3.1 Les graphes de largeur de clique $\leqslant 3$

3.1.1 Définition

Il est classique de définir une nouvelle classe de graphes à partir d'une ancienne classe de graphes et d'*opérations de composition*, c'est-à-dire d'opérations qui, étant donnés un ou plusieurs graphes, nous fournissent un nouveau graphe : on forme naturellement la nouvelle classe en appliquant les opérations de toutes les manières possibles aux graphes de l'ancienne classe, puis aux résultats précédents, un nombre fini de fois. Ainsi, en prenant comme seule opération la substitution, on obtient à partir des cliques, des stables et des graphes premiers, l'ensemble de tous les graphes ; si on ne part que des cliques et des stables, on obtient les graphes sans P_4. Autre exemple, au premier chapitre nous avons vu que les graphes distance-héréditaires s'obtiennent à partir des stables avec les opérations « ajouter un sommet pendant » ou « ajouter un jumeau à un sommet ». À chaque fois l'appartenance d'un graphe à la nouvelle classe est liée à l'existence d'une décomposition de ce graphe.

Les graphes de largeur de clique $\leqslant 3$ vont également se définir à partir d'opérations de composition (c'est même la seule caractérisation que l'on en connaisse), sauf qu'il ne s'agira pas de compositions de graphes *stricto sensu* : il va nous falloir généraliser les graphes aux *graphes étiquetés*.

Définition 3.1. Dans ce chapitre le terme « **graphe** » désignera les graphes dont les sommets sont étiquetés par 1, 2 ou 3.

Définition 3.2. Soit G un graphe. On note $V_1(G) = \{v \in V(G) \mid v \text{ est étiqueté } 1\}$; et similairement pour $V_2(G)$ et $V_3(G)$.

Définition 3.3. Soit G un graphe. On note $E_{ij}(G) = \{e \in E(G) \mid e = a–b, a \in V_i, b \in V_j\}$ l'ensemble des **arêtes de type** $i–j$, pour $i \neq j \in \{1, 2, 3\}$. On dit que E_{ij} **est complet** si toutes les arêtes de type $i–j$ existent.

Le graphe étiqueté est en effet l'objet le plus naturel à considérer pour l'étude des graphes de largeur de clique $\leqslant 3$. Cela ne signifie pas que cette notion soit sans intérêt dans le cadre des graphes non étiquetés, bien au contraire ; c'est simplement que, pour construire un graphe non étiqueté à l'aide des opérations de composition, il faudra utiliser des graphes étiquetés comme étapes intermédiaires. Il faut donc voir la généralisation aux graphes étiquetés comme un effet secondaire des techniques employées, plutôt que comme un intérêt majeur des théories développées ; l'algorithme que nous proposons résout en fait le problème dans le cas général des graphes étiquetés, qui s'avère ne pas être plus difficile.

En fait on ne peut pas vraiment construire de graphes non étiquetés, il faut utiliser une convention.

Définition 3.4. Dans ce chapitre les graphes non étiquetés seront considérés comme des graphes dont tous les sommets sont étiquetés par 1. [1]

[1] Il peut paraître inélégant de privilégier artificiellement l'étiquette 1, alors que toutes les étiquettes jouent *a priori* le même rôle. C'est juste une facilité de présentation : un puriste peut imaginer que l'on a

C'est le sens technique qui se cache derrière la question « Est-ce que ce graphe (non étiqueté) est de largeur de clique ⩽3 ? », pour laquelle nous cherchons un algorithme.

Ces précisions apportées, voici les opérations de composition :
- $\oplus$: opération qui réalise l'union disjointe de plusieurs graphes[2]. C'est cette opération qui permet d'avoir des graphes de plus en plus gros ;
- $\eta_{i,j\ i\neq j\ \in\{1,2,3\}}$: opération unaire qui relie les sommets étiquetés par i à ceux étiquetés par j (rend E_{ij} complet). C'est cette opération qui permet d'avoir des arêtes dans les graphes. Cette opération ne crée pas d'arêtes multiples : elle est sans effet sur les couples de sommets déjà voisins. Notez qu'elle n'est autorisée que pour $i \neq j$: elle ne permet pas de transformer en clique un ensemble de sommets de même étiquette ;
- $\rho_{i\to j\ i,j\in\{1,2,3\}}$: opération unaire qui change les étiquettes i en étiquettes j. C'est cette opération qui rend intéressante mais difficile l'étude des graphes de largeur de clique ⩽3.

D'autre part la classe de base à partir de laquelle nous construisons les graphes de largeur de clique ⩽3 est tout simplement la classe des graphes à un sommet. On peut ainsi représenter un graphe de largeur de clique ⩽3 par un arbre de décomposition, dont les nœuds au moins binaires sont étiquetés par $\oplus$, les nœuds unaires par un $\rho_{i\to j}$ ou un $\eta_{i,j}$, et les feuilles par un graphe à un sommet. À chaque nœud interne est associé un graphe, défini par le sous-arbre qu'il engendre, comme l'illustre FIG. 3.1. Comme les feuilles sont très simples à décrire on peut aussi utiliser une représentation équivalente à l'arbre de décomposition : une expression mathématique.

Définition 3.5. Les **graphes de largeur de clique** ⩽3 sont les graphes pouvant être obtenus à partir de graphes à un sommet et des opérations de composition. On appelle **3-arbre**[3] un arbre de décomposition d'un graphe de largeur de clique ⩽3, et **3-expression** l'expression associée.

3.1.2 Quelques propriétés simples

Avant d'en venir à l'origine du problème de la reconnaissance des graphes de largeur de clique ⩽3, familiarisons-nous avec eux en en étudiant quelques propriétés simples.

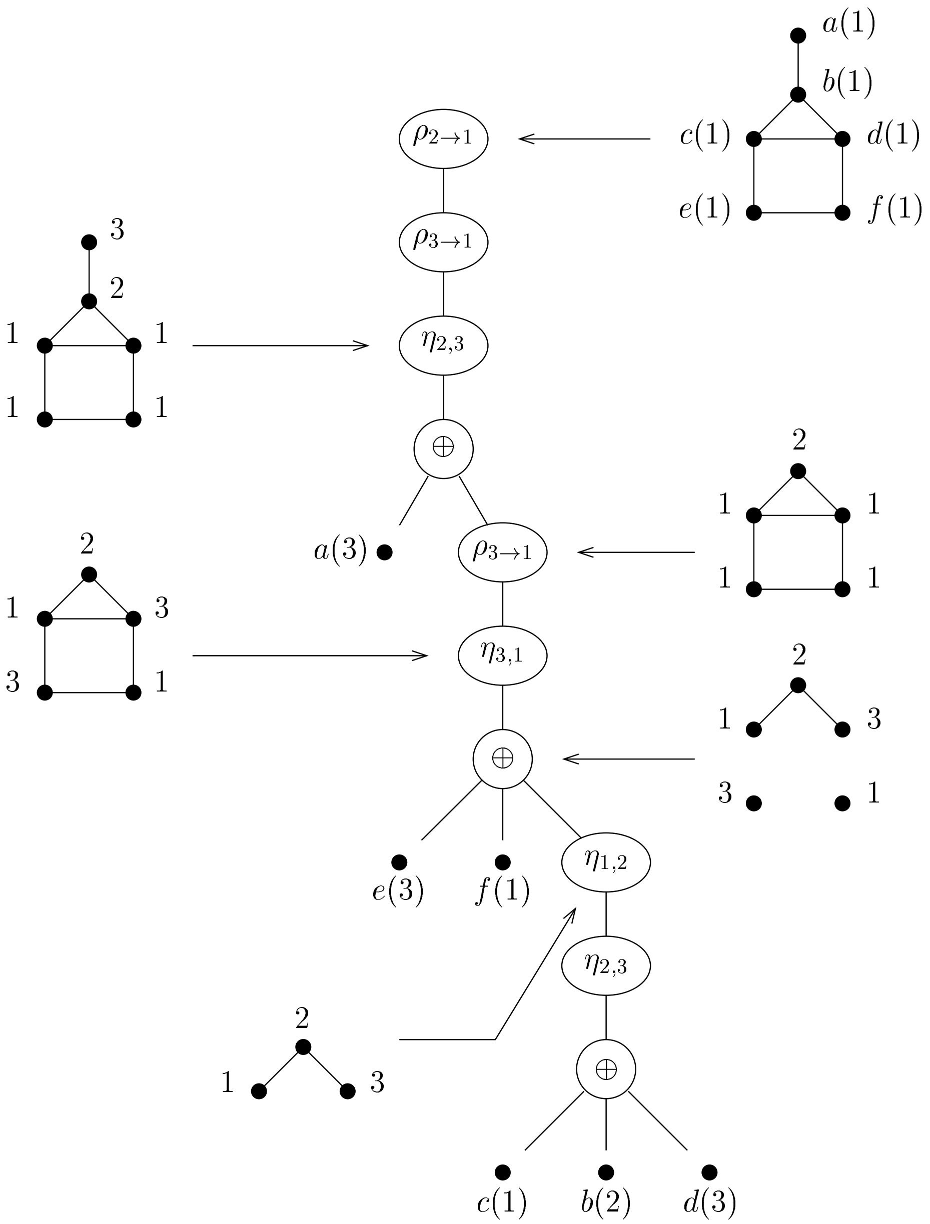
---

préalablement quotienté l'ensemble des graphes étiquetés par les permutations des étiquettes.

D'ailleurs, lorsqu'on dira qu'un graphe G « contient un graphe H », il faudra comprendre « à une éventuelle permutation des étiquettes près ».

[2]Nous n'entrons pas dans les détails formels qui seraient nécessaires à une définition propre de cette opération : il faudrait vérifier que les graphes à réunir sont bien disjoints, ou bien définir cette opération sur des classes d'isomorphisme. En effet, dans le cadre de ce travail, négliger ces subtilités ne présente aucun danger.

[3]Cette définition pose un problème de conflit terminologique : on verra plus loin que les k-arbres au sens classique sont une classe de graphes particulière. Comme ces deux notions n'apparaissent pas dans le même contexte, il ne me semble pas indispensable de résoudre ce conflit dans le cadre de ce travail (les k-arbres classiques ne sont mentionnés qu'au cours d'une excursion dans la largeur arborescente, section 3.2.4) ; dans un cadre plus général, il faudrait sans doute adopter une terminologie plus lourde, et parler de « cw-3-arbre ».



$$\rho_{2\to1}\circ\rho_{3\to1}\circ\eta_{2,3}\Big(a(3)\oplus\ \ \rho_{3\to1}\circ\eta_{3,1}\Big(e(3)\oplus f(1)\oplus\ \ \eta_{1,2}\circ\eta_{2,3}\big(c(1)\oplus b(2)\oplus d(3)\big)\Big)\Big)$$

FIG. 3.1 – Une décomposition d'un graphe de largeur de clique ⩽3

Tout d'abord on observe que la décomposition d'un graphe n'est pas unique : par exemple les opérations $\eta_{1,2}$ et $\eta_{1,3}$ sont commutatives. Pire, après un $\eta_{1,2}$ par exemple, on peut toujours refaire un autre $\eta_{1,2}$; ou bien, après un $\rho_{2\to 1}$, on peut toujours faire un $\rho_{1\to 2}$ suivi d'un $\rho_{2\to 1}$... En fait il est plutôt gênant de prendre en compte de telles opérations inutiles, parce que le 3-arbre peut avoit une taille quelconque à cause d'elles. Nous allons donc restreindre la définition d'un 3-arbre :

Définition 3.6. Un **3-arbre** (au sens strict) est un 3-arbre sans opérations inutiles[4].

Maintenant, après trois $\eta_{i,j}$ et deux $\rho_{i\to j}$, il n'est plus possible d'effectuer une nouvelle opération unaire. D'autre part, comme un 3-arbre d'un graphe à n sommets comporte n feuilles, il comporte au plus $n-1$ nœuds $\oplus$. On en conclut qu'il comporte $O(n)$ nœuds et feuilles : il s'agit d'un codage efficace du graphe.

La décomposition n'est toujours pas unique, loin de là, et nous reviendrons sur cette question.

Autre propriété, les graphes de largeur de clique ⩽3 forment une classe héréditaire. En effet, supprimer un sommet d'un graphe correspond à supprimer une feuille d'un 3-arbre de ce graphe, puis à corriger le reste de l'arbre ; cela montre qu'un sous-graphe d'un graphe de largeur de clique ⩽3 est aussi de largeur de clique ⩽3.

Étudions enfin une propriété qui va beaucoup nous simplifier la tâche :

Lemme 3.7. *Un graphe est de largeur de clique ⩽3 si et seulement si tous ses modules premiers, ainsi que son graphe caractéristique, sont de largeur de clique ⩽3.*

Il faut toutefois prendre garde que l'on parle de graphes étiquetés :

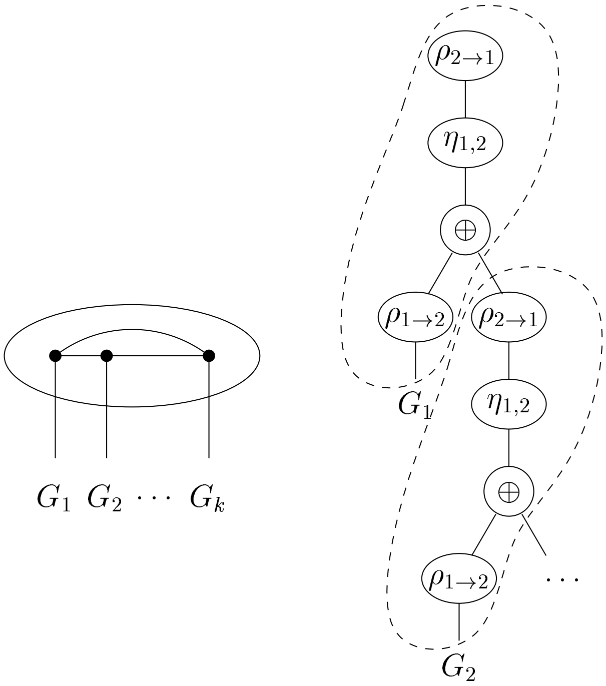
Définition 3.8. Soit G un graphe (étiqueté). Un sous-ensemble de $V(G)$ est dit **unicolore** si tous ses éléments portent la même étiquette ; **bicolore** s'ils utilisent exactement deux étiquettes distinctes ; et **tricolore** sinon.

Définition 3.9. Un **module d'un graphe étiqueté** est un sous-ensemble de sommets *unicolore* qui soit module du graphe en oubliant les étiquettes.

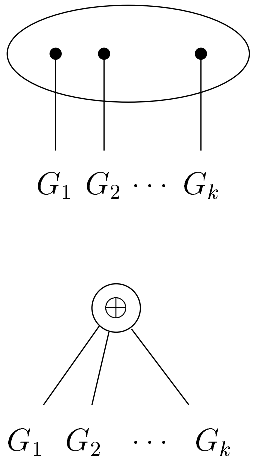
On ne s'intéresse qu'aux sommets qui vont avoir le même comportement : c'est pourquoi les modules doivent être unicolores. Il y a donc moins de modules, et plus de graphes premiers, que si l'on oubliait les étiquettes. Attention, cette définition pose le problème qu'un graphe non connexe (par exemple) peut être premier ; mais il a semblé plus simple de procéder ainsi.

Pour trouver la décomposition modulaire d'un graphe étiqueté non unicolore, il s'agit surtout de trouver les modules maximaux (qui ne peuvent être le graphe entier) : d'une part ces modules ont une décomposition « normale », et d'autre part, une fois qu'ils sont contractés, il reste bien un graphe premier. Il est donc facile de trouver cette décomposition à partir de la décomposition du graphe en oubliant ses étiquettes : il suffit de parcourir

[4] Si l'on veut une définition formelle, on peut définir une suite d'opérations unaires comme étant sans opérations inutiles si chaque $\eta_{i,j}$ ajoute effectivement des arêtes (il y en a trois au plus) ; et si aucun $\rho_{i\to j}$ n'est suivi d'un $\rho_{j\to k}$. Il n'est pas difficile de voir que l'on peut toujours se ramener à un 3-arbre sans opérations inutiles (par exemple, on peut systématiquement commencer par les $\eta_{i,j}$).



(a) Série



(b) Parallèle

(En supposant que les graphes G_1 à G_k soient unicolores d'étiquette 1.)

FIG. 3.2 – Nœuds série et parallèle d'une décomposition modulaire

cette dernière en partant des feuilles, pour repérer les sous-arbres maximaux unicolores. On a alors la décomposition des modules maximaux, et il ne reste plus qu'à les contracter pour construire la racine : on peut donc calculer toute la décomposition en temps linéaire.

La signification du lemme étant établie, nous pouvons donner sa preuve.

Preuve : ($\Rightarrow$) Si G est un graphe de largeur de clique ⩽3, alors ses modules comme son graphe caractéristique sont des sous-graphes, donc de largeur de clique ⩽3 également. ($\Leftarrow$) Si H est un graphe de largeur de clique ⩽3, s un sommet de H, et H' un graphe unicolore de largeur de clique ⩽3, de même étiquette que s, alors le graphe G obtenu en substituant H' à s est encore de largeur de clique ⩽3. En effet, il est facile de construire un 3-arbre de G : il suffit de remplacer la feuille correspondant à s dans un 3-arbre de H par un 3-arbre de H' ; les sommets de H' vont tous se comporter comme se comportait s au cours de la construction puisqu'ils ont la même étiquette, et donc H' se substitue bien à s.

On peut ainsi reconstituer un 3-arbre d'un graphe G à partir des 3-arbres de ses modules et de son graphe caractéristique. Pour achever la preuve du lemme, il suffit de trouver les 3-arbres des stables et des cliques, ce qui est réalisé FIG. 3.2.

Ce lemme fait mieux que nous ramener au cas des graphes premiers ; ou plutôt les graphes premiers ont une propriété qui va nous être très utile.

Lemme 3.10. *Dans un 3-arbre d'un graphe premier, si un graphe associé à un nœud est unicolore, alors il n'a qu'un sommet (c'est une feuille).*

En effet dans le cas contraire ce graphe correspondrait à un module du graphe total, puisque les sommets en question continueraient de partager la même étiquette au cours de la construction, et donc gagneraient les mêmes voisins.

3.2 Notion de largeur de clique ; intérêt du problème

3.2.1 Largeur de clique k

Nous venons de présenter la classe des graphes de largeur de clique ⩽3. Mais, bien sûr, il ne s'agit que d'un cas particulier : en général, pour une constante k donnée, on autorise l'utilisation de k étiquettes distinctes, et l'on obtient les notions de k-arbre et de graphes de largeur de clique $\leqslant k$ (le terme de « largeur de clique » fait donc référence à un recouvrement des arêtes du graphe par des bipartis complets, ou « bi-cliques »). De même que pour la largeur de clique ⩽3, ces graphes peuvent être étiquetés avec au plus k étiquettes distinctes (on dit que ce sont des k-graphes), mais l'on s'intéresse généralement aux graphes non étiquetés, vus comme des graphes étiquetés avec 1. L'ensemble des k-graphes de largeur de clique $\leqslant k$ est noté $\mathrm{cw}(k)$ (en référence au terme anglais « clique-width »).

Depuis son introduction en 1993 par Courcelle, Engelfriet et Rozenberg [Courcelle et al., 1993], cette notion a suscité un intérêt croissant dans le domaine de l'algorithmique ; la recherche est actuellement très active, et il semble bien que les théories en cours de développement n'en soient qu'à leur adolescence. La propriété phare des graphes de largeur de clique $\leqslant k$ est qu'une vaste classe de problèmes, éventuellement NP-complets dans le cas général, y sont polynomiaux. Pour préciser cette affirmation nous devrons faire une petite excursion dans le domaine de la logique ; mais auparavant, nous mentionnons quelques résultats connus sur ces graphes.

Les graphes de largeur de clique 1 sont très simples, puisqu'il s'agit des graphes sans arêtes. Les graphes de largeur de clique ⩽2 ne sont pas très difficiles à caractériser non plus : il s'agit des cographes[5]. On a donc une caractérisation par sous-graphe interdit de ces deux classes (le P_2 et le P_4 respectivement), et l'on sait les reconnaître en temps polynomial, et même linéaire.

Par contre on ne sait rien du cas général : on ne sait pas s'il est possible de reconnaître les graphes de largeur de clique $\leqslant k$ en temps polynomial (pour k fixé) ; on ne connaît aucune caractérisation de ces classes en dehors de leur définition par k-arbre.

Définition 3.11. La **largeur de clique** d'un graphe est le plus petit entier k tel que le graphe soit de largeur de clique $\leqslant k$.

On ne sait même pas si le problème général du calcul de la largeur de clique exacte est NP-complet ou non.

Il n'est donc pas évident de borner supérieurement la largeur de clique d'une classe de graphes. Cela a pourtant été réalisé dans certains cas, comme certaines classes de graphes présentant peu de P_4 : pour $q \geqslant 7$, les graphes tels que tout sous-graphe à q sommets comporte au plus $q-3$ P_4 sont de largeur de clique $\leqslant(q-1)$ [Makowsky et Rotics, 1999]. On sait aussi que les graphes distance-héréditaires sont de largeur de clique ⩽3 [Golumbic et Rotics, 2001]. Sur ces exemples, on peut même obtenir une décomposition en temps linéaire.

[5]On le voit en regardant la dernière opération d'un 2-arbre : un $\eta_{1,2}$ correspond à la mise en série de deux graphes (l'un étiqueté 1, l'autre 2), un $\oplus$ à une opération parallèle.

Au niveau d'un graphe individuel, on sait que la notion de largeur de clique est à peu près stable par passage au complémentaire : $\mathrm{cw}(\overline{G}) \leqslant 2\,\mathrm{cw}(G)$ [Courcelle et Olariu, 2000]. [Johansson, 1998] observe également que[6]

$$\mathrm{cw}(G) \leqslant n(G) - \lfloor \log_2 n(G) \rfloor + 1;$$

et surtout, il montre que

$$(\forall c, p \in]0,1[) \quad \Pr\big\{\mathrm{cw}(G(n,p)) < c.n\big\} \underset{n\to+\infty}{\longrightarrow} 0,$$

c'est-à-dire qu'il n'y a pas de borne supérieure intéressante linéaire en $n(G)$ à la largeur de clique d'un graphe, et qu'un « graphe aléatoire » a une largeur de clique très proche de n.

Qu'est-ce qu'un graphe de grosse largeur de clique ? Un problème actuel de la théorie est le manque de méthodes pour borner inférieurement la largeur de clique d'un graphe donné ; mentionnons tout de même un résultat de [Golumbic et Rotics, 2001] : la grille carrée $n \times n$ a une largeur de clique d'exactement $n + 1$ (voir aussi la page 158). Ce papier montre aussi que certaines classes ne sont pas de largeur de clique bornée, comme les graphes de permutation ou les graphes d'intervalles unitaires. Les « grosses » classes de graphes ne peuvent pas non plus être de largeur de clique bornée, puisqu'il y a $2^{O(n)}$ graphes de largeur de clique $\leqslant k$ à n sommets.

Maintenant que l'on a expliqué que l'on ne sait pas grand chose de ces graphes, expliquons en quoi cela nous manque.

3.2.2 Logique monadique du second ordre

Logique monadique MS

Les problème dont nous avons annoncé qu'ils seraient solubles sur la classe des graphes de largeur de clique $\leqslant k$ devront s'exprimer dans une certaine logique, que nous allons maintenant introduire. Les définitions à venir sont tirées de [Courcelle et al., 2000] ; en fait nous ne donnons pas ici les définitions les plus générales, estimant que les résultats plus simples que l'on va donner seront suffisants pour notre propos : le lecteur désireux d'en savoir plus est prié de se reporter à cette référence.

Rappelons que la logique du second ordre autorise l'utilisation de variables, éventuellement quantifiées, qui représentent des relations d'arité donnée. La logique **monadique** du second ordre restreint l'utilisation de telles variables aux relations d'arité un ; autrement dit elle autorise l'utilisation de variables d'ensemble, qui représentent des sous-ensembles du domaine d'une structure logique : si X est une telle variable, $X(u)$ correspond à $u \in X$.

Pour modéliser les graphes en tant que structures logiques, nous devons définir un vocabulaire, c'est-à-dire que nous devons introduire un certain nombre de constantes qu'utiliserons les formules. En fait il nous en suffit d'une : on note τ_1 le vocabulaire comprenant simplement la relation E (binaire), qui correspondra à l'adjacence des sommets d'un

[6]En fait il montre que $\mathrm{cw}(G) \leqslant n - k$, pour tout k tel que $2^k + k \leqslant n - k$: si je ne m'abuse on en déduit la borne que je donne.

graphe. Pour G un graphe, on note $G(\tau_1)$ la structure logique $\langle V(G), E(G)\rangle$: c'est bien une structure de type τ_1, sur le domaine $V(G)$; et $E(x,y)$ correspond bien à l'affirmation que l'arête $x{-}y$ est présente.

Problèmes de décision

Cela nous permet de définir les **problèmes de décision** $\mathbf{MS}(\tau_1)$ **sur** $\mathscr{G}$: ce sont les problèmes qui peuvent s'exprimer sous la forme « Étant donné $\mathsf{G} \in \mathscr{G}$, est-ce que $\mathsf{G} \models \phi$? », où ϕ est une formule close de la logique monadique du second ordre sur le vocabulaire τ_1, et $\mathscr{G}$ est une classe de graphes, c'est-à-dire un ensemble de structures de type τ_1 qui correspondent à des graphes. Bien sûr ϕ et $\mathscr{G}$ ne font pas partie des données du problème (ce sont son expression et son domaine) : la seule donnée est G. Si $\mathscr{G}$ correspond à l'ensemble de tous les graphes, on parle simplement de problème de décision $\mathrm{MS}(\tau_1)$.

Par exemple, la 3-colorabilité est un problème $\mathrm{MS}(\tau_1)$: en effet il peut être exprimé à l'aide de la formule

$$\phi \equiv\ (\exists X_1, X_2, X_3)$$
$$\text{Partition}(X_1,X_2,X_3) \text{ et Indépendant}(X_1) \text{ et Indépendant}(X_2) \text{ et Indépendant}(X_3),$$

où $\text{Partition}(X_1,X_2,X_3) \equiv$
$$(\forall v)(v \in X_1 \text{ ou } v \in X_2 \text{ ou } v \in X_3)$$
$$\text{et } \neg(\exists v)\ ((v \in X_1 \text{ et } v \in X_2) \text{ ou } (v \in X_1 \text{ et } v \in X_3) \text{ ou } (v \in X_2 \text{ et } v \in X_3)),$$

et $\text{Indépendant}(X) \equiv (\forall u,v)\ (u \in X \text{ et } v \in X \Rightarrow \neg E(u,v))$.

On voit bien que la même construction peut être appliquée à la k-colorabilité, qui (pour k fixé !) est donc également un problème $\mathrm{MS}(\tau_1)$.

Problèmes d'optimisation

En ajoutant des variables libres à ϕ, et des fonctions d'évaluation linéaires, on peut étendre la définition aux problèmes d'optimisation. Un **problème d'optimisation Lin-**$\mathbf{MS}(\tau_1)$ **sur** $\mathscr{G}$ est un problème qui peut s'exprimer sous la forme

Étant donnés $\mathsf{G} \in \mathscr{G}$, $a_i \in \mathbb{N}$, $i = 1$ à l, et $f : V(\mathsf{G}) \mapsto \mathbb{N}$, trouver des ensembles X_1, …, X_l tels que

$$\langle \mathsf{G}, X_1, \dots, X_l\rangle \models \phi(X_1,\dots,X_l)$$

et
$$\sum_{1\leqslant i\leqslant l} a_i\, f(X_i) \;=\; \mathrm{Max}\{\sum_{1\leqslant i\leqslant l} a_i\, f(X_i') \;\mid\; \langle \mathsf{G}, X_1', \dots, X_l'\rangle \models \phi(X_1',\dots,X_l')\},$$

où $f(X_i)$ est une notation courte pour $\sum_{x\in X_i} f(x)$ (l'évaluation de X_i), et ϕ est une formule de $\mathrm{MS}(\tau_1)$ comportant l variables libres. On voit que ϕ caractérise les solutions « valables », parmi lesquelles on en cherche une maximum selon une fonction d'évalution linéaire f et des coefficients a_i (qui font partie des données du problème). Résoudre ce problème correspond soit à trouver une solution qui maximise les solutions valables, soit à répondre qu'il n'existe aucune solution valable ; un problème d'optimisation est donc une généralisation d'un problème de décision. Si $\mathscr{G}$ correspond à l'ensemble de tous les graphes, on parle simplement de problème d'optimisation Lin$\mathrm{MS}(\tau_1)$.

Par exemple, le problème de la clique pondérée maximum est un problème Lin$\mathrm{MS}(\tau_1)$. En effet il est facile d'écrire une formule ϕ qui exprime que sa variable libre est une clique :

$$\text{Clique}(X) \;\equiv\; (\forall u, v)\ (u \in X \text{ et } v \in X \text{ et } u \neq v \Rightarrow E(u,v)),$$

et l'on cherche simplement à maximiser $f(X)$, pour un f donné. En particulier le problème de la clique (« Y a-t-il dans ce graphe une clique de taille au moins k ? ») est dans Lin$\mathrm{MS}(\tau_1)$; de même, par symétrie, que le problème de l'indépendant pondéré maximum.

Remarque : *La définition de Lin*$\mathrm{MS}(\tau_1)$ *donnée ici ne fait intervenir qu'une seule fonction d'évaluation ; c'est une simplication de la vraie définition, qui autorise à passer* m *fonctions en paramètre pour optimiser* $\sum_{i,j} a_{ij}\, f_j(X_i)$.

Appartenance d'un problème à Lin$\mathrm{MS}(\tau_1)$

On voit facilement que de nombreux autres problèmes (prenons un autre exemple : celui de l'ensemble dominant « Y a-t-il dans ce graphe un ensemble de taille au plus k qui voit le reste du graphe ? ») sont dans Lin$\mathrm{MS}(\tau_1)$. D'autres semblent ne pas y appartenir à première vue, alors qu'un peu de réflexion permet d'en trouver une expression dans Lin$\mathrm{MS}(\tau_1)$.

Supposons par exemple qu'un problème fasse intervenir des chemins induits. Comme nous ne pouvons pas utiliser d'ensemble d'arêtes, on pourrait croire qu'un tel problème ne soit pas exprimable ; mais en fait les chemins induits ne sont pas un obstacle à l'expressibilité d'un problème. En effet, d'une part, on peut exprimer qu'un sommet a un ou deux voisins dans un ensemble ; puis qu'un ensemble (vu comme un sous-graphe) n'a que des sommets de degré (interieur) deux, sauf deux de degré un : cela caractérise les unions d'un chemin induit avec un nombre quelconque de cycles. D'autre part, on peut exprimer qu'un ensemble est strictement inclus dans un autre, et qu'un ensemble a des voisins dans un autre ensemble : cela permet d'exprimer qu'un ensemble ne contient pas plusieurs composantes connexes. On en déduit qu'il existe dans $\mathrm{MS}(\tau_1)$ une formule Chemin-Induit(X).

L'appartenance d'un problème à Lin$\mathrm{MS}(\tau_1)$ n'est donc pas toujours évidente ; et la non-appartenance non plus. Ainsi le problème de savoir si un graphe contient un circuit hamiltonien n'est pas dans Lin$\mathrm{MS}(\tau_1)$ [Courcelle, 1997], alors que ce qui précède pourrait laisser espérer qu'il en existe une expression astucieuse. De même, le problème de la coloration optimale n'est pas dans Lin$\mathrm{MS}(\tau_1)$ [Lautemann, 1993] : c'est intuitif (il semble difficile de se passer d'une variable représentant une partition des sommets), mais nécessite néanmoins une preuve.

Cependant, le point important est qu'une large classe de problèmes peuvent facilement s'exprimer dans Lin$\mathrm{MS}(\tau_1)$; et que l'expressibilité est une classification des problèmes perpendiculaire à la classification habituelle selon leur complexité algorithmique, comme le montrent les exemples NP-complets que j'ai donnés.

3.2.3 Le théorème d'algorithmique

L'intérêt algorithmique de la largeur de clique tient au théorème suivant.

Théorème 3.12. *[Courcelle et al., 2000] Pour tout entier k fixé, pour tout problème de Lin*MS*(τ_1), il existe un algorithmique polynomial qui résout ce problème sur la classe des graphes de largeur de clique $\leqslant k$, à condition que le graphe soit fourni sous la forme d'une k-expression.*

*Cet algorithme peut être construit à partir de l'expression du problème dans Lin*MS*(τ_1), et il a même une complexité linéaire (en $c(k, \phi)(n + m)$).*

Ce théorème fait saliver à cause des nombreux problèmes NP-complets que contient Lin-MS(τ_1). En voyant ce théorème, quelques souvenirs confus de cours de logique remontant à la surface, je me suis dit qu'il devait être horriblement compliqué à démontrer, et qu'il valait mieux l'utiliser comme une boîte noire. En fait je ne dois pas être le seul ; en tout cas un certain nombre de malentendus ont l'air de circuler sur l'intérêt exact de ce théorème et les applications pratiques que l'on peut en espérer.

Pourtant la preuve de ce théorème [Rotics, 1998] n'est pas inabordable[7] : le point clé est qu'il y a un nombre fini de formules de hauteur de quantification donnée (à équivalence tautologique près). L'idée est la suivante : d'une certaine manière, un problème à résoudre sur un k-graphe passe au quotient selon les opérations de compositions, c'est-à-dire qu'il se transforme en un ou plusieurs problèmes à résoudre sur de nouveaux k-graphes lorsque l'on descend le long de l'arbre de décomposition ; et ces transformations préservent la hauteur de quantification. Il est possible de précalculer des fonctions sur l'ensemble fini de formules en question (c'est la construction de l'algorithme), de manière à permettre ces transformations en temps constant ; on se ramène ainsi à résoudre un grand nombre de problèmes sur les feuilles de la décomposition, qui sont des graphes à un sommet. Au bout du compte l'algorithme linéaire est un parcours d'arbre assez simple.

Tel que l'algorithme est présenté dans [Rotics, 1998], il n'est pas très difficile d'évaluer la constante $c(k, \phi)$ (qui en fait est un $c(k, h)$, où h est la hauteur de quantification de ϕ) en fonction du nombre fini de formules en question : et c'est essentiellement la même chose. Ce dernier nombre n'est à son tour pas difficile à majorer : on trouve une tour d'exponentielles de hauteur h, avec un k en haut. Je ne sais pas s'il est possible d'en trouver une bonne minoration, mais il me semble qu'on ne devrait essentiellement rien gagner, c'est-à-dire que ce nombre est effectivement une tour d'exponentielles. Il est donc illusoire de laisser espérer que ce théorème puisse s'utiliser en pratique, en l'état actuel des choses : même le problème de la clique maximum (hauteur de quantification 2) sur les graphes de largeur de clique $\leqslant 3$ est probablement hors de portée (je ne parle même pas du coût de construction de l'algorithme).

Je dis « en l'état actuel des choses », parce qu'en fait tout est fait à la louche dans cet algorithme : par exemple travailler à équivalence tautologique près est une approximation

[7] En fait je pense même qu'avec quelques efforts pédagogiques ce théorème pourrait être perçu comme simple et intuitif. De manière générale il me semble que ce domaine de recherche souffre d'une absence d'ouvrage de synthèse qui faciliterait l'accès au nouveau venu.

grossière, de nombreuses formules devenant indistinguables sur les graphes finis, et encore plus sur les graphes de largeur de clique bornée ; et de manière générale, on peut voir l'algorithme comme un automate fini, dont on pourrait minimiser le nombre d'états pour un problème donné (voir la discussion page 159). Il est donc trop tôt pour dire si cette théorie deviendra effective ou non.

Quoi qu'il en soit, ce théorème est pour l'instant à prendre au sens existentiel ; et c'est un fait bien connu que le chercheur qui sait qu'il y a une solution à son problème la trouve plus facilement, c'est là tout l'intérêt du théorème.

Ou plus précisément c'est là tout l'intérêt du théorème quand on peut l'appliquer... Le problème vient de la nécessité de fournir une k-expression : *a priori* on ne peut pas affirmer qu'un problème $\mathrm{Lin}\textsc{ms}(\tau_1)$ est polynomial sur une classe de graphes de largeur de clique bornée sans prouver qu'il existe un algorithme polynomial calculant une k-expression d'un graphe de cette classe. Il faut donc réunir deux conditions pour profiter de la puissance du théorème : se placer sur une classe de graphes de largeur de clique bornée d'une part ; et exhiber un algorithme de calcul d'une k-expression d'autre part.

Ce travail a été fait dans [Courcelle et al., 2000] sur des classes de graphes présentant peu de P_4 : et quelques preuves ont suffi pour unifier et généraliser un certain nombre de travaux précédents ; il est donc hors de doute que le théorème d'algorithmique peut rendre de grands services. Et la question de savoir s'il pourrait en rendre de plus grands encore se pose avec d'autant plus d'acuité : un algorithme de reconnaissance des graphes de largeur de clique $\leqslant k$, fournissant automatiquement le deuxième prérequis, simplifierait beaucoup l'utilisation du théorème, sur des classes de graphes plus grosses.

L'état actuel des connaissances sur la largeur de clique évoque celui qu'ont connu vers le début de leur histoire de nombreux problèmes de détermination d'un paramètre de complexité, comme la coloration : probablement NP-complet en général, trivialement polynomial à k fixé pour de petites valeurs de k, on ne sait pas quelle est la complexité du problème pour des valeurs de k supérieures à un certain seuil k_0. Puis la recherche avance, et en général deux cas de figure se présentent : soit le problème s'avère NP-complet pour $k = k_0$, et il n'est pas très difficile de généraliser la preuve à $k \geqslant k_0$; soit l'on parvient à trouver des techniques polynomiales pour reconnaître les objets de type k_0, et en fait l'essence de la complexité du problème a été révélée : ces techniques se généralisent à $k \geqslant k_0$.

L'intérêt du problème de la reconnaissance des graphes de largeur de clique $\leqslant 3$ trouve son origine dans ces analogies : on pensait que $k_0 = 3$, et que la solution à cette question donnerait la solution au problème général. Malheureusement, comme on va le voir, la solution polynomiale que l'on propose ne se généralise pas : ce travail révèle surtout qu'il semble y avoir un cas intermédiaire entre les cas triviaux et le cas général, et que finalement k_0 serait plutôt égal à 4 ; ou du moins nous n'avons révélé aucune essence de complexité flagrante.

3.2.4 Liens avec la largeur arborescente

Pour capturer des problèmes comme celui du cycle hamiltonien, il faut pouvoir manipuler des ensembles d'arêtes, c'est-à-dire ajouter l'ensemble des arêtes au domaine des structures logiques considérées ; il faut alors ajouter des constantes au vocabulaire pour pouvoir différencier les sommets des arêtes. On introduit donc le vocabulaire τ_2, qui comporte trois relations : une relation binaire R (pour les adjacences sommet-arête), et deux relations unaires E et V (vérifiées respectivement sur l'ensemble des arêtes et l'ensemble des sommets : en fait on a toujours $E(x)$ ou $V(x)$). Comme précédemment, à tout graphe G on peut associer une structure $G(\tau_2)$: mais cette fois son domaine est $V(G) \cup E(G)$.

On peut alors parler des problèmes Lin$\textsc{ms}(\tau_2)$ sur une classe de graphes ; cet ensemble de problèmes contient bien sûr Lin$\textsc{ms}(\tau_1)$, et il est même strictement plus gros comme le montre l'exemple du problème du cycle hamiltonien. Il ne contient toujours pas le problème classique de la coloration optimale [Lautemann, 1993], mais on aimerait bien étendre les résultats précédents à cet ensemble plus vaste. Malheureusement il semble que ce ne soit pas possible (rappelons que P_1 et NP_1 désignent les classes des langages sur une lettre de P et de NP) :

Théorème 3.13. *[Courcelle et al., 2000] Si $P_1 \neq NP_1$, alors il existe un problème* $\textsc{ms}(\tau_2)$ *sur la classe des cliques qui n'est pas soluble en temps polynomial.*

(C'est une question ouverte que de remplacer l'hypothèse $\mathrm{P}_1 \neq \mathrm{NP}_1$ par $\mathrm{P} \neq \mathrm{NP}$ dans l'énoncé de ce théorème.)

Comme les cliques sont de largeur de clique $\leqslant 2$, on ne peut pas utiliser les classes de largeur de clique bornée pour étendre le théorème d'algorithmique. On connaît cependant un autre paramètre de complexité, appelé largeur arborescente, pour lequel l'homologue de ce théorème s'applique (les cliques n'ont donc pas une largeur arborescente bornée). En fait ma présentation, centrée sur la largeur de clique, est à rebours de l'ordre dans lequel ces notions ont été découvertes : la largeur arborescente, plus ancienne, est un paramètre sur lequel on dispose maintenant de nombreuses informations (voir par exemple [Kloks, 1993] pour un exposé clair des définitions, [Bodlaender, 1993] et [Bodlaender, 1998] pour une compilation des résultats connus).

Définition 3.14. On appelle k-**clique**[8] une clique de taille k.

Définition 3.15. Soit $k \geqslant 1$. Un k-**arbre** est un graphe admettant une séquence d'élagage des sommets dont le voisinage est une k-clique, dont le graphe élagué est une k-clique.

Un k-arbre est en particulier un graphe triangulé, de taille de clique maximum $k+1$; un 1-arbre est un arbre au sens usuel.

Définition 3.16. Un k-**arbre partiel** est un graphe pouvant être obtenu à partir d'un k-arbre en enlevant des arêtes.

[8] Dans certains contextes le terme de « k-clique » ferait référence à une clique *maximale* de taille k dans un graphe donné ; nous emploierons ici ce terme sans cette restriction.

Définition 3.17. [Robertson et Seymour, 1986a] La **largeur arborescente** d'un graphe G, notée $\mathrm{tw}(G)$ (en référence au terme anglais « tree-width »), est le plus petit entier k tel que G soit un k-arbre partiel.

Un graphe G de largeur arborescente k peut être représenté à l'aide de sous-arbres d'un arbre : il existe un arbre A, une fonction R associant à chaque sommet un sous-arbre de A (un ensemble connexe de nœuds), tels que

$$(\forall x, y \in V(G))(x{-}y \Rightarrow R(x) \cap R(y) \neq \emptyset),$$

et tel qu'au plus $k+1$ sommets soient associés à un nœud donné. En fait cette représentation[9] est une autre caractérisation des k-arbres partiels ; elle s'appelle **décomposition arborescente**, et elle joue le même rôle qu'une k-expression pour la largeur de clique.

Théorème 3.18. *[Bodlaender, 1996] Pour $k \geqslant 1$ fixé, il existe un algorithme linéaire calculant la décomposition arborescente d'un graphe de largeur arborescente $\leqslant k$.*

Ce théorème n'est pas vrai si l'on ne fixe pas k : le calcul de la largeur arborescente est NP-complet en général [Arnborg et al., 1987].

Théorème 3.19. *[Courcelle, 1990, Courcelle et Mosbah, 1993, Arnborg et al., 1991] Pour tout entier k fixé, pour tout problème de Lin$\mathrm{MS}(\tau_2)$, il existe un algorithmique polynomial qui résout ce problème sur la classe des graphes de largeur arborescente $\leqslant k$.*

Cet algorithme peut être construit à partir de l'expression du problème dans Lin$\mathrm{MS}(\tau_2)$, et il a même une complexité linéaire (en $c(k, \phi)(n+m)$).

(Les trois références proposent trois constructions d'algorithmes différentes.)

Le théorème 3.12 est donc l'exact parallèle de ce théorème, à ceci près qu'on dispose ici déjà d'un algorithme linéaire pour calculer la décomposition arborescente d'un graphe. En fait le théorème 3.12 peut même être vu comme une généralisation de ce théorème sur des classes de graphes plus grosses (mais pour des problèmes plus restreints) : en effet, si une classe de graphe est de largeur arborescente bornée, alors elle est également de largeur de clique bornée.

Théorème 3.20. *[Corneil et Rotics, 2001] Pour tout graphe G, $\mathrm{cw}(G) \leqslant 3.2^{\mathrm{tw}(G)-1}$.*

Il n'y a bien sûr aucune minoration de ce type, comme le montre l'exemple des cliques ; mais sous certaines conditions la largeur de clique d'une classe de graphe (c'est-à-dire la plus grande largeur de clique de ses membres) peut être minorée en fonction de la largeur arborescente de cette classe [Courcelle et Olariu, 2000].

En fait le théorème 3.19 admet une sorte de réciproque.

Définition 3.21. Soit $\mathscr{G}$ une classe de graphes (vue comme un ensemble de structures de type τ_2). On dit que **la théorie $\mathrm{MS}(\tau_2)$ de $\mathscr{G}$ est décidable** s'il existe un algorithme capable de décider, étant donnée $\phi \in \mathrm{MS}(\tau_2)$, si $(\forall \mathsf{G} \in \mathscr{G})\ \mathsf{G} \models \phi$.

[9] En fait il s'agit plutôt d'une représentation d'un k-arbre dans lequel se plonge le graphe G : on trouve un tel graphe en mettant l'arête $x{-}y$ chaque fois que $R(x) \cap R(y) \neq \emptyset$ (transformant l'implication en équivalence). Il faut donc garder les informations décrivant G en plus de sa décomposition arborescente.

Théorème 3.22. *[Seese, 1991] Si $\mathscr{G}$ a une théorie* $\mathrm{MS}(\tau_2)$ *décidable, alors $\mathscr{G}$ est de largeur arborescente bornée.*

Les classes de graphes de largeur arborescente bornée surgissent donc de manière naturelle lorsque l'on explore les rapports entre la logique et l'algorithmique de graphes ; on se demande si ce n'est pas aussi le cas des classes de graphes de largeur de clique bornée (en adaptant la définition précédente) :

Conjecture 3.23. *[Seese, 1991, Courcelle et Engelfriet, 1995] Si $\mathscr{G}$ a une théorie* $\mathrm{MS}(\tau_1)$ *décidable, alors $\mathscr{G}$ est de largeur de clique bornée.*

Un certain nombre de résultats partiels en ce sens sont établis [Courcelle, 1995].

3.2.5 Voies de recherche actuelles

Techniques de minoration

Comme on l'a vu, les résultats concernant la largeur de clique sont parallèles à ceux concernant la largeur arborescente, sauf que les théorèmes se transforment en conjectures ou questions ouvertes. C'est peut-être que l'on ne connaît pas d'équivalent à certains outils techniques ; en particulier [Robertson et Seymour, 1986b] montre que les graphes de grande largeur arborescente contiennent de grandes grilles carrées comme mineurs, mais l'on n'a pas de telles caractérisations pour la largeur de clique. Pour l'instant la meilleure technique connue pour borner inférieurement la largeur de clique d'un graphe n'est pas applicable dans le cas général :

Théorème 3.24. *[Gurski et Wanke, 2000] Un graphe G sans $K_{p,p}$ (non nécessairement induits) a une largeur arborescente d'au plus* $3(p-1)\mathrm{cw}(G) - 1$.

Cette technique permet tout de même de montrer que des variations sur le thème de la grille ont de grandes largeurs de clique.

Réécriture

On a déjà expliqué que l'absence d'algorithme polynomial de calcul d'une k-expression est un obstacle à l'utilisation du théorème d'algorithmique sur la largeur de clique. Chercher un tel algorithme, sur une classe de graphes donnée ou sur des graphes quelconques, n'est pas la seule manière de contourner cet obstacle : on peut aussi essayer de modifier le théorème pour qu'il n'y ait plus besoin de lui fournir une k-expression. Cette voie de recherche présente l'avantage d'être robuste, dans la mesure où elle resterait ouverte s'il s'avère que le calcul d'une k-expression est NP-complet pour certaines valeurs de k ; mais elle nécessite que l'algorithme construit par le théorème modifié ne soit pas basé sur un parcours d'une k-expression, c'est-à-dire que les principes algorithmiques soient entièrement revus.

Cette voie a déjà été empruntée dans le cas de la largeur arborescente : [Arnborg et al., 1993] montre que tout problème de décision $\mathrm{MS}(\tau_2)$ admet un autre type d'algorithme polynomial (linéaire en temps mais pas en espace) sur les classes de graphes de largeur arborescente bornée. Cet algorithme consiste à remplacer localement des sous-graphes par d'autres sous-graphes, selon des règles de réécriture, jusqu'à ce qu'aucune règle ne s'applique, puis à regarder si le graphe final appartient à un certain ensemble ; et pour tout problème $\mathrm{MS}(\tau_2)$, pour tout k, il existe un ensemble fini de règles de réécritures et de graphes finaux, tel que l'algorithme soit correct si le graphe d'entrée est bien de largeur arborescente $\leqslant k$. (Là aussi, la construction de l'algorithme, c'est-à-dire la détermination des ensembles de règles et de graphes finaux, est prohibitive en général.) Cet algorithme court-circuite donc complètement le recours à une décomposition du graphe fourni.

Il ne semble pas que ce genre de résultats soit connu pour la largeur de clique : les techniques de réécriture sont pour l'instant cantonnées aux graphes de largeur arborescente bornée. Ces techniques forment pourtant un outil important : par exemple, [Espelage et al., 2001] s'en sert pour montrer que l'on peut calculer en temps polynomial la largeur de clique d'un graphe, si l'on se place sur une classe de largeur arborescente bornée.

Approximation

En fait, même si le calcul d'une k-expression s'avère NP-complet sur la classe des graphes de largeur de clique $\leqslant k$, rien n'interdit d'imaginer que l'on puisse tout de même appliquer le théorème d'algorithmique 3.12. En effet ce calcul correspond à l'optimisation d'un paramètre, alors que pour appliquer le théorème, il suffit de disposer d'une approximation de cette valeur optimale : il suffit de disposer d'un algorithme calculant une $f(k)$-expression, pour une fonction f *quelconque*. En particulier, il n'est pas nécessaire que l'on sache reconnaître en temps polynomial qu'un graphe est de largeur de clique $\leqslant k$: l'algorithme aurait la possibilité, étant donné un graphe qui n'est pas de largeur de clique $\leqslant k$, de trouver quand même une $f(k)$-expression.

Montrer que le calcul d'une k-expression est NP-complet serait finalement un résultat assez limité : ce serait juste que la voie la plus simple est fermée.

Effectivité

Le problème de l'effectivité de la théorie n'a pour l'instant pas été abordé, à ma connaissance, dans le cas de la largeur de clique. Bien sûr, il serait trop beau que l'on puisse disposer d'un système construisant automatiquement un algorithme efficace à partir d'une expression logique ; mais on aimerait quand même faire mieux que se contenter de dire à l'algorithmicien « Il existe une solution linéaire ; à toi de la trouver. ».

(a) Grossièreté des bornes

On a vu que le principe de cette théorie est qu'un problème à résoudre sur un k-graphe passe au quotient selon les opérations de compositions : il se transforme en un ou plusieurs problèmes à résoudre sur de nouveaux k-graphes lorsque l'on descend le long de l'arbre de décomposition. La constante de proportionnalité de l'algorithme entier, $c(k, \phi)$, est essentiellement le nombre de problèmes distincts que l'on peut rencontrer ainsi ; on s'est contenté de majorer ce nombre par $c(k, h(\phi))$, le nombre total de formules, à équivalence tautologique près, sur des k-graphes, de hauteur de quantification $h(\phi)$. On a dit que cette majoration est grossière, et voici un argument : on va montrer qu'un nombre aussi grand que $c(k,h)$ est une arme bien plus puissante que ce que l'on a vu jusqu'à maintenant pourrait laisser croire ! On peut en effet transformer légèrement l'algorithme présenté dans [Rotics, 1998] pour qu'il fonctionne de la manière suivante :

- On note $\mathscr{P}_{kh}$ l'ensemble des problèmes $\mathrm{Lin}\textsc{ms}(\tau_1)$ de hauteur de quantification h sur des k-graphes, à équivalence tautologique près : cet ensemble a une taille $c(k,h)$. (Bien sûr l'algorithme ne manipule pas ces problèmes en tant que tels : de son point de vue un problème est un entier de $[\![1, c(k,h)]\!]$, c'est-à-dire que l'on utilise un codage quelconque ; et les transformations de problèmes sont des manipulations d'entiers.)
- Trouver toutes les solutions de tous les problèmes de $\mathscr{P}_{kh}$ sur les graphes à un sommet (cela peut être précalculé) ;
- faire remonter ces solutions le long de l'arbre de décomposition : cela prend un temps linéaire ;
- *obtenir ainsi toutes les solutions de tous les problèmes* $\mathscr{P}_{kh}$ sur le graphe fourni en entrée. Il ne reste plus qu'à fournir à l'algorithme le code des problèmes dont on cherche la solution.

Comme résoudre un seul problème est *a priori* plus simple que tous les résoudre, on s'attend à ce que $c(k,\phi)$ soit nettement inférieur à $c(k,h(\phi))$ [10].

Cela suggère deux directions de recherche pour résoudre ce problème d'effectivité.

(b) De nouvelles classes de formules

Premièrement, sur un plan encore théorique, on peut essayer de décrire des classes de formules pour lesquelles on sait montrer que $c(k,\phi)$ n'est pas trop élevé : cette direction produirait des raffinements du théorème d'algorithmique 3.12, dans la mesure où la constante de proportionalité serait également maîtrisée.

Peut-être plus important, elle ouvrirait la voie à une nouvelle classification des problèmes, grâce à l'introduction de ce nouveau paramètre $c(k,\phi)$: peut-être que certains problèmes sont intrinsèquement difficiles, dans la mesure où, pour les résoudre, il faut savoir résoudre une grande variété de problèmes différents. Cette approche est bien sûr très naïve, puisqu'il n'est pas forcément difficile de résoudre une grande variété de problèmes faciles ; mais, quant au fait qu'il y a peut-être un paramètre intrinsèque à découvrir, nous avons pour indice la conjecture 3.23, selon laquelle les classes de largeur de clique bornée sont des objets naturels. Cela ne signifie pas que la largeur de clique soit elle-même un paramètre naturel (n'importe quel paramètre « à variation finie » par rapport à la largeur de clique fournit les mêmes classes bornées ; c'est juste que la largeur de clique est le paramètre qui semble le plus simple du point de vue des manipulations de formules), mais peut-être est-il possible de donner un sens à $c(k,\phi)$ qui ne soit pas dépendant du « modèle de calcul » utilisé.

[10] Et même, en remarquant que l'équivalence tautologique est trop fine pour nos besoins, à ce que $c(k,\phi)$ soit nettement inférieur à $c'(k,h(\phi))$, le nombre de classes de formules distinguables par les graphes de largeur de clique $\leqslant k$. En effet, il est clair que, dans l'algorithme que l'on vient de voir, on peut se contenter d'un ensemble de formules de cette taille.

(c) Extension à d'autres problèmes

Deuxièmement, pour en revenir à la question initiale (on désire résoudre un problème donné en pratique), on peut effectuer les passages au quotient à la main, et trouver ainsi le meilleur $c(k, \phi)$ possible.

On pourrait objecter que toute la théorie ne sert finalement à rien en pratique : constatant qu'une technique classique pour résoudre un problème consiste à le faire passer au quotient selon une certaine décomposition, on fait toute une théorie sur une catégorie de problèmes qui passent au quotient selon une catégorie de décompositions, espérant par là passer d'une approche au cas par cas à une approche plus générale, pour finalement dire que tout le travail reste à faire si l'on considère un cas donné. C'est que, encore une fois, cette théorie n'est pas effective : elle ne fait que proposer une méthode, et décrire formellement les situations où l'on sait qu'elle fonctionne, selon une définition théorique du terme. Elle ne peut pas remplacer magiquement des années de travail par un humain : elle ne peut que le guider dans sa recherche.

En fait, on peut même dire que la théorie actuelle se sous-estime, et une autre voie de recherche consiste à étendre la gamme des problèmes qui sont solubles en temps polynomial sur les classes de largeur de clique bornée ; on peut en effet se demander pourquoi la théorie serait limitée à la production d'algorithmes linéaires. Cette approche a été envisagée au cas par cas : par exemple, le problème du nombre chromatique [Kobler et Rotics, 2001] ; ou pour des classes de problèmes présentant certaines propriétés : par exemple la classe que propose [Wanke, 1994][11] contient le chemin hamiltonien. Ces algorithmes souffrent encore des mêmes défauts que les algorithmes construits par le théorème d'algorithmique, c'est-à-dire que les bornes sont trop grossières : par exemple, la complexité de l'algorithme de calcul du nombre chromatique est analysée par les auteurs, et elle fait intervenir notamment un facteur $n^{2^{2k+1}}$. Il semble que la recherche soit actuellement orientée vers ce type de résultats théorique, et peut-être faudra-t-il qu'un algorithme de décomposition soit exhibé pour que, les applications pratiques étant à portée de main, la question de l'effectivité des algorithmes prenne de l'importance.

(d) Structures sous-jacentes

On devrait pourtant pouvoir faire mieux que dire à l'algorithmicien de graphes « Fais passer ton problème au quotient selon les opérations de composition. » : on devrait pouvoir lui fournir une structure, qui soit un objet naturel d'algorithmique de graphes, qu'il puisse utiliser pour effectuer son passage au quotient. Autrement dit, nous devrions chercher une description globale des graphes de largeur de clique $\leqslant k$, à utiliser pour résoudre les problèmes $\mathrm{MS}(\tau_1)$, au lieu de dépendre d'une description technique : il s'agit de fournir à la théorie, qui a un pied dans la logique, un deuxième pied dans les structures, pour en faire une théorie qui marche.

Cette unification a déjà été réalisée dans le cadre de la largeur arborescente, grâce à la notion de k-arbre partiel, mais rien de tel n'existe encore pour la largeur de clique : il faut

[11]En fait, si l'on regarde bien les définitions, montrer qu'un problème appartient à cette classe revient à montrer qu'il passe au quotient, c'est-à-dire qu'il faut le quotienter à la main : il ne s'agit donc pas vraiment d'une caractérisation de problèmes pour lesquels un passage au quotient est possible.

se débrouiller avec un arbre de décomposition. D'ailleurs, le fait que nous ne sachions pas généraliser notre algorithme de décomposition des graphes de largeur de clique $\leqslant 3$ vient, à mon avis, de ce que nous n'avons pas bien décrit la structure des graphes de largeur de clique $\leqslant 3$: il me semble que nous tâtonnons plutôt à l'aveuglette. De là vient la complexité (au sens usuel) de notre algorithme, et notre ignorance de ce qu'il nous faut chercher pour les largeurs de clique supérieures.

C'est pourquoi ce chapitre se concluera sur la question de la structure des graphes de largeur de clique $\leqslant 3$.