)$ des couples $(u, 0)$ où u est une occurrence de $\mathbf{a}^{bcl}$, de $\mathbf{ab}$ ou de $\overrightarrow{\mathbf{ab}}$. Ces couples seront les arêtes du graphe à construire. On définit $V(t)$ comme l'ensemble des couples $(u, 1)$ tels que u est une occurrence de $\mathbf{a}$ ou de $\mathbf{a}^{bcl}$, et des couples $(u, 1), (u, 2)$ tels que u est une occurrence de $\mathbf{ab}$ ou de $\overrightarrow{\mathbf{ab}}$. Ces couples seront les sommets du graphe. Ainsi, $(u, 1)$ et $(u, 2)$ seront la a-source et la b-source respectivement de l'arête définie par l'occurrence u de la constante $\mathbf{ab}$ (ou de l'arc dans le cas de la constante $\overrightarrow{\mathbf{ab}}$). À cause des opérateurs de composition parallèle qui identifient des sources de mêmes étiquettes, un même sommet du graphe $G(t)$ sera le plus souvent défini par plusieurs éléments de $V(t)$. On définira une relation d'équivalence $\approx$ sur $V(t)$ qui signifie que deux couples représentent le même sommet. Les sommets de G seront les classes d'équivalence de cette relation.

i) Dans le cas particulier du terme $t = \overrightarrow{\mathbf{cd}}//ren_{a\longleftrightarrow c}(\overrightarrow{\mathbf{ab}}//\mathbf{fg}_b(\overrightarrow{\mathbf{ab}}//\overrightarrow{\mathbf{bc}}))$, définir les ensembles $E(t), V(t)$ et l'équivalence $\approx$.

ii) Montrer que l'équivalence $(u, i) \approx (v, j)$ ne dépend que des constantes et des opérations unaires associées aux nœuds de la chaîne qui relie u à v dans l'arbre syntaxique du terme t.

iii) Définir précisément cette équivalence en termes de l'appartenance à des langages rationnels des mots formés des opérations associées aux nœuds sur les chemins qui relient u et v à leur plus grand ancêtre commun dans t.

[On commencera par traiter les questions ii et iii dans le cas des termes sans renommages de sources].

7) **Propriétés inductives**

Déterminer les propriétés inductives permettant de tester sur les arbres syntaxiques des graphes de largeur arborescente au plus 3 si ces graphes :

- sont connexes,

- sont sans circuit,
- ont un circuit Hamiltonien (qui passe par chaque sommet une fois et une seule ; un 8 n'est pas Hamiltonien).

Remerciements : Je remercie Selma Djelloul et Mustapha Kante pour leurs commentaires qui m'ont conduit à clarifier de nombreuses formulations.

6.7. Bibliographie

1. Livres et articles de synthèse développant les notions présentées dans ce chapitre

[1] H. Bodlaender, A Partial k-Arboretum of Graphs with Bounded Treewidth. *Theoret. Comput. Sci.* **209** (1998) 1-45

[2] H. Bodlaender, A Tourist Guide through Treewidth, *Acta Cybernetica* **11** (1993) 1-22

[3] H. Bodlaender, Discovering treewidth, SOFSEM 2005 : Theory and Practive of Computer Science : 31st *Conference on Current Trends in Theory and Practive of Computer Science*, Lecture Notes in Computer Science **3381** (2005) 1-16. Version revue en ligne sur :
http ://www.cs.uu.nl/people/hansb/

[4] A. Brandstädt, V. Lê, J. Spinrad, *Graph Classes : A Survey*, SIAM, Philadelphia, 1999.

[5] B. Courcelle, Basic notions of Universal Algebra for Language Theory and Graph Grammars, *Theoretical Computer Science* **163** (1996) 1-54.

[6] B. Courcelle, The expression of graph properties and graph transformations in monadic second-order logic, Chapter 5 du « Handbook of graph grammars and computing by graph transformations, Vol. 1 : Foundations », G. Rozenberg ed., World Scientific (New-Jersey, London), 1997, pp. 313-400.

[7] B. Courcelle, J.Makowsky, U. Rotics, On the Fixed Parameter Complexity of Graph Enumeration Problems Definable in Monadic Second Order Logic, *Discrete Applied Mathematics* **108** (2001) 23-52.

[8] R. Diestel, *Graph Theory*, Second Edition, Springer-Verlag, 2000. Disponible gratuitement en lecture seule sur :
http ://www.math.uni-hamburg.de/home/diestel/books/graph.theory/

[9] R. Downey et M. Fellows, *Parameterized Complexity*, Springer-Verlag, 1999

[10] G. Rozenberg, ed., *Handbook of graph grammars and computing by graph transformations*, Vol. 1 : Foundations, World Scientific (New-Jersey, London), 1997.

[11] S. Hedetniemi, References on partial k-trees, *Discrete Applied Maths* **54** (1994) 281-290.

[12] J.Kleinberg, E. Tardos, *Algorithm design*, Addison-Wesley 2004.

[13] B. Mohar, C.Thomassen, *Graphs on surfaces*, John Hopkins University Press, Baltimore, 2001.

[14] A. Recski : *Matroid theory and its applications in electric network theory and in statics*, Algorithms and Combinatorics Vol. **6**. Springer-Verlag, 1989

[15] J. Spinrad, *Efficient graph representations*, Fields Institute Monographs, American Mathematical Society, 2003

[16] Comon et al., *Tree Automata Techniques and Applications*, Livre en ligne, Consultable gratuitement,
http ://www.grappa.univ-lille3.fr/tata/

2. Articles de recherche spécialisés

[17] H. Bodlaender, A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.* **25**(1996) 1305-1317.

[18] H. Bodlaender, J. Gilbert, H. Hafsteinsson, T. Kloks, Approximating Treewidth, Pathwidth, Frontsize, and Shortest Elimination Tree. *J. Algorithms* 18(1995) 238-255.

[19] H. Bodlaender, T. Hagerup : Parallel Algorithms with Optimal Speedup for Bounded Treewidth. *SIAM J. Comput.* **27** (1998) 1725-1746

[20] H. Bodlaender, T. Kloks, Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs. *J. Algorithms* **21** (1996) 358-402.

[21] V. Bouchitté, D. Kratsch, H. Müller and I. Todinca, On treewidth approximations, Discrete Applied Maths **136** (2004) 183-196.

[22] B. Courcelle, J. Makowsky, U. Rotics. Linear time solvable optimization problems on certain structured graph families, *Theory of Computer Systems* **33** (2000) 125-150.

[23] B. Courcelle, S. Olariu, Upper bounds to the clique-width of graphs, *Discrete Applied Mathematics* **101** (2000) 77-114.

[24] B. Courcelle, S. Oum, Vertex-minors, monadic second-order logic and a conjecture by Seese, *J. Comb. Theory B* volume 97 (2007) 91-126.

[25] B. Courcelle, R. Vanicat, Query efficient implementations of graphs of bounded clique-width, *Discrete Applied Mathematics* **131** (2003) 129-150

[26] S. Djelloul, Tree decompositions and Cartesian products of graphs, ICGT 2005, Article soumis.

[27] P. Fraigniaud, N. Nisse, Connected treewidth and connected graph searching, Actes du "7th Latin American Symposium", Lec. Notes Comput. Sci. **3887** (2006) 479-490.

[28] N. Klarlund : Mona & Fido : The Logic-Automaton Connection in Practice. *Computer Science Logic 1997*, Lecture Notes in Computer Science, **1414** (1998) 311-326

[29] J. Lagergren, Upper bounds on the size of obstructions and intertwines, *Journal of Combinatorial Theory Series B*, **73** (1998) 7–40

[30] C. McDiarmid, B. Reed : Channel assignment on graphs of bounded treewidth, *Discrete Mathematics* **273** (2003) 183-192.

[31], S. Oum, P. Seymour, Approximating branchwidth and clique-width, *J. Comb. Th. B* volume 96 (2006) 514-528.

[32] N. Robertson, P. Seymour, Graph Minors, XIII. The Disjoint Paths Problem, *Journal of Combinatorial Theory Series B* **63** (1995) 65-110.

[33] N. Robertson, P. Seymour, Graph Minors, XX. Wagner's conjecture, *Journal of Combinatorial Theory, Series B*, **92**,(2004) 325-357

[34] N. Robertson, P. Seymour, R. Thomas, Quickly Excluding a Planar Graph, *Journal of Combinatorial Theory, Series B*, **62** (1994) 323-348

[35] D. Seese,The structure of the models of decidable monadic theories of graphs, *Annals of Pure and Applied Logic*, **53** (1991) 169-195

[36] P. Seymour, A bound on the excluded minors for a surface, 2005, soumis pour publication.
http ://www.math.princeton.edu/˜pds/

[37] M. Thorup : All Structured Programs have Small Tree-Width and Good Register Allocation. *Inf. Comput.* **142** (1998) 159-181

[38] A. Takahashi, S. Ueno and Y. Kajitani, Minimal acyclic forbidden minors for the family of graphs with bounded path-width, *Discrete Mathematics* **127** (1994) 293-304

[39] A. Yamaguchi, K. F. Aoki, H. Mamitsuka, Graph Complexity of Chemical Compounds in Biological Pathways, *Genome Informatics* **14** (2003)376-377